2001

NATURAL ALGORITHMS IN DIGITAL FILTER DESIGN

PENBERTHY HARRIS, STEPHEN

NATURAL ALGORITHMS IN DIGITAL FILTER DESIGN

by

STEPHEN PENBERTHY HARRIS, MA (Oxon)

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

Department of Communication and Electronic Engineering

May 2001

NATURAL ALGORITHMS IN DIGITAL FILTER DESIGN

Stephen Penberthy Harris

Digital filters are an important part of Digital Signal Processing (DSP), which plays vital rôles within the modern world, but their design is a complex task requiring a great deal of specialised knowledge. An analysis of this design process is presented, which identifies opportunities for the application of optimisation.

The Genetic Algorithm (GA) and Simulated Annealing are problem-independent and increasingly popular optimisation techniques. They do not require detailed prior knowledge of the nature of a problem, and are unaffected by a discontinuous search space, unlike traditional methods such as calculus and hill-climbing.

Potential applications of these techniques to the filter design process are discussed, and presented with practical results. Investigations into the design of Frequency Sampling (FS) Finite Impulse Response (FIR) filters using a hybrid GA/hill-climber proved especially successful, improving on published results. An analysis of the search space for FS filters provided useful information on the performance of the optimisation technique.

The ability of the GA to trade off a filter's performance with respect to several design criteria simultaneously, without intervention by the designer, is also investigated. Methods of simplifying the design process by using this technique are presented, together with an analysis of the difficulty of the non-linear FIR filter design problem from a GA perspective. This gave an insight into the fundamental nature of the optimisation problem, and also suggested future improvements.

The results gained from these investigations allowed the framework for a potential 'intelligent' filter design system to be proposed, in which embedded expert knowledge, Artificial Intelligence techniques and traditional design methods work together. This could deliver a single tool capable of designing a wide range of filters with minimal human intervention, and of proposing solutions to incomplete problems. It could also provide the basis for the development of tools for other areas of DSP system design.

Contents

Abstract ii

Table of Contents iii

List of Figures vi

List of Tables viii

Acknowledgements x

Author's declaration xi

1 Introduction 1
- 1.1 Digital filters . . . 1
- 1.2 Natural Algorithms . . . 2
- 1.3 Publications . . . 2
- 1.4 Aims and objectives . . . 3

2 Basic Filter Design Theory 4
- 2.1 Stages in Filter Design . . . 4
- 2.2 Finite Impulse Response Filters . . . 5
  - 2.2.1 Frequency Sampling FIR filters . . . 6
  - 2.2.2 Linear phase FIR filters . . . 8
  - 2.2.3 Recursive Frequency Sampling FIR Filters . . . 9
  - 2.2.4 Other FIR Design Techniques . . . 10
    - 2.2.4.1 Optimal Method . . . 10
    - 2.2.4.2 Window Method . . . 11
- 2.3 Infinite Impulse Response Filters . . . 11
  - 2.3.1 IIR Filter Theory . . . 12
    - 2.3.1.1 Alternative structures . . . 14
  - 2.3.2 Design methods . . . 16
  - 2.3.3 Quantisation effects . . . 17
  - 2.3.4 Error Spectral Shaping . . . 19
  - 2.3.5 Coefficient pairing and ordering . . . 19
- 2.4 The Rôle of Optimisation in Filter Design . . . 21
  - 2.4.1 Specification . . . 21
  - 2.4.2 Coefficient Calculation . . . 22
  - 2.4.3 Structure Realisation . . . 22
  - 2.4.4 Analysis of Finite Wordlength Effects . . . 23

2.4.5 Implementation 24

3 An Introduction to Genetic Algorithms and Other Natural Algorithms 26
3.1 Background to the GA 26
3.2 Outline of a Standard Binary-Coded Algorithm 27
3.2.1 Encoding the problem 28
3.3 Initialisation 29
3.4 Selection 30
3.5 Reproduction 31
3.6 Crossover 31
3.7 Mutation 33
3.7.1 Mutation and parameter encoding 34
3.7.2 Other forms of mutation 35
3.8 The Fitness Function 36
3.8.1 Fitness Scaling 36
3.9 Advantages and Disadvantages of the GA 37
3.10 Convergence Theory 38
3.11 Floating-point Chromosome GA 39
3.11.1 Floating-point Crossover 40
3.11.2 Floating-point Mutation 43
3.12 Other GA techniques 44
3.13 Hybridisation 45
3.14 The Search Space 46
3.15 Simulated Annealing 47
3.15.1 Applications to Filter Design 48
3.16 Differential Evolution 48
3.17 Evolutionary Strategy 49
3.18 Genetic Programming 49
3.19 Tabu Search 50

4 Optimising Frequency Sampling Filter Coefficients by Hybrid GA 52
4.1 Introduction 52
4.1.1 Selection of FS filters for GA optimisation 52
4.2 Use of the GA for FS filter Design 53
4.2.1 The Fitness Function 54
4.3 Extensions to the Floating-Point GA 55
4.4 Simplex method hybrid hill-climber 55
4.5 Extensions to the crossover selection scheme 57
4.6 Results for FS filters 60
4.6.1 The FIR Filter search space 66
4.6.2 Concurrent Optimisation of the Wordlength 69
4.7 Conclusions 73

5 IIR Coefficient Optimisation by GA and SA 75
5.1 Introduction 75
5.2 Use of the GA 76
5.3 Results 78

5.3.1 The IIR Filter Parameter Space 80
5.4 Discussion and Conclusions 81

6 Multi-criterion Optimisation **84**
6.1 Introduction 84
6.2 Techniques 84
6.2.1 Weighted sum of fitnesses 85
6.2.2 The Pareto-optimal set 86
6.2.3 Vector-Evaluated GA 88
6.2.4 Goldberg's fitness allocation method 89
6.3 Applications of MCO optimisation to filter design 92
6.4 GA difficulty measures and deception 93
6.4.1 Epistasis 93
6.4.2 Fitness-distance correlation 95
6.5 Alterations to the GA 96

7 An Analysis of the Suitability of GA-based Optimisation for Non-linear Phase FIR Filter Design **97**
7.1 Introduction 97
7.2 Non-linear phase FIR filters 98
7.2.1 Effects of coefficient quantisation 100
7.3 Use of the GA 102
7.3.1 Design performance 103
7.4 Analysis of non-linear FIR filter design 106
7.4.1 The parameter space 106
7.5 Measures of GA-difficulty 108
7.5.1 Epistasis 110
7.5.2 Fitness-distance correlation 112
7.6 Results 114

8 An Extended Multi-objective GA for IIR Filter Design **118**
8.1 Introduction 118
8.2 Chromosome design 119
8.3 The fitness function 121
8.4 Effects of quantisation 123
8.4.1 Coefficient quantisation 124
8.4.2 Noise 125
8.5 Filter structure 125
8.6 Use of the GA 127
8.7 Results 127
8.8 Discussion 140

9 Conclusion and Future Work **142**
9.1 Review 143
9.1.1 Digital Filter Design 143
9.1.2 Finite Impulse Response Filters 143
9.1.3 Infinite Impulse Response Filters 145
9.1.4 Optimisation techniques 148

9.2 Future Work . . . 148
9.2.1 New areas in FIR Filters . . . 148
9.2.2 New areas in IIR Filters . . . 150
9.2.3 Further Natural Algorithm Techniques . . . 151
9.2.3.1 Genetic Algorithms . . . 151
9.2.3.2 Simulated Annealing . . . 152
9.2.3.3 Tabu Search . . . 152
9.2.3.4 Genetic Programming . . . 153
9.3 Intelligent Filter Design Tool . . . 153
9.4 Conclusion . . . 156

A Techniques **165**
A.1 Increased calculation efficiency for recursive FIR filters . . . 165
A.2 Simplex method hill-climber . . . 166
A.3 Matlab initialisation script for MCO IIR design . . . 168
A.4 Single-criterion Genetic Algorithm . . . 171
A.5 Multi-criterion Genetic Algorithm . . . 176

B Publications **178**

List of Figures

2.1 Stages in the design of a digital filter. 5
2.2 Samples of the frequency response of a lowpass filter. 7
2.3 Samples of the frequency response with transition samples. 8
2.4 Frequency sample positions for FS filters. 9
2.5 Second-order direct form 1 IIR filter section. 13
2.6 Second-order canonic (or 'direct form 2') IIR filter section. 13
2.7 Cascaded second-order canonic filter sections with signal scaling. 15
2.8 Example of a lattice-structure IIR filter. 15
2.9 Canonic filter section with signal scaling and error spectral shaping. . . 20

3.1 Illustration of constant binary encoding. 29
3.2 Example Arithmetic crossover for a two-gene chromosome. 42
3.3 Example Whole Arithmetic crossover for a two-gene chromosome. . . . 43

4.1 The allowed region of the search space. 54
4.2 Dynamic crossover selection method. 58
4.3 Type I, Highpass FIR filter with four transition samples. 62
4.4 Type II, Lowpass FIR filter with four transition samples. 62
4.5 Type I, Bandpass FIR filter with three transition samples. 63
4.6 Type II, Bandstop FIR filter with four transition samples. 63
4.7 Improvements in fitness with generation. 65
4.8 Search space for a one transition sample FIR filter. 67
4.9 Search space for a two transition sample FIR filter. 67
4.10 Search space with additional constraints. 68
4.11 Improvements in constrained fitness with generation. 69
4.12 Quantised-coefficient FS filter response. 72

5.1 Response comparison for SA, GA and BZT. 79
5.2 Search space slice for perturbed IIR filter. 82
5.3 Search space slice for random coefficient filter. 82

6.1 Example illustrating pareto-optimal and non-dominated sets. 87
6.2 Example of VEGA NDS. 89
6.3 Example of NDS ranking calculations. 90
6.4 Example population distribution between niches. 91

7.1 A desired response template. 101
7.2 Illustration of the effect of coefficient quantisation on the search space. 101
7.3 Loose-tolerance test template. 104
7.4 Comparison of linear and non-linear phase filters found by GA. 105

7.5 Slice through the 1,13,LS data set, with a best fitness of 11.3dB. 109
7.6 Slice through the 1,13,P data set, which has has a best fitness of 40.5dB. 109
7.7 Slice through the 1,13,R data set, with a best fitness of 32.2dB. 109
7.8 Illustration of deception. 110
7.9 Random filter fitness plot. 115
7.10 5% perturbed fitness plot. 115
7.11 1% perturbed fitness plot. 115

8.1 Coefficient stability triangle. 120
8.2 Quantised Cartesian pole-zero positions. 122
8.3 Quantised polar pole-zero positions. 122
8.4 Chromosome structures. 126
8.5 Example of an early non-dominated set. 128
8.6 Best result for dp=0.5dB, ds=40dB target at 16 bits. 129
8.7 Best result for dp=0.75dB, ds=40dB target at 16 bits. 130
8.8 Best result for dp=1.0dB, ds=40dB target at 16 bits. 131
8.9 Best result for dp=1.0dB, ds=50dB target at 16 bits. 132
8.10 Best result for dp=1.0dB, ds=60dB target at 16 bits. 133
8.11 Best result for dp=1.0dB, ds=70dB target at 16 bits. 134
8.12 Best result for dp=1.0dB, ds=80dB target at 16 bits. 135
8.13 Best result for dp=1.0dB, ds=90dB target at 16 bits. 136
8.14 Best result for dp=1.0dB, ds=80dB target at 24 bits. 137

9.1 The potential structure of an intelligent filter design tool. 154
9.2 FIR design example 155

A.1 Steps taken by a Simplex hill-climber. 167

List of Tables

2.1 Designable FIR filters. 9

3.1 A comparison between binary and Gray coding formats. 35

4.1 Results for recursive-form FS FIR filters designed by GA. 60
4.2 Comparison of results for Type-I, $N = 16$ filter. 61
4.3 Comparison of results for Type-I, $N = 33$ filter. 61
4.4 Comparison of results for Type-I, $N = 65$ filter. 61
4.5 Comparison of results for Type-I, $N = 125$ filter. 61
4.6 Comparison of results for Type-I, $N = 128$ filter. 61
4.7 Specifications for quantised and unquantised filters. 72
4.8 Transition samples for the filters described in the text and Table 4.7. . 72

7.1 Comparison of linear and non-linear phase FIR filters. 106
7.2 The Epistasis variance (σ_ϵ^2), calculated for regions of the search space. . 111
7.3 FDC calculations for various regions of the search space. 113

8.1 Comparison of results for MCO optimisation of IIR filters. 129
8.2 Comparison of results for GA and BZT-designed IIR filters. 140

Acknowledgements

I would like to thank my Director of Studies, Professor Emmanuel Ifeachor for his support and encouragement during this project. Thanks are also due especially to Nick Outram and Rob Clark for providing invaluable suggestions and insights into the intricacies of DSP.

A degree...is a first step down a ruinous highway. You don't want to waste it so you go on to graduate work and doctoral research. You end up a thoroughgoing ignoramus on everything in the world except for one subdivisional sliver of nothing.

Ralph Nimmo in 'The Dead Past' by Isaac Asimov

Ow geryow aga honan yw gasadow ha kemmysk a furneth ha gokkineth.

Taliesin

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

A programme of advanced study was undertaken, including an analysis of the rôle of optimisation in digital filter design, and the application of Genetic Algorithms (GA) and Simulated Annealing to a range of filter design problems. The difficulty of the problem was analysed from a GA perspective, and a framework proposed for an 'intelligent' filter design tool.

Two relevant conferences were attended, papers being presented at both. A paper of results and investigations was published in a relevant IEEE journal.

Publications:

- E.C. Ifeachor and S.P. Harris, 'A New Approach to Frequency Sampling Filter Design', *Proceedings IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, 1993
- S.P. Harris and E.C. Ifeachor, 'Automating IIR Filter Design by Genetic Algorithm', *Proceedings GALESIA International Conference*, 1995
- S.P. Harris and E.C. Ifeachor, 'Automatic Design of Frequency Sampling Filters by Hybrid Genetic Algorithm Techniques', *IEEE Transactions on Signal Processing*, volume 46(12), pp3304–3314, 1998

Conferences attended:

- 1993 IEE/IEEE Workshop on Natural Algorithms in Signal Processing
- 1995 GALESIA Internation Conference

Signed ...S P Harris.....

Dated ...16/12/2001.....

Chapter 1

Introduction

Digital Signal Processing (DSP) plays a crucial part in the modern world. Its use in devices ranging from mobile phones and home entertainment systems to medical equipment means that we are increasingly reliant on it in all walks of life. Digital filters are an important part of DSP, but they have a major drawback in that it requires a great deal of specialised knowledge to design them successfully, and this is not always easily available. This investigation will examine potential applications of so-called 'Natural' Algorithms, such as Genetic Algorithms and Simulated Annealing, to optimisations within the digital filter design process.

1.1 Digital filters

Within DSP, digital filters have an important rôle as frequency selectors, with many applications in the fields of audio, communications and biomedicine. Their ability to boost, remove, or otherwise adjust information within a signal makes them a powerful tool for system designers. Their flexibility allows a wide range of potential uses, from removing mains frequency noise from sensitive biomedical data to controlling the signal levels in audio mixing desks.

Although digital filters are extremely useful, they are generally not easy to design and require a great deal of optimisation to produce a high-quality system. Their design

is traditionally broken down into a number of steps (see Chapter 2), each of which is regarded as an independent optimisation task. The steps are repeated until a design is found which performs satisfactorily. These steps are usually performed by different methods, each of which requires a lot of specialised knowledge on behalf of the designer. This iterative approach also means that it is difficult to control the trade-off between the different performance measures, thereby making it harder to produce a filter with the desired characteristics.

An attractive aim within DSP is the development of a single, wide-ranging tool, which can be applied to a number of design problems. By encapsulating expert knowledge into a single tool, it would simplify the development of new digital systems.

1.2 Natural Algorithms

In recent years, 'Natural Algorithms' such as the Genetic Algorithm (GA) and Simulated Annealing (SA) have become increasingly popular techniques due to their applicability to a wide range of numeric and non-numeric optimisation problems [1]. The GA is based on the principles of natural selection and survival of the fittest, and works on a 'population' of possible solutions to the problem. SA was inspired by metal annealing, in which hot metal is cooled very slowly in order to allow it to find the lowest-energy crystal structure, and generally works on few or only one possible solution.

The wide-ranging applicability of GA and SA to optimisation problems suggested that they could be good techniques to use for the various aspects of digital filter design. This investigation examines the potential which the two may have in the design of digital filters, concentrating on the GA.

1.3 Publications

This work has resulted in the publication of three papers. The first, in the 1993 Proceedings of the IEE/IEEE Workshop on Natural Algorithms in Signal Processing,

introduced the design of Finite Impulse Response (FIR) filters by GA [2]. A second paper, on the application of GAs and SA to Infinite Impulse Response (IIR) filters was presented at the 1995 GALESIA conference [3]. The third paper, an extension to the work presented in the first, was published in the IEEE Transactions on Signal Processing in 1998 [4].

1.4 Aims and objectives

The aim of this work is to examine the rôle of optimisation within the filter design process and to identify areas where the inherent search and optimisation capabilities of Natural Algorithms could be used to advantage.

The specific objectives are to:

- undertake an analysis of the key optimisation tasks within the filter design process
- investigate the application of GA and SA techniques to these optimisation tasks
- undertake an analysis of the suitability of the chosen algorithms for their selected design tasks
- specify a framework for an automated filter design tool

Following a brief introduction to filter design, and a more in-depth description of the GA, its application to filter design will be discussed, together with the techniques and analysis used to determine the suitability of the natural algorithms to digital filter design.

Chapter 2

Basic Filter Design Theory

Digital filters are found in a wide variety of situations within digital systems, where they can be used to alter the signal in specific ways, for example to reduce low or high frequency noise, or to extract data in a specific frequency range. Their diverse uses make them especially useful within the field of DSP.

There are two basic types of digital filter, Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). FIR filters have an impulse response of finite duration, while IIR filters, which have a recursive structure, have one of infinite length. IIR filters are more computationally efficient than FIR, requiring less storage space and fewer calculations, but are also harder to design and are more susceptible to finite wordlength effects. FIR filters have the added advantage that they can be made with exactly linear phase, which reduces distortion in sensitive systems.

2.1 Stages in Filter Design

It has already been mentioned that the filter design process is generally broken down into several steps, which are shown in Figure 2.1. These steps are usually performed repeatedly until an acceptable solution has been found. This approach is limited in that the performance of a good filter found during one step may be compromised during another, so that a filter may be found which has a good magnitude response but poor

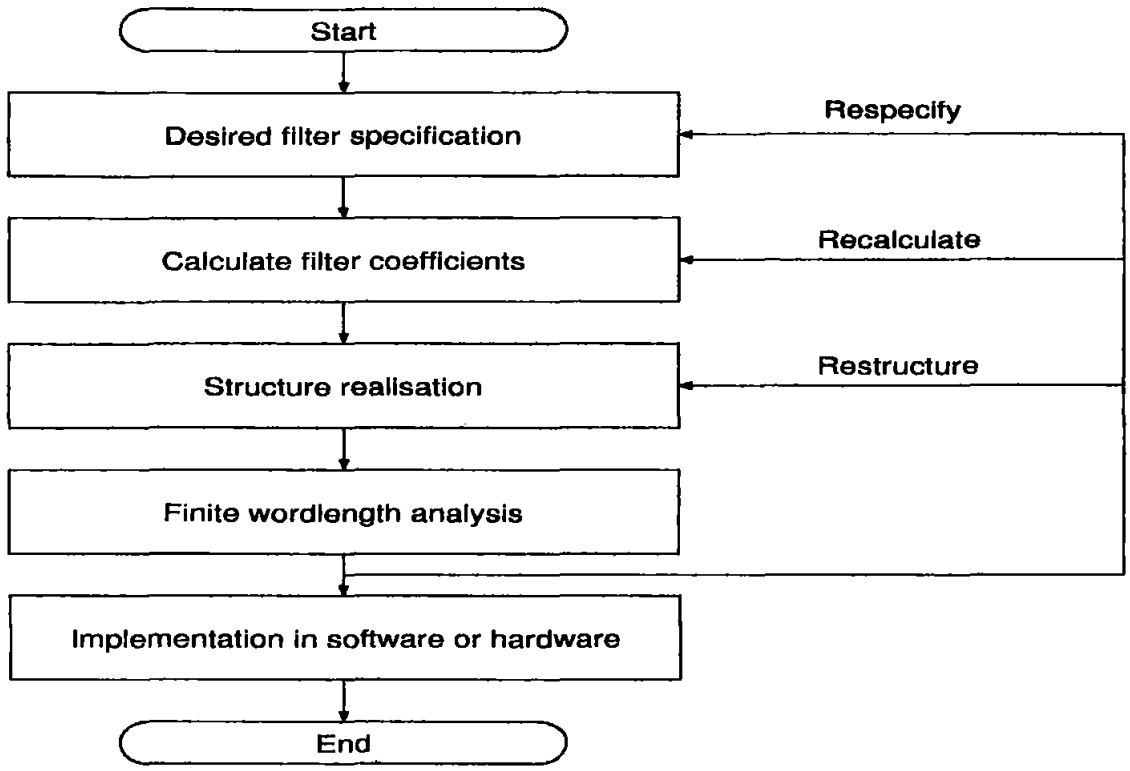


Figure 2.1: Stages in the design of a digital filter.

phase response, or *vice versa*. Part of the investigation into the use of GAs for filter design will cover the ability of the GA to perform multi-criterion optimisation, as this could potentially be used to perform several design steps simultaneously and trade off the performance measures automatically.

2.2 Finite Impulse Response Filters

Finite Impulse Response (FIR) filters are characterised by having a finite set of coefficients, and an impulse response of finite duration. FIR filters can exist in either recursive or non-recursive forms, the recursive form having the advantage that it can be much more efficient in storage and calculation, particularly for narrow band filters. FIR filters have the added advantage that they can be designed with exactly linear phase. This causes the different frequency components of the signal to be delayed by an amount proportional to their frequency, which reduces signal distortion. This is

particularly important in audio and biomedicine, where vital data may otherwise be lost. If the linear phase constraint can be relaxed, then the order for some filters can be reduced to as few as half of the coefficients needed for a fully linear-phase system. This reduction is examined in more detail in Chapter 7.

The output of an FIR filter is given by:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

where N is the filter length, $h(k)$ is the kth impulse response coefficient, x is a vector of inputs, and $y(n)$ is the nth output. This filter's transfer function, $H(z)$ is given by:

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

FIR filters are simple to design and implement, but need a large number of coefficients to achieve sharp cutoffs or high attenuations, so can be too slow to be used in high-speed or real-time situations. They also require more storage for coefficients and intermediate results than IIR, which means that their hardware implementations can be more costly.

2.2.1 Frequency Sampling FIR filters

Frequency Sampling (FS) filters are among the simplest to design, requiring the optimisation of just a few values, although this simplicity means that there is little control over the resulting coefficients.

An FIR filter can be characterised uniquely by a set of samples of the frequency response, generally taken at regular intervals, as in Figure 2.2. This set of samples gives the filter coefficients themselves by taking its inverse Discrete Fourier Transform (DFT). For nonrecursive FIR filters, the coefficient values are the same as the impulse response samples, while for the recursive FS filter, the coefficients are the frequency response samples. While taking a forward DFT of the coefficients will return the original sample values, it does not indicate the response of the filter in the intervening spaces.

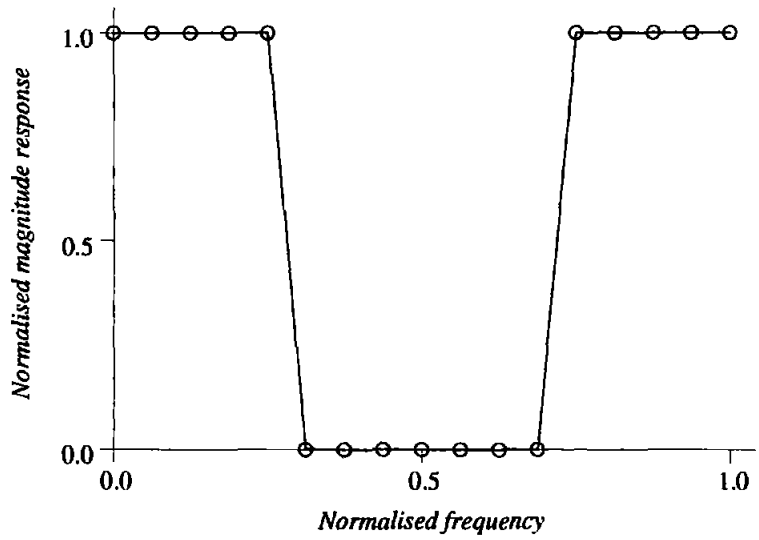


Figure 2.2: Samples of the frequency response of a lowpass filter.

To discover the filter's performance in the intervening space between the known, fixed samples, the impulse response can be zero-padded to a reasonably high number of points (512 or 1024 is generally sufficient), and a forward DFT taken, which produces an interpolated response. This high-resolution response can then be examined to determine the filter performance. The number of points is generally chosen to be a power of two so that a Fast Fourier Transform (FFT) can be used for increased calculation speed.

For a 'brick wall' type of desired response, the interpolated response is very poor. Since the sharp transition between bands does not allow any control over the degree of ripple cancellation, it should be possible to improve the filter response by including additional, variable samples to smooth off the transition between pass- and stopbands. This is illustrated in Figure 2.3.

In order to produce a filter with the best possible performance in terms of passband ripple and stopband attenuation, these *transition samples* must be optimised. Rabiner, Gold and McGonegal's seminal work in this area [5] resulted in the publication of tables of transition sample values for a range of filter types, with varying orders and bandwidths, which are still widely used today. This approach is limited by the need to interpolate transition sample values to obtain unpublished filter coefficients, which

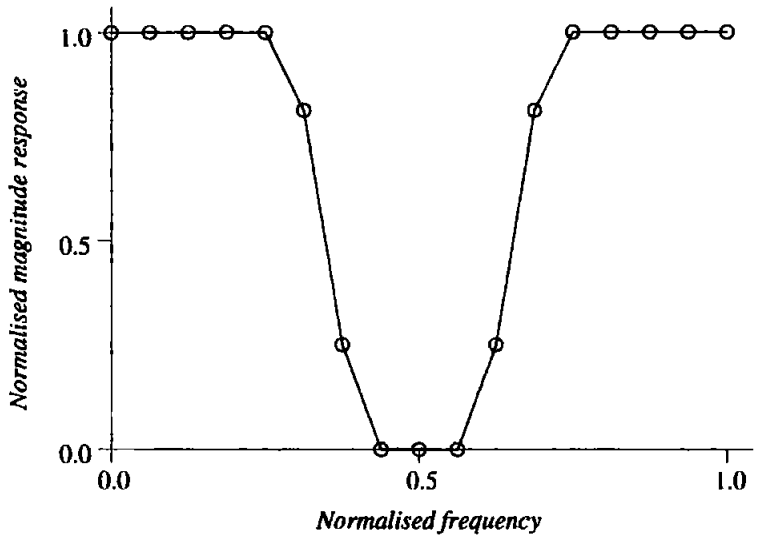


Figure 2.3: Samples of the frequency response with transition samples.

results in sub-optimal filters. The range of published filters is also limited to those with a maximum of four transition samples, for which only a few were included.

2.2.2 Linear phase FIR filters

It is often desirable for a digital filter to have *linear phase*, whereby the different frequency components of the signal are delayed by an amount proportional to their frequency. This reduces the level of distortion, which is especially important in audio [6, 7] and biomedical applications [8, 9], where the value of information may otherwise be reduced.

Linear phase can be imposed on an FIR filter by constraining the impulse response to be either symmetric or anti-symmetric, although this in turn restricts the types of filter which can be designed.

There are four 'Types' of FIR filter, which are related through having slightly different arrangements of sampling points, as shown in Figure 2.4. The first distinction between them is that the filter order, N, can be either odd or even. The second distinction is whether the first sample is taken at $\omega = 0$ or $\omega = \pi/N$. Rabiner, Gold and McGonegal [5] refer to filters similar to the former as *Type I*, and the latter as

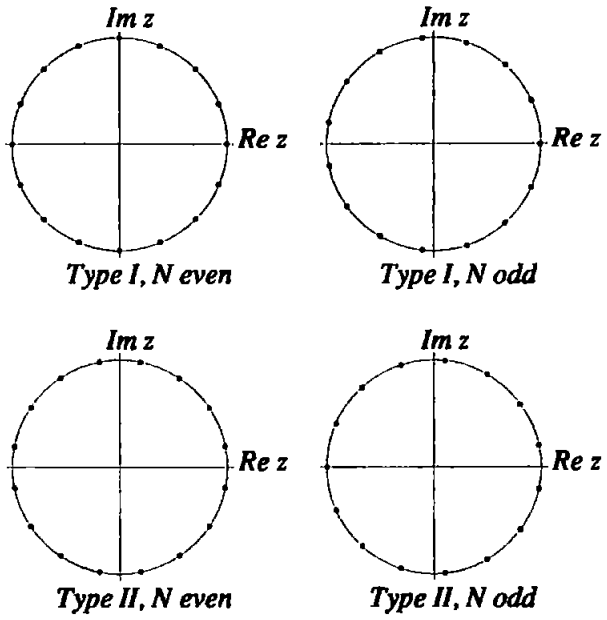


Figure 2.4: Illustrations of the z-plane positions of the frequency samples for the four types of Frequency Sampling filter.

Filter Type	Possible filters
Type-I, N odd	All filters
Type-I, N even	Lowpass and bandpass
Type-II, N odd	All filters
Type-II, N even	Lowpass and bandpass

Table 2.1: Designable FIR filters.

Type II filters.

Mathematical limitations [10] mean that different Types can only be used to design certain frequency selective sorts of filters, as shown in Table 2.1.

2.2.3 Recursive Frequency Sampling FIR Filters

It is possible to express the transfer function of any Frequency Sampling filters with either a recursive or non-recursive expression [11, 12]. The non-recursive form, which has already been described above, is less efficient relative to the recursive form for narrow-band filters where most of the frequency samples are zero (i.e. in the stopband).

Since the recursive form's coefficients are the samples themselves, if a sample is zero its effect need not be included and it therefore requires fewer calculations. It also allows a greater degree of pre-calculation to take place, and unlike the interpolation of the whole response with a DFT, just a portion of the response needs to be calculated each time, thereby increasing the computational efficiency. The pre-calculation is presented in Appendix A.1. The recursive form allows the radius of the poles and zeros to be reduced to prevent pole instability in a fixed-precision environment.

In the implementation of the recursive FS filter optimised by GA, the GA could be used to either optimise the radius simultaneously with the transition samples, or the radius could be fixed, and the corresponding optimum transition samples found.

The speed increases afforded by the use of recursive form FIR filters in the GA were significant enough for it to be adopted as the standard type of filter used in these studies.

2.2.4 Other FIR Design Techniques

A wide variety of alternative FIR design techniques exists, such as Least Squares [13, 14], but the most popular are the Optimal and Window methods:

2.2.4.1 Optimal Method

The Optimal method seeks to find a filter with the minimum maximum ripple across all pass- and stopbands. This filter will have an equiripple response, where all the magnitude response extrema have the same magnitude. As it is relatively straightforward to produce the filter response from the frequencies of these extrema, the problem lies in finding the location of the extrema in the frequency domain. This is generally achieved by using a computer program implementing the Parks-McClellan algorithm [15], which uses the Remez exchange method to search for the extremal frequencies within the equiripple response. Filters containing stopbands with different ripples can be designed by altering their weightings. The technique is widely used, but can have problems when

designing multi-band filters with varying transition widths, as convergence to a suitable solution is not guaranteed, and there may be local ripples within the transition bands. Xu and Daley [16, 17] have shown their GA to be superior to linear programming and Optimal Method coefficient roundoff techniques when designing an Optimal filter with quantised coefficients. Çiloğlu and Ünver [18] designed optimal filters with finite-wordlength coefficients using Simulated Annealing.

2.2.4.2 Window Method

The Window method makes use of the fact that it is easy to define an ideal impulse response for a desired brick-wall style filter response [11]. However, if this infinite-precision, infinite-length response is truncated and sampled using a finite number of samples to produce FIR filter coefficients, the corresponding filter's frequency response will have an excessive amount of ripple. The Window method seeks to overcome this by multiplying the ideal response by a *Window function*, which seeks to reduce the effects of the truncation by gradually reducing the impulse response to zero within the selected number of samples. A variety of functions have been proposed, such as the Hamming and Kaiser, with different characteristics, but none give the designer precise control over the band-edges and ripples of the filters they produce, which can make them unsuitable for critical applications. Keane et al [19] have used Genetic Programming (GP), a variation on the GA able to optimise expressions, to find an impulse response function for a control system. Applying this method to filter design could improve the quality of Window method filters by allowing the window function itself to be optimised, rather than being constrained to the limited range of standard functions.

2.3 Infinite Impulse Response Filters

IIR filters are distinguished from FIR by having an infinite duration impulse response. They exist only in recursive forms, where the filter output is dependent not only on the

previous and current inputs, but also the previous outputs. They are generally designed using second- and first-order sections, which are less susceptible to finite wordlength effects than large single structures. These low-order sections can be joined in a number of ways with different noise and finite wordlength characteristics.

Due to their recursive nature, IIR filters are able to have a much sharper cut-off and a higher attenuation than FIR filters, for significantly fewer coefficients. They do however have a major disadvantage in that they cannot be forced to have exactly linear phase. With the correct design constraints, however, it may be possible to design filters with a near-linear response over a limited region.

When used in a fixed precision environment, IIR filters are more susceptible to finite wordlength effects, such as noise and instability, than FIR, although by varying the structure these effects can be reduced. Their recursive nature and finite wordlength sensitivity also means that IIR filters are less straightforward to design.

2.3.1 IIR Filter Theory

IIR filters, like FIR filters, have a finite set of coefficients, but they are no longer the same as the impulse response, and are used in a different way to determine the filter's performance.

The output of an IIR filter can be described by:

$$y(n) = \sum_{k=0}^{N} a_k x(n-k) - \sum_{k=1}^{M} b_k y(n-k) \tag{2.1}$$

where a_k and b_k are the coefficients of the filter, $x(n)$ and $y(n)$ are its input and output streams respectively, and N and M are the number of a_k and b_k filter coefficients, with $M \geq N$. It can be seen from this that the current output, $y(n)$, is a function not only of the current and previous inputs, $x(n-k)$, but also the past outputs, which gives the filters their recursive character.

The equivalent direct transfer function, $H(z)$, is given by:

$$H(z) = \frac{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}}{1 + b_1 z^{-1} + \ldots + b_M z^{-M}} \tag{2.2}$$

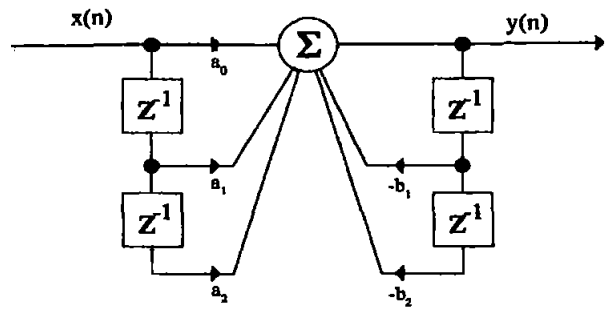


Figure 2.5: Second-order direct form 1 IIR filter section.

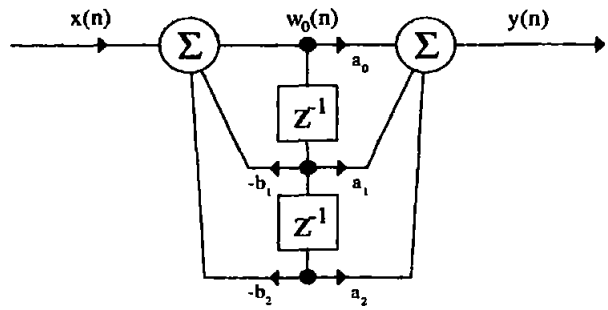


Figure 2.6: Second-order canonic (or 'direct form 2') IIR filter section.

This form of the IIR filter could be designed and implemented directly, but it is extremely sensitive to finite wordlength effects, even at filter orders as low as $M = 4$, so it is not commonly used. In order to produce a more stable system, the filter is usually broken down into *first-* and *second-order sections*, which are optimised simultaneously in one of a variety of series and parallel topologies to produce the final filter. The two main types of second-order sections are shown in Figures 2.5 and 2.6

The canonic (or direct form 2) section requires less storage than the direct form 1 section, but due to the existence of two adders, requires *input scaling* to prevent overflows on the output. This can require considerable additional calculations when determining the noise performance of the filter so can be less attractive at the design stage. The direct form requires no such scaling, due to the cyclic overflow nature of two's complement arithmetic, which can allow an intermediate overflow and still return the correct output provided the overflow is reversed by a later calculation.

When factored into second-order sections, $H(z)$ is given by:

$$H(z) = \prod_{k=1}^{N/2} \frac{a_{0k} + a_{1k}z^{-1} + a_{2k}z^{-2}}{1 + b_{1k}z^{-1} + b_{2k}z^{-2}} \tag{2.3}$$

The positions of the roots of the numerators and denominators give the positions of the *zeros* and *poles* respectively. When the value of z approaches that of a zero, the numerator approaches zero, and therefore so does the output. Zeros therefore generally define the location of the stopband. When z is close in value to a root of the denominator, i.e. a pole, the denominator evaluates to a very small number, so the division results in a large output. Poles therefore generally occur in the passband. In pole-zero form, with complex conjugate pole-zero pairs, $H(z)$ is given by:

$$H(z) = \prod_{k=1}^{N/2} \frac{(z - r_k^o e^{j\omega_o})(z - r_k^o e^{-j\omega_o})}{(z - r_k^p e^{j\omega_p})(z - r_k^p e^{-j\omega_p})} \tag{2.4}$$

where r_k^o and r_k^p are the radii of the kth zero and pole respectively, and ω_o and ω_p are their angles. If the radius of a pole is too large this can result in an overflow, so the pole radius has to be carefully controlled to be no greater than unity. In a finite wordlength system, the radius has to be reduced further in order to prevent perturbations caused by quantisation pushing poles close to or outside the unit circle, with a shorter bit-length requiring a smaller radius. It is safe for zeros to occur at any radius.

Within the canonic (or direct form 2) section shown in Figure 2.6, there are two adders, one in a feedback path. Unlike the direct form 1 section, intermediate overflows in the output of the first adder (shown as $w(n)$) can be passed on to the second, and from there to the output, giving an overall incorrect result. To counter this, the input signal must be scaled down to prevent overflows in $w(n)$, and the signal entering the second adder must be scaled up again to restore the output level to its correct value. A filter made up of cascaded canonic sections is shown in Figure 2.7.

2.3.1.1 Alternative structures

Many other possible structures exist, a popular one being the lattice structure, which has been used in speech processing [20]. An example two pole lattice structure is given

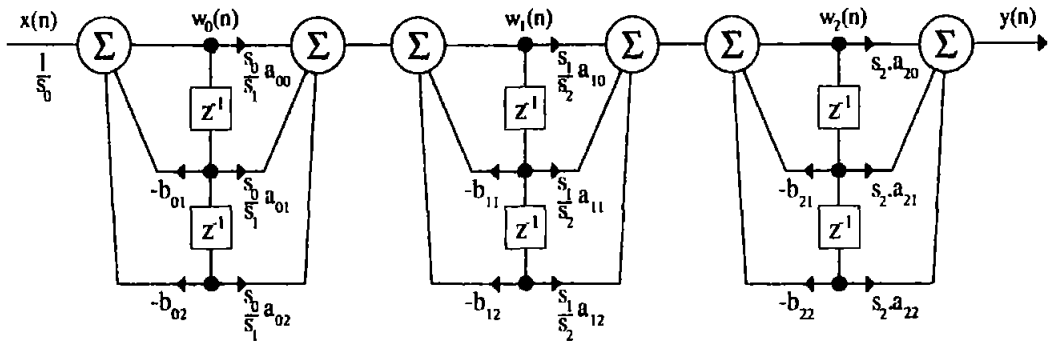


Figure 2.7: Cascaded second-order canonic filter sections with signal scaling.

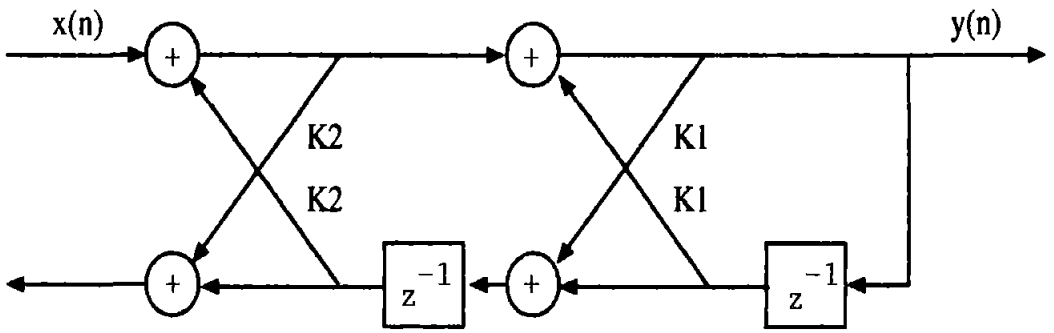


Figure 2.8: Example of a lattice-structure IIR filter.

in Figure 2.8.

The lattice structure has the attractive properties that the filter is guaranteed to be stable if the magnitude of the K coefficients are all less than unity, and that it is less affected by coefficient roundoff than the direct or cascade structures. Flockton and White [21, 22] have successfully applied the GA to the problem of adaptive system identification. This involves using the GA to optimise the quantised coefficients of a lattice filter based on a continuous input, so that its output approaches the output of an unknown system, in order to identify and model it. Sriranganathan et al [23] have applied GAs to the optimisation of lattice filter coefficients limited to simple sums of powers-of-two values for simpler implementation. Both of these approaches intrinsically take account of coefficient quantisation, which is difficult to do using standard methods. Chellapilla et al [24] have shown that the lattice structure has a much simpler search space than the direct or cascaded structures, and is suitable for a gradient-descent

algorithm when using full-precision coefficients as it is unimodal. Etter et al [25] have, however, applied the GA to a cascade-structured adaptive IIR filter with a multi-modal search space, with promising results.

A special type of filter known as an *all-pass* filter [10] can be made from standard second-order sections if the zero angle is the same angle as the pole angle, but the zero radius is the pole radius reflected in the unit circle, i.e. for every pole at $z = re^{j\omega}$ there is a zero at $z = (1/r)e^{j\omega}$. In this configuration, the magnitude response is constant, but the phase response is not. Parallel connections of all-pass sub-filters (PCAS) with other structures such as delays can produce frequency-specific filters, because the non-linear phase causes destructive interference for particular frequencies, giving the overall effect of a frequency-selective filter. All-pass filters have the advantage that they are less susceptible to finite wordlength effects than standard structures. Krukowski et al [26] present a standard method for converting any IIR filter transfer function into a sum of all-pass sections, which can be implemented efficiently in parallel for faster processing. Lawson [27], and Krukowski and Kale [28] present different standard approaches to producing frequency-specific PCAS-based filters with approximately linear phase, thereby approximating the output of a linear-phase FIR filters with many fewer coefficients, while Lu et al [30] have used Simulated Annealing (SA) to design PCAS filters with approximately linear phase. Lawson and Wicks [29] have used Simulated Annealing (SA) to design (PCAS) filters with finite wordlengths, showing that, like the GA, SA can intrinsically account for coefficient quantisation without having to optimise it in a subsequent step.

2.3.2 Design methods

A number of design methods exist for the optimisation of IIR filter coefficients. For very simple filters, experienced designers can place poles and zeros directly by inspection. This approach is fast and simple, but does require a deal of familiarity with this type of digital filter.

For more complex filters or for novice users, other techniques such as the *impulse invariant* method and *bilinear z-transform* (BZT) work by converting analogue filters into their digital equivalents.

Another design method, which does not involve the analogue domain at all, is the Least Squares method [20], which can be used when a filter is needed which most closely approximates a known impulse response. The output obtained by passing an impulse through a desired filter cascaded with its inverse is used to generate a set of linear equations which can be solved to give the set of coefficients which best match the desired filter. Kobayashi and Imai [31] propose an alternative weighted LS method for optimisation in the frequency domain, but the method is complex and is slow to converge for equiripple filters.

The Least Squares approach has similarities with *adaptive filtering*, where filter adapts its characteristics dynamically according to changes in the input signal. This allows it to, for example, give better noise reduction [11] or for system identification. Genetic Algorithms have been used for the latter by Flockton and White [21] and Etter et al [25], while Chen et al [32] have used Simulated Annealing.

Linear Programming [10, 33] involves maximising a linear function subject to a number of linear constraints. Its use in filter design requires the optimisation problem to be reworked in such a way that the problem becomes linear, and may require subsequent adjustments to the desired response to make the problem solvable. Rabiner et al [34] used the technique to optimise a range of direct-form IIR filters with respect to their magnitude-squared response, although the use of the direct form means the filters are highly susceptible to coefficient changes, and makes sharp-cutoff and high order filters difficult to design.

2.3.3 Quantisation effects

It has been seen that, due to their recursive nature, IIR filters are more sensitive to the effects of quantisation. This can affect both the coefficient values, and the results of

arithmetic operations [35]. Standard methods of design do not take these into account, and so when coefficients are quantised to the wordlength of the system, they perturb the filter characteristics. The smaller the number of bits used to represent a coefficient, the larger the perturbations will be on average, and so the more the filter response will differ from the full-precision response. It is possible to optimise the coefficients after quantising their full-precision values, but as the optimum set of quantised coefficients can be very different to the set of quantised real-valued coefficients, a technique which optimises the coefficients directly in a quantised form would be preferable.

While having quantised coefficients alters the filter response, if the system uses quantised or fixed-precision arithmetic operations, then further distortions and noise can be introduced into the signal as it passes through the filter, as the reduced precision of the calculations moves the results away from their true values. The analysis of this noise is also important in DSP as it can affect the suitability of a filter for a particular application [36]. .

It is common practice to scale the filter coefficients to help minimise or prevent overflow. For example, in 'L2 norm' scaling, which seeks to limit the power of the signal [11], the scaling factor s for second-order section i is given by:

$$s_i = \left[\sum_{k=0}^{\infty} f^2(k)\right]^{\frac{1}{2}}$$

where $f(k)$ is the impulse response from the input to the internal node w_i for section i as shown in Figure 2.7.

For this quantised, scaled, sixth-order filter, the roundoff noise gain is given by:

$$\sigma_{or}^2 = \frac{q^2}{12}\left[3\sum_{k=0}^{\infty} f_1^2(k) + 5\sum_{k=0}^{\infty} f_2^2(k) + 5\sum_{k=0}^{\infty} f_3^2(k) + 3\right]$$

where $f_i(k)$ is the impulse response between the first adder in section i and the output [11]. The q^2 factor (the square of the quantisation step size) was ignored in this work as only filters with the same quantisation step size were compared together.

2.3.4 Error Spectral Shaping

The noise which occurs in a second order section can be reduced by *Error Spectral Shaping* (ESS). This is a technique in which the quantisation error is fed back into the filter in such a way that it reduces or even eliminates roundoff errors over regions of the response.

There are many filter structures which can be used, one suitable canonic section is given in Figure 2.9 (after [11]). In this structure, $e_1(n)$, the difference between the pre- and post-quantisation value of $y'(n)$, is passed through another set of coefficients into the adders. It is possible to reduce the effects of the noise by careful selection of these additional coefficients. In the figure, $e_2(n)$ is the error caused by requantising the ESS inputs into the left-hand adder, $e_3(n)$ is the equivalent quantisation error for the ESS inputs into the right-hand adder, and $e_4(n)$ is the quantisation error on the output from the right-hand adder. ESS coefficients are usually chosen to be powers of two or integers to minimise the noise contribution of the ESS filter itself.

Many modern DSP chips contain a double-precision accumulator, which can hold high-accuracy numbers. This allows calculations to be performed within a second-order section without adding roundoff noise at each multiplication, because the solution only needs to be truncated to a lower accuracy when it is being written to memory. This means that the filter implementation is inherently less noisy and more accurate. The increasing wordlength of DSP chips means that some aspects of quantisation effects are having less impact on filter performance.

2.3.5 Coefficient pairing and ordering

For full precision arithmetic, it does not matter which order the second-order sections appear in, nor which of the numerators is paired with which denominator. Once the system uses fixed precision, this is no longer the case, since each section affects the signal in a different way and the noise from each section is passed through all subsequent ones. This means that the order in which the signal passes through the

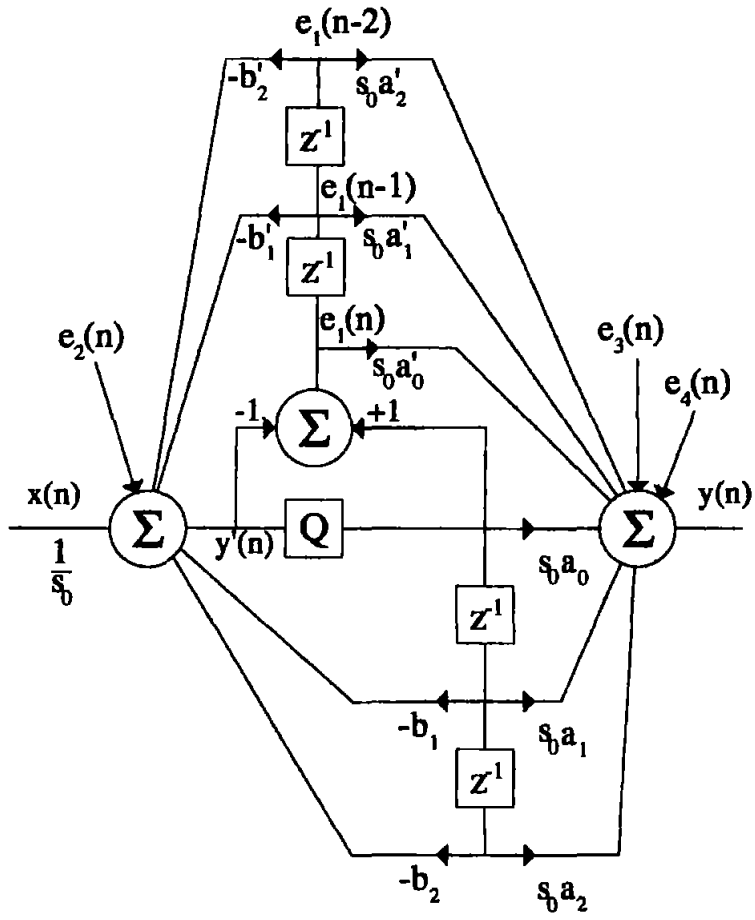


Figure 2.9: Canonic filter section with signal scaling and error spectral shaping.

sections will affect the overall noise on the output. It also implies that since the noise of a section depends on the coefficients within it, the pairing of the numerators and denominators also affects the overall noise of a system. The pairing and ordering is not a trivial problem, the total number of possible pole-zero pairs being given by:

$$\phi_N = \left[\left(\frac{N}{2}\right)!\right]^2$$

which for $N = 10$ gives 14,400 possible filters. While this number of filters is easily searched exhaustively by modern computers, changing the ordering and pairing of the poles and zeros changes both the necessary scaling, and the noise characteristics of the filter. Standard design techniques [37] perform the optimisation of the coefficients, and pole-zero pairing and ordering in two independent steps, so any filters produced can only be optimal in either sense. What is required is a multi-criterion optimisation method, which allows the designer to specify the desired weightings of the importance of the different design criteria such as frequency and phase response and roundoff noise gain, and which then optimises the filter with respect to these combined criteria.

2.4 The Rôle of Optimisation in Filter Design

Within each filter design step, optimisation can be used to improve the final design. These steps will now be examined for IIR filters made of cascaded second-order sections, to determine the potential uses of optimisation and how natural algorithms could be used to advantage.

2.4.1 Specification

The specification covers the desired characteristics of the filter, namely the phase and magnitude responses, together with other behaviours which may be limited by the desired implementation. For example, the DSP chip to be used may have limits on the I/O data rate, wordlength or highest available operating frequency.

There are several opportunities for optimisation here, covering the desired frequency

response, phase response, wordlength, delay etc., all of which will have a bearing on the best way to implement the filter. Due to constraints which may be forced by the chosen implementation, such as a fixed wordlength or sampling frequency, it may not be possible to optimise the specification independently of the implementation, and the two may need to be optimised together to find the best compromise.

The best means of optimisation here is probably fuzzy logic or an expert system, which could use a database of available DSP processors to determine the best one to use for a given specification, or, in reverse, could return the nearest possible specification for a given DSP chip. This step is not especially suited for optimisation by the GA and SA methods under investigation.

2.4.2 Coefficient Calculation

The calculation of the filter coefficients is perhaps the most important step in designing digital filters, as the coefficients play the greatest part in determining the characteristics of the filter. There are several methods of designing IIR filters (Section 2.3.2), which often involve the complication of converting an analogue filter to its digital equivalent.

The GA could be used to by-pass this step, by working directly on the coefficients or the pole-zero positions, in order to find the coefficients which best fit the desired magnitude and frequency responses. A fitness function could be constructed which drove the GA towards these responses, so that the designer need not know anything of the actual operations involved.

Ways in which this can be accomplished will be covered in later chapters.

2.4.3 Structure Realisation

There is a great deal of potential for structural optimisation with IIR filters [38, 39]. Not only is there a choice between the topology of second-order sections to use in the filter, there could even be a mixture of topologies. It is also possible for the sections to be positioned in a cascade or parallel structure, each with its own noise characteristics

and storage requirements.

The actual optimisations which could be performed here are dependent on which parts of the design cannot be changed. If there is a fixed wordlength, for example, or a limit on the amount of intermediate storage, then structural optimisation cannot be performed independently of other optimisations, such as noise, as the structure and wordlength affect the noise characteristics of the filter. However if some of these restrictions can be lifted then the realisation structure could be included in the GA's chromosome, and the various performance measures affected by the structure incorporated into the fitness function, so that the GA can include their effects while optimising the coefficients and structure. Some GA-based filter optimisations explicitly include the structure of each filter section, such as the approach of Roberts and Wade [40], which builds up a filter from a library of simple standard filter sections. Suckley [41] has shown a similar GA-based approach to perform better than other, standard techniques. Uesaka and Kawamata [42] have used a Genetic Programming method to design second-order filter structures with low coefficient sensitivity.

2.4.4 Analysis of Finite Wordlength Effects

As has been mentioned above, the effect of using a finite-precision implementation can have a deleterious effect upon the signal passing through it. The level of optimisation to be applied to the finite wordlength analysis is best determined on a per-problem basis.

The simplest way to include some form of analysis is simply to optimise the coefficients directly in a quantised form. This will result in a filter with a lower optimum performance than the one with full-precision coefficients, but the coefficients will automatically take into account the finite precision of the implementation. These coefficients will, in general, perform better than quantised full-precision coefficients. Schaffer and Eshelman [43] have shown the GA to be able to successfully optimise FIR filters with coefficients limited to powers-of-two integers, which allow multiplies to be

replaced with quicker and more efficient shifts.

This level of optimisation does not take into account the finite wordlength effects of the calculations within the system, which also affect the filter performance. To add this analysis to the optimisation requires a much more detailed model of the system, in which the truncated results of calculations are also included. The filter which would be found by such an optimisation would have the best performance of these filters as it would have taken into account the effects of using a fixed-precision system intrinsically, during the optimisation process, and no additional subsequent analysis would be necessary.

Since both the structure of each second-order section and its coefficients will affect the overall noise of an IIR filter, noise optimisation cannot be carried out alone: there must be some feedback so that the structure and coefficients can be altered in a way which helps to reduce the noise. However, unlike the other finite wordlength analyses above, reducing the noise will result in a worse magnitude response performance, and *vice versa*, because there are only so many degrees of freedom which can be exploited to improve the filter performance. This means that a multi-criterion optimisation technique, such as those described in Chapter 6 would have to be used. A filter designed in this way would be a compromise between good performance in each aspect of the design, but the ability to produce such a filter without resorting to a number of individual analytical steps would be to advantage.

2.4.5 Implementation

The best method of implementing the system, in either software or hardware, will be determined by the specification of the system. Different DSP systems use different wordlengths, data rates etc., so are suitable for use in different situations. Even software implementations will have restrictions according to the system they are written for, which could have limited memory, speed or bit-lengths.

Dempster and Mcleod have proposed an analytical method for exploiting calculation

redundancies to implement previously-designed finite-wordlength FIR filters using the minimum number of adders instead of full multiplications, thereby increasing speed and simplifying the implementation [44]. Redmill et al [45, 46] have used GAs to optimise filter coefficients with respect to both filter performance and number of adders simultaneously.

As suggested above, a search for the best implementation for a given system could be performed by an expert system or fuzzy logic, which would return a suitable implementation method after examining a database for those which match the specifications most closely.

It should be clear from the above analysis that there is a large amount of interdependence between all of the steps in designing a filter. This means that it impossible to perform an optimisation with respect to the aspects covered by one section without altering, and perhaps reducing, the performance in another aspect. To overcome this, the only way to produce an overall 'optimum' filter is to perform the optimisation with respect to all the possible design criteria simultaneously so that they can all be traded off against each other at the same time.

The application of the Genetic Algorithm to a range of these optimisation tasks will be covered in later chapters.

Chapter 3

An Introduction to Genetic Algorithms and Other Natural Algorithms

3.1 Background to the GA

The Genetic Algorithm (GA) is a search and optimisation technique [47, 48], which was inspired by theories of natural selection and evolution by the survival of the fittest. In nature, the survival of a species can be viewed as an optimisation task, where the problem is one of adapting to the surrounding environment. Those species which are well-adapted will survive to adulthood, and will then be able to pass their good genes on to their own offspring. These offspring will contain various combinations of genes from two successful parents, and should therefore describe some successful individuals in turn. At each generation, the survival of the successful offspring will concentrate 'good' genes in the population, while offspring which are sickly or poorly-adapted to the environment will either die before breeding or be unable to compete successfully for a mate, so will not pass their genes on.

A process of millions of gene crossovers and a very small number of mutations, combined with the elimination of poor individuals, causes a gradual evolution of the

population until it contains only individuals which are well-adapted to the environment and have the best chances of survival. The success of this process in nature is obvious, but it is perhaps less clear as to how this relates to a computer algorithm for engineering design. This chapter will describe the operation and use of a Genetic Algorithm, and discuss its advantages and disadvantages over conventional techniques.

3.2 Outline of a Standard Binary-Coded Algorithm

In nature, the population of seals, for example, consists of a number of individuals, each of whose cells contains DNA (deoxyribonucleic acid). This DNA makes up the genes that fully describe the seal. Each member of the population contains DNA which differs slightly, resulting in differing sizes, colourings, strengths, acuity of sight and smell etc. Elements of DNA from both parents are combined randomly in the offspring, resulting in them having attributes of both. Those offspring with good, new combinations of genes will be more likely to survive in turn and pass them on to subsequent generations, while the poorer ones die out and are lost. This is in effect a concentration of the information, contained in the DNA, about what makes a 'good' seal.

The engineering design process contain many similarities to that of the problem of evolution: there is the goal of a suitable solution or solutions; the characteristics of a solution can be tuned by altering parameters; and there is a means of determining the quality of the solution. In nature, good solutions are marked by an ability to survive and reproduce, ensuring the continuation of the species, while a good engineering solution fulfils the design specifications to within acceptable tolerances. The tunable parameters in the natural systems are the genes in each individual's DNA, while in an engineering context they are the parameters to the model which allowed the quality of the system to be determined. This suggests that by encoding the design parameters of an engineering problem in computer-storable simulated chromosomes and simulating the natural selection and reproduction processes, the solutions they contain could be

evolved into good or optimal solutions.

The basic operation of a standard GA can be summarised as follows:

1. Produce an initial *population* of *individuals*, each containing a random solution to the problem under investigation.
2. Determine the quality of each solution, known as their *fitness*.
3. *Select* members of the population according to their fitness.
4. Copy or *reproduce* the selected members of this *parent* population, with different frequencies, to form the next *child* population.
5. Perform *crossover* and *mutation* on the members of this child population.
6. Return to 2 until the maximum number of generations has been reached, or the best solution found so far is within acceptable tolerances.

These steps will now be examined in detail.

3.2.1 Encoding the problem

In order for the GA to perform successfully for a given design problem, the problem must be encoded correctly. The GA does not, in general, optimise a problem model directly, but rather by optimising parameters to it, which alter its performance. These parameters must be stored in a suitable form according to their use and range, as either binary, integer or floating point values. The use of a floating-point representation is discussed in Section 3.11, but in a standard GA, the parameters are encoded into a binary bit-string with a fixed format, i.e. the parameters always appear in the same order, and each bit always has the same meaning, as illustrated in Figure 3.1. This string, or chromosome, is decoded later on by the *fitness function*, which determines how good a solution the chromosome represents, a measurement called its *fitness*. The calculation and importance of the fitness is discussed more fully in Section 3.8.

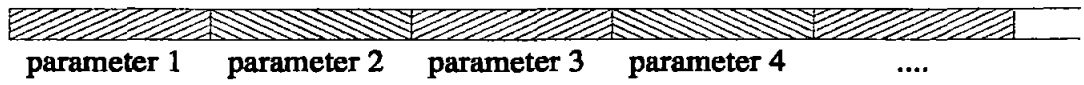


Figure 3.1: Illustration of the constant encoding of the binary chromosome of a standard GA.

3.3 Initialisation

The initialisation of the population determines which areas of the solution space the GA samples at the start of a run. If nothing is known about the locations of the good regions of the space, then it is important to initialise the population to cover as much of the space as possible, as evenly as possible, so the chromosomes are filled with random bits. While it may be possible to initialise the population by spacing it evenly within the space, this could mean that each model parameter would only have a few values and bit patterns, and it would then take the GA some time to generate intermediate values. Other methods, such as the Sobol sequence [33] could be used to initialise the population using a quasi-random sequence. This would guarantee a maximally even spread of points, regardless of the number generated, but in some applications may only cover a limited number of values for each parameter. Initialising the population at random helps to ensure that a wide range of parameter values are sampled, which in turn subsequently helps the GA to search the space effectively.

If the problem has known constraints on parameter values, or the location of good areas is known, then this information can be used to good effect during the initialisation procedure by either biasing or limiting the values to their known good regions or allowed ranges [49, 50]. This clearly speeds up the convergence of the GA to the good areas, but is only of use if the problem is known or can be analysed.

The number of members stored in the population must be enough to maintain diversity in the gene pool, while being small enough to ensure that storage and CPU time limits are not exceeded. Values of 30–100 are most common for standard Genetic Algorithms.

3.4 Selection

It is important that the selection method should keep the best solutions from each generation, but it is perhaps less obvious that it should also retain a selection of the poorer members in order to maintain gene diversity in the population. Without this, the population will rapidly converge on whichever members of the initial populations have the highest fitnesses, resulting in a poor final solution or very slow optimisation, unless the initialisation has been fortunate enough to pick a point close to the optimum. This is known as *premature convergence*, and the GA designer much take precautions to prevent it while still allowing the GA to converge on the optimum solutions.

There are a variety of selection methods which can be used, the simplest being *weighted roulette wheel* selection [48], which gives each member of the population a probability of being selected which is proportional to its fitness. The roulette wheel selection method has the problem that it is quite possible for good solutions within the population to be missed during the selection process and lost while the poorest solutions are retained.

While roulette wheel selection is very quick and simple to implement, a more useful method, and one which is widely used, is *Stochastic Remainder Selection* without replacement (SRS), which has been shown [48] to be one of the best general selection techniques. SRS ensures that the members with above-average fitness are always reproduced, with a number of copies proportional to the degree of excess fitness above the average, while all the other members of the population also have a chance of being selected to fill the remaining gaps in the population, which helps to maintain diversity.

SRS involves selecting the individuals to be reproduced in the following way:

1. Linearly scale all fitnesses to an average of one.

2. For each member whose scaled fitness is ≥ 1, allocate a number of copies equal to the integer part of the scaled fitness, and subtract that number from the scaled fitness, leaving all the scaled fitnesses fractional.

If the new population now contains fewer members than the old, further selections must be made to fill the remainder of the population:

3. Pick a random member and a random number in the range 0–1. If this number is less than the remaining fractional scaled fitness of that member, then allocate one copy to the new population, and set the scaled fitness to zero to prevent that member being selected again.

4. Repeat 3 until the new population is the same size as the old.

Other selection strategies have been successful in other applications, but SRS was selected for use in the single criterion GAs used in this work.

3.5 Reproduction

The selected members of the population are now copied or *reproduced*, to form a new population. The selection method, as described above, will determine the number of copies to make of each member of the population. The next population will usually have the same number of members as the previous one.

3.6 Crossover

The result of selection is a group of individuals who have been selected either for having a high fitness, or to fill any remaining spaces in the population and maintain diversity. Selection means the average fitness of the new population is higher than the old, but the maximum is unchanged as no new members have been created. Repeated selections alone would therefore result in a population just containing copies of the best chromosome from the initial population. In nature, new chromosomes are created by sexual reproduction in which two parent chromosomes combine to give offspring containing a mixture of characteristics from both.

In the binary GA, this is simulated most simply by randomly picking a pair of 'parents' and a crossing point along the chromosome, and swapping the bits after that point between the two chromosomes, giving two offspring strings with some genes from each parent. This has the result of combining the genes from two solutions which have survived long enough to reproduce, to form two new, potentially even better solutions. The next round of selections will determine which of these offspring are fit enough to survive and pass their genes on in turn. The repeated action of crossover and selection is the main driving force in the Genetic Algorithm, and results in the proliferation of those genes which cause the chromosomes containing them to have a high fitness. By repeatedly selecting and crossing high-fitness strings, the genes are gradually brought together to form even better solutions. This is discussed more fully in Section 3.10.

Crossover is implemented by picking random pairs of (preferably different) individuals, and, with a fairly large probability (typically 0.6), choosing to swap a selection of bits between the pairs. There are several standard schemes for swapping the bits:

1-point crossover: A random point is chosen along the chromosome, and all the bits after that point are swapped between the two parents to form two offspring. For example:

$$\begin{matrix} \{00000 \mid 000\} \\ \{11111 \mid 111\} \end{matrix} \rightarrow \begin{matrix} \{00000111\} \\ \{11111000\} \end{matrix} \tag{3.1}$$

This method has the disadvantage that points towards the ends of the strings are crossed less frequently than those in the middle.

2-point crossover: Two random points are selected and the bits between them are swapped. This is essentially the same as mapping the chromosome to a ring and swapping the bits between two randomly-selected points, which helps to remove the end effects and so causes the string to be optimised more evenly along its length:

$$\begin{matrix} \{00 \mid 000 \mid 000\} \\ \{11 \mid 111 \mid 111\} \end{matrix} \rightarrow \begin{matrix} \{00111000\} \\ \{11000111\} \end{matrix} \tag{3.2}$$

Multi-point crossover: Many crossing points are picked and the bits between them alternately swapped and kept. This is still more even than 2-point, but it is arguable if the gains are worth the extra computational time, and the disruption to good chromosomes which have been built up is higher:

$$\begin{matrix}\{00 \mid 000 \mid 00 \mid 0\} \\ \{11 \mid 111 \mid 11 \mid 1\}\end{matrix} \rightarrow \begin{matrix}\{00111001\} \\ \{11000110\}\end{matrix} \tag{3.3}$$

Uniform crossover: For maximum evenness, this technique chooses randomly whether or not to swap *each bit* along the length of the string. This is very disruptive, so compensation, possibly in the form of *elitism* (see Section 3.12) can be used to prevent good solutions from being lost:

$$\begin{matrix}\{0 \mid 0 \mid 00 \mid 0 \mid 0 \mid 0 \mid 0\} \\ \{1 \mid 1 \mid 11 \mid 1 \mid 1 \mid 1 \mid 1\}\end{matrix} \rightarrow \begin{matrix}\{01001010\} \\ \{10110101\}\end{matrix} \tag{3.4}$$

3.7 Mutation

The crossover/selection mechanism tends to concentrate the population in the high fitness regions of the search space, which is the desired search action. However, it is quite possible, especially with highly multi-modal surfaces or those with similar height peaks, for the GA to converge to a sub-optimal peak. To help it to escape and continue searching, *mutation* is used. Mutation causes a very small number of bits, typically 0.01–1% of the total number of bits in the population to be flipped at random. For example:

$$\{11\underline{1}11111\} \rightarrow \{11\underline{0}11111\} \tag{3.5}$$

This changes the values of genes within the chromosome, and thereby pushes a few members of the population into new, possibly remote areas of the parameter space, allowing the GA to continue its search for higher fitness regions. While this occasionally finds a better solution and so helps the GA, it is disruptive and is as, if not more likely to move the chromosome to a poorer solution, so is only applied with a very low probability.

3.7.1 Mutation and parameter encoding

While the standard binary encoding is simple to understand and decode, it has undesirable characteristics from the viewpoint of the GA which can make it unsuitable for some applications such as numerical optimisation.

When a decimal number is represented using a binary encoding, the number of bits which change between consecutive numbers is known as the *Hamming distance*, and can vary widely from one bit to the full bit length of the number. For example, to change from 0 to 1 only requires one bit to change, but 127 to 128 changes eight different bits (01111111 to 10000000 for an eight-bit representation). This situation, where adjacent gene values have a large Hamming distance between them, is known as a *Hamming cliff*. If such a Hamming cliff lies between the current and optimum values, then mutation will be unlikely to overcome it as it only changes a few bits per generation.

There is an alternative binary encoding called *Gray coding* which overcomes this difficulty by ensuring that consecutive numbers only differ by one bit. The difference between the two encodings for the first 16 numbers is shown in Table 3.1. A binary number can be easily converted to Gray coding by Exclusive-ORing it with itself shifted right one bit:

$$gray = bin \wedge (bin \gg 1)$$

To convert it back from Gray coding to binary, each bit is added to the one before it, ignoring the carry:

$$(bin_0,\ bin_1,\ bin_2, \ldots) = (gray_0,\ gray_0 + gray_1,\ gray_0 + gray_1 + gray_2, \ldots)$$

The major advantage of using Gray coding is that it is possible to move between adjacent decimal values by flipping a single bit. This means mutation is always able to 'creep' from one value to the next by flipping a specific bit, although it is still possible to move larger distances by flipping other bits.

Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Table 3.1: A comparison between the binary and Gray-coded forms of the first 16 numbers.

3.7.2 Other forms of mutation

It can be helpful for some applications to weight the probability of mutation along the length of the string, perhaps to reduce the disruptiveness of sensitive regions of the chromosome, or where the chromosome requires different forms of mutation at different locations to maintain the validity of solutions. Altering the degree of disruptiveness (Section 3.11.2), or dynamically changing the mutation probability as the run progresses can also be beneficial. This could either be a simple reduction with time, or more complex, such an increase if the GA fails to improve for a number of generations, then a decrease once it begins to improve.

While it is difficult to predict what crossover and mutation rates are best for a given problem, and indeed most optimum rates are determined through trial and error, if the search space has relatively few peaks, then the weighting should generally be on the crossover to ensure convergence, while for a hilly space, a higher mutation rate can be useful to make the GA sample the space more widely and to allow it to escape from local suboptimal peaks.

3.8 The Fitness Function

The fitness function takes the chromosome, extracts model parameters from it, passes them to the model, and returns a value which is a performance or suitability measure of that chromosome. The importance of choosing the correct fitness function cannot be overstated. It is the only viewpoint the GA has of the space it is searching, and the only performance measure of the solutions it has found. Not only does it therefore determine what the GA is actually looking for, it also influences the speed of convergence and how the selection mechanism performs. If the chosen fitness function makes the search space appear very hilly, the GA is more likely to suffer from premature convergence, whereas if it appears too flat, then the GA will not be driven towards the optimum strongly enough, and will take a long time to converge.

3.8.1 Fitness Scaling

The values returned by the fitness function are usually a simple measure of the model's performance, but may not be particularly suitable for direct use in the selection process. For example, if fitnesses have a small range or a large offset, it reduces the relative difference between individuals in the SRS algorithm and therefore also reduces selection pressure. Alternatively, having too great a range can cause premature convergence around any good members in the initial populations, so artificially scaling the fitnesses to a suitable range can help the GA. The offset can be removed by translating the fitnesses to a fixed lower bound, while the range can be adjusted by a power scaling to increase or reduce it. If a ranking selection mechanism (e.g. tournament selection [51, 52]) is used, then the fitnesses' absolute values are irrelevant since only their relative values are used to determine the fittest, so scaling is unnecessary.

3.9 Advantages and Disadvantages of the GA

A major disadvantage of conventional hill-climbing strategies is that they can become trapped on sub-optimal peaks. Unless the search is started in the immediate neighbourhood of the optimum peak, it will not be able to find it. The GA is able to escape from these sub-optimal peaks by the process of mutation, and by crossover between dissimilar chromosomes.

Since the GA only uses point samples of the search space, it does not require any gradient information. This means that it is possible to optimise complex models for which analytical solutions are extremely difficult to obtain. The GA has a particular strength in that it can be used for multi-criterion optimisation, automatically trading off performances with respect to different criteria, a process which is described further in Chapter 6.

Although the GA takes many fewer samples than a random or exhaustive search, the fact that it still requires a great many function evaluations means that it will often be much slower than standard analytical techniques where these exist.

The other main problem with the GA is that while it is *statistically* guaranteed to find the optimum, in practice for more complex problems it will generally only return a solution which approaches the optimum, except for simpler problems. If a near-optimum solution is not good enough, then once the GA has converged, a hybrid technique (such as a hill-climber) can be used to perform the final optimisation and find the true optimum. This is discussed in Section 3.13. Alternatively, the GA could be restarted using the previous best solution as a population seed. Ishibuchi and Murata [53] went to the extreme of using a local search on every solution found by GA.

3.10 Convergence Theory

While it should now be clear how the GA works, it is perhaps not so obvious as to *why* it works. There have been a number of attempts to describe and predict its behaviour, the original being that of Holland's *Building Block Hypothesis* [47], although more recent analyses [54, 55] suggest other mechanism can give more accurate predictions of the GA's performance under some circumstances. His 'Schema Theory' predicted a string's fitness by the bit templates or *schemata* (singular *schema*) it contains. The templates are described by the ternary alphabet of {1,0,#}, where the '#' is a 'don't care' symbol. For example, the schema {1##} would match any three-bit string with the first bit set. A schema therefore describes a set of chromosomes with similarities at a subset of the total number of bit positions.

Schema theory predicts that if a chromosome contains certain schemata it will have a higher than average fitness, and a lower one if it contains others. During selection, therefore, the higher-fitness schemata will tend to be propagated, while the poorer ones will be lost, and by performing crossover these higher-fitness schemata can be brought together to form even fitter individuals. Since crossover is very disruptive, it is found that schemata whose fixed bits have a wide span are not propagated as well as shorter ones. Under this scheme there are therefore some restrictions: the high-fitness schemata should be short in order to reduce the risk of them being damaged during crossover, and it must also be possible to predict to some degree the fitness of a chromosome from just a small section of it. If this is not possible, then no short templates can exist and the only good solutions will be either complete or nearly complete ones. Under these conditions the GA does not perform well, so it is good practice to design a GA around a chromosome in which related genes are adjacent, and there is no relationship or interdependence between widely-separated genes.

Under schema theory, the GA implicitly processes many schemata in parallel as each member of the population contains a large number of them. Since the GA is trying to bring the high-fitness schemata together to form an optimum chromosome,

the population size should ideally be chosen so that after the random initialisation it contains, on average, one copy of each schema, of which there exist a total of 3^B for a chromosome of length B. Although there are more useful schemata in a larger population, the minimum size increases slower than the chromosome length so a large number of bits does not necessarily mean a punitive amount of storage or speed decrease due to a huge population size. In general, the population size used is smaller than this minimum size, but the GA is still able to perform satisfactorily.

As the cardinality of the alphabet used rises, so does the minimum population size that is needed to contain, on average, at least one of each schema. For a floating point GA, there are essentially an infinite number of possible schemata, so the GA should require a vast population to even begin to work properly, as a normal population with a binary gene will only contain a few of the possible schemata. Goldberg [56] speculates that the GA builds up its own lower-cardinality 'virtual alphabet' from the features in the search space, but the reality of this scheme is unproven.

Recently, doubt has been cast on the validity of the Building Block Hypothesis, and a number of alternative approaches, e.g. using Markov chains [54, 55] , have been proposed, although all have their faults, and the true nature of the GA's optimisation processes remains unresolved.

3.11 Floating-point Chromosome GA

The emphasis so far has been on the original, and still most common, binary form of the GA, but it is by no means the only chromosome encoding which has been used. Other encodings, using integers [40] or real numbers [57, 58], have also proved suitable for different applications. Some problems, such as numerical function optimisations or scheduling are less suited to a binary representation, and can be more conveniently encoded in higher-cardinality alphabets. This can allow each gene to represent a single optimisation parameter directly, making it easier to write the fitness function, and removing the need for decoding routines which add complexity and time to the fitness

calculations. Although Holland's schema theory predicts that a binary alphabet is the best encoding for the GA, the successful use of other encodings has shown that this prediction is uncertain and other processes may be occurring which allow the GA to work successfully. The optimal encoding is therefore problem-specific.

In the optimisation of many real-world scientific and engineering problems, a real-coded chromosome is attractive due to the high accuracy of measurement which is generally necessary, since it requires a very long binary string to obtain a high numerical accuracy. By encoding such problems with floating-point numbers, they can be used directly in the problem model which for complex problems requiring many function evaluations can be a substantial time saving as no decoding is required.

Real-coded genes have further advantages in that they also avoid some binary-coding specific problems such as Hamming cliffs. Although Gray coding overcomes this problem, it still requires many bits to achieve a high accuracy. For a floating-point GA, mutation can perturb the gene about its existing value to different degrees and so hill-climb towards the optimum. Another advantage is that the GA will converge faster with a high-cardinality alphabet, which is helpful if the optimum is easily found, either by the GA or an add-on hill-climber. It is still possible for the GA to be *blocked* (see also *deception* in Section 6.4), which happens when the initial sampling of the search space causes the GA to move towards regions which are blocked from hill-climbing towards the optimum peak by either valleys or other, suboptimal peaks.

In order to use a real-coded chromosome it is necessary to adapt the GA to some degree, as the binary crossover and mutation techniques mentioned above are not suitable in their current form. The changes will now be examined in detail.

3.11.1 Floating-point Crossover

The floating-point GA adopted for this work was derived from that of Janikow and Michalewicz [59], where the chromosome is regarded as a vector. This GA was specially designed for numerical optimisation, and uses a vector-like chromosome of a string

of real numbers with a number of crossover methods. These utilise *convex vector combination* of the chromosomes to generate new gene values and chromosomes. A dynamic form of mutation is also used, which reduces the degree of disruption as the run progresses to reduce the loss of good solutions once the population has converged.

Convex combination works as follows. First two parent chromosomes (vectors) are selected, for example, **a** and **b**, where

$$\begin{aligned} \mathbf{a} &= \{a_1, a_2, a_3, \ldots, a_n\} \\ \mathbf{b} &= \{b_1, b_2, b_3, \ldots, b_n\} \end{aligned} \tag{3.6}$$

If a crossover point is selected at the third gene position, the offspring are given by:

$$\begin{aligned} \mathbf{c} &= \{a_1, a_2, \rho.a_3 + (1-\rho).b_3, \ldots, a_n\} \\ \mathbf{d} &= \{b_1, b_2, (1-\rho).a_3 + \rho.b_3, \ldots, b_n\} \end{aligned} \tag{3.7}$$

where ρ is a uniform random number in the range 0–1. This results in two offspring, mostly containing the characteristics of one parent, but with a degree of the other at one gene position. This can be extended to cover more than one gene position to give offspring with a greater mix of the properties of both parents. In all, five types of crossover are used within the GA, and one picked at random for each crossover to be performed:

Normal crossover: Strings are crossed the same way as in single-point binary, with crossing points between genes, i.e. between a_k and a_{k+1}.

Multiple crossover: This is similar to multi-point binary crossover, with crossing points selected as in the Normal crossover above.

Arithmetic crossover: This is the simple one-gene form of convex combination shown in Equations 3.6 and 3.7.

Multiple Arithmetic crossover: Here several genes, selected at random, are crossed as in Arithmetic crossover.

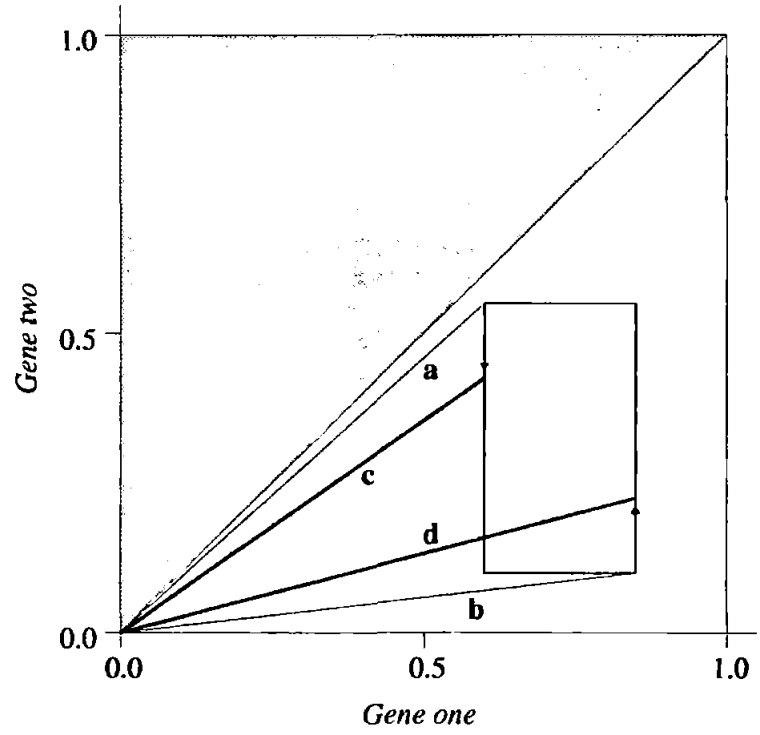


Figure 3.2: Example Arithmetic crossover for a two-gene chromosome.

Whole Arithmetic crossover: This is a special case of Multiple Arithmetic crossover, where the entire string undergoes Arithmetic crossover.

It should be noted that neither Normal nor Multiple crossovers actually change gene values, as they simply swap them unchanged between individuals. The three types of Arithmetic crossover do introduce new gene values into the population.

An example of the effect of Arithmetic crossover for a two-gene string is given in Figure 3.2, where gene two has been crossed. Figure 3.3 illustrates Whole Arithmetic crossover for the same type of chromosome. It is clear from the positions of the child chromosomes created in these figures (labelled **c** and **d**), that the action of both types of crossover is to rotate the vectors around their average point, followed by a scaling centred on this point. This closely mimics the effect that normal crossover has on binary strings [60]. For three dimensional chromosomes (i.e. with three genes), the effect is of a rotation and scaling around the central point of a cuboid.

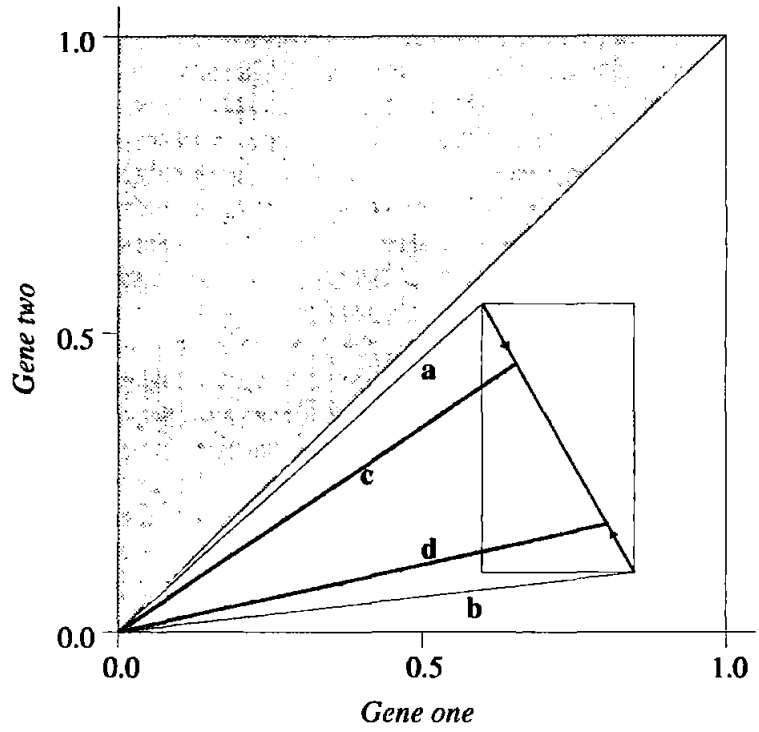


Figure 3.3: Example Whole Arithmetic crossover for a two-gene chromosome.

3.11.2 Floating-point Mutation

The type of bit-flipping mutation used in the binary GA clearly has no analogous equivalent for real genes. While it would be a simple matter to replace each gene with a random number, this is very disruptive, and could easily destroy any good solutions which have been found. A better alternative is to simply move the gene around its current value, within any bounds which have been imposed, but this again has its problems. The size of the movement must be big enough to allow the GA to search the space effectively at the start of the run, without being so large that solutions are lost after convergence. One solution to this, which was used here, is to start the run with a reasonable size of perturbation so that the space is covered well, and then reduce it progressively throughout the run so that solutions are not lost as the population converges on the better regions of the space. It is possible to tune the rate of decrease of the perturbation according to the problem in hand.

The method used here was to first produce a value *dmpow* which reduced exponen-

tially from 1 to 0 as the run progressed:

$$dmpow = \left(1 - \frac{generation}{maxGenerations}\right)^5$$

It is possible to alter the rate at which the perturbation decreases by increasing or decreasing the power term in this expression. This value was then used to generate a randomly-distributed perturbation amount:

$$dmv = \frac{1 - (random(0,1))^{dmpow}}{2}$$

and this in turn was used to either increase or decrease the gene value within its limits of 0–1:

$$newValue = \begin{cases} oldValue + (1 - oldValue) * dmv & \text{random}(0,1) < 0.5 \\ oldValue - oldValue * dmv & \text{otherwise} \end{cases} \quad (3.8)$$

This had the effect of moving a gene's value a larger amount at the start of a run, but rapidly reducing the range of the perturbation as the run progressed.

3.12 Other GA techniques

For some problems, such as flowshop scheduling and routing which used list-based chromosomes, the GA's performance can be improved by including another form of gene manipulation known as *inversion*. Inversion takes a section of the chromosome and reverses it, replacing it at the same position in the string. This can be very disruptive to the building blocks in the chromosome, so is only applied with a low probability, but can be instrumental in finding optimal solutions. It is however of little use in numerical optimisation problems, where the meaning of a particular bit in the chromosome is fixed.

The random nature of the GA has the disadvantage that it is possible for a good solution to be lost if all copies of it are picked for crossover with inferior solutions. This is especially true for new solutions whose fitness only improves slightly on the previous best, where there may only be one or two copies in the population. In order to prevent

these solutions from being lost, a technique known as *elitism* can be used. Instead of simply copying the newly-created population over the old one, they are both examined, and one or more of the best solutions from the old population are copied over the worst in the newly generated one, ensuring that the population always contains at least one unaltered copy of the best solution found.

It has been shown that the GA implicitly processes many schemata in parallel, and this can be extended by having several GAs running in parallel on the same problem, either simulated on a single computer, or distributed over a network. At intervals, copies of good solutions found by each GA are sent to the other GAs, which add them to their own populations and continue processing. This has many advantages, more solutions can be tried, a greater gene diversity can be maintained, each population can converge to a different solution and still be aware of others by the input of new solutions, and the effective population size can be much larger than is feasible for a single GA.

3.13 Hybridisation

Although the GA is able to search large spaces very efficiently and escape from local sub-optimal regions, its random nature means that it is not guaranteed to find an optimum solution, even if it has converged to the high-fitness region containing it.

The GA can be helped to complete the optimisation process by the addition of a *hybrid* search technique, such as a hill-climber. This can be used in a number of ways, for example as a final process to be called on once the GA has completed its run. This approach assumes that the GA has converged on the locality of the optimum during the run. Another approach, which was adopted for part of this work, is to call on the local search if the GA has not made an improvement on the best fitness it has found for a number of generations in succession. This method is not reliant on the GA having found the region of the optimum, and should merely provide a 'kick-start' to get the GA working again by injecting a selection of good new genes into the population.

This hybrid approach to the GA has been successfully applied to many problems, from optimising layouts to modelling metabolic systems [61, 62]. Hill-climbers that can be used include the Simplex method for floating point genes, and bit-flipping for binary. Krukowski and Kale [63] have shown bit-flipping optimisations to work successfully for IIR filters.

The first bit-flipping hill-climber developed for use in the binary GA simply scanned across the string, flipping each bit, and retaining a change if it increased the fitness. This was repeated until no bit-flip caused an increase in fitness. It was realised that this method always looked at each coefficient in turn, but as will be shown later, changing one coefficient moves the position of the optimum for the others. This means that the hill-climber finds the optimum value of the first gene with relation to the current values of the other genes, and not the true optimum, and so on down the whole length of the chromosome. Although applying the algorithm repeatedly might help, there is no guarantee of finding any truly good solutions.

3.14 The Search Space

The search space is an important concept in GA theory. It relates the problem parameters encoded in the chromosome with the complexity of the fitness function. By using the fitness of a chromosome as a height, a contour map of the fitness function's output can be drawn against the values of the genes within the corresponding chromosome. The hilliness of this map, together with other analyses, such as whether it contains discontinuities or other features, can be used to estimate the difficulty different techniques will have in solving the problem. Discontinuities and multiple peaks can cause problems for hill-climbers and calculus methods, for example, while hill-climbers which only optimise one parameter at a time will find it hard to move along angled features in the fitness landscape.

The GA is generally able to work in hilly, discontinuous landscapes, but there are other more subtle features which can cause it problems, such as a landscape where

it is only led towards the optimum by very small-scale structures, while the larger-scale features tend to lead away from the optimum. This particular effect is known as *deceptiveness*, and is covered in more detail in Chapter 7.

3.15 Simulated Annealing

Hill-climbing is a well-known technique for finding optima in a search space, but which solution it finds in a multi-modal space depends on where it is initialised. If its initial point is not close to the optimum, depending on the nature of the space, it may never find it at all. Using an analogy from thermodynamics, it is similar to quenching a hot metal—the crystal structure formed under these conditions is the first low-energy state that was found. Other, potentially lower-energy structures may exist, but it is necessary to cool the metal slowly (or *anneal* it) to allow it time to search for the lowest-energy state. There may even be substantial changes in structure as it cools, reflecting a wide-ranging search for a stable, low-energy structure.

The technique of Simulated Annealing (SA) optimisation [33] attempts to model this slowly-cooling approach instead of the rapidly-quenched style of the hill-climber. SA usually works in a similar fashion to a hill-climber by looking around its current position and always accepting moves to a better position. However, it differs in that it also allows a proportion of moves to a *worse* position, which will allow it to escape from its current optimum and search within another one. When metals are cooled, the reduction in heat energy lowers its ability to make large structure changes, so it becomes increasingly confined to its current position. This is reflected in the SA technique by initially allowing many moves to worse positions, but progressively reducing this probability as the number of iterations increases. This allows the SA to perform a wide-ranging examination of the search space at first, but to increasingly concentrate the search on any good areas it finds. The speed with which the range of the search contracts is highly problem-dependent, and trial-and-error must be used to find a sufficiently slow rate to allow an efficient search, while being fast enough for the result

to be generated within a useful time.

Depending on the implementation, SA may be statistically guaranteed to find the optimum, but as this may not be probable within a suitable timescale, multiple runs may be necessary. The SA could be restarted with the best point of the previous run, or this point could perhaps be used to discourage the search from revisiting the same optimum, thereby increasing the efficiency of the search.

The implementation of SA optimisation used in this work [33] was based on the Simplex Method described in Sections 4.4 and A.2, and utilises a Simplex Method hill-climber which has been modified to allow occasional moves to poorer solutions.

3.15.1 Applications to Filter Design

Simulated Annealing has been successfully applied to several aspects of filter design. Pitas [64] presented a SA-based optimisation for the length of a median filter, which, although it only had a small search space, indicates that SA is suitable for some filter design tasks. Smith and Henderson [65] have applied SA to ordering the sections of a cascade-realisation FIR filter, to optimise the roundoff noise. SA was found to be able to find orderings close to the minimum possible, and to be usable in environments where the search space was too large for an exhaustive search to be feasible. Chen et al [32] have used an SA-based technique to optimise adaptive IIR filters for system identification, where hill-climbing techniques can fail due to the occurrence of local maxima in the search space. PCAS filters with finite wordlengths have been designed by Lawson and Wicks [29] using a binary SA.

3.16 Differential Evolution

Storn [66] made use of a crossover method similar to the Arithmetic Crossover above, in his optimisation of IIR filters by Differential Evolution. This is a similar technique to the GA, in which instead of selecting two members of the population to cross, the coefficients of each member in turn are mixed with fixed proportions of those of the

best member and two other members of the population selected at random, to produce a single offspring. The offspring is kept if it is a better solution than its parent.

The technique was successfully applied to the optimisation of a tenth-order IIR filter cascaded with a fixed-coefficient FIR filter, where both the magnitude and group delay responses were optimised by penalising the fitness function if a solution went outside desired response templates.

3.17 Evolutionary Strategy

Evolutionary Strategy (ES) [1, 48] is very similar to a floating point GA, but generally has no crossover, relying instead on just mutation and selection to perform the optimisation. Franzen et al [67] compared the results of ES with Simulated Annealing for designing FIR filters with quantised coefficients, and found both to have similar performance, but neither performed much better than quantising the coefficients found by standard techniques.

3.18 Genetic Programming

Genetic Programming (GP), made popular by Koza [68], is a means of optimising mathematical functions to perform a particular task. Utilising tree representations of the available operations (such as simple mathematical operators like $*$, $/$, $+$ and $-$ or boolean operations like AND, NOT, OR) and the input parameters, crossover can be represented by swapping 'branches' of the tree, and mutation by replacing an existing operator with a different one. Computer programming languages such as LISP are often used as their syntax is already in a suitable tree-like form.

Applications of GP to filter design have a rather different approach to those for GA and SA in that it is not simply used for coefficient optimisation, but for functional and structural optimisation. Keane et al [19] have used GP to optimise an impulse response function which models an unknown system, and Rodríguez-Vázquez et al [69] have

performed multi-objective system identification. while Uesaka and Kawamata [42] have designed second-order filter structures with low coefficient sensitivity using automatic GP techniques.

3.19 Tabu Search

Tabu Search is a sequential search technique introduced by Glover [70] in which recently-visited solutions are made 'taboo' and cannot be revisited, thereby forcing the search into new areas. Current solutions are transformed into a range of potential new solutions, known as the *neighbourhood.* The transformation method must be chosen so that the path between any two solutions can be followed by repeatedly applying the transformation. The set of solutions within the neighbourhood is compared with the *tabu list* of previously visited points, and forbidden solutions removed. The next starting location is then picked from the remaining solutions, and its own neighbourhood examined. A variety of techniques have been proposed for picking the next solution from the neighbourhood: an unvisited solution is always preferable, or if none is available, a previously-visited solution can be selected according to how recently or how often it has been visited [71].

Tabu Search has been used successfully by Traferro and Uncini for designing adaptive filters with powers-of-two coefficients, which has a restricted search space [72]. For filters with integer coefficients, and especially with full-precision coefficients, the number of points in the tabu list will be extremely large as the number of solutions within the search space is much higher—an N-bit integer can represent 2^N values compared to the N values of a powers-of-two representation of the same range. For a single 16-bit integer range this gives a search space $2^{16}/16 = 4096$ times larger. A 32-bit floating-point representation of a single number contains around 10^8 more points than the powers-of-two representation, so the tabu list will also need to be orders of magnitude larger. This may make the technique unsuitable for high-precision or high-order filter optimisations due to storage and/or time constraints, as the time taken to compare the

solutions in the neighbourhood with those in the tabu list will rise linearly with the size of the tabu list. Battiti and Tecchiolli [73] have proposed a potential solution to this problem, by just using TS on a combinatorial representation to find promising regions, with a local search performing final optimisation. The TS parameters are adjusted dynamically in order for the search to be effective on a variety of problems.

Tabu search can also be used for multi-criterion optimisation, using an adaptation proposed by Hansen [74], which enables it to return a range of solutions with different performance tradeoffs, by weighting each Tabu Search within a population of solutions so that it tends to keep away from its neighbours, while moving towards better solutions.

Chapter 4

Optimising Frequency Sampling Filter Coefficients by Hybrid GA

4.1 Introduction

In this chapter, the first application of GAs to a filter design problem will be described. A suitable optimisation problem will be selected, and the method of applying the GA presented. Results are presented which lead to the publication of a paper covering this work in the IEEE Transactions on Signal Processing [4].

4.1.1 Selection of FS filters for GA optimisation

In order to begin the investigations into the use of the GA, it was necessary to select a filter type and optimisation problem to use as an initial vehicle. FIR filters were chosen for their simplicity, using the Frequency Sampling design as a first step since they only require the optimisation of a few numbers (the transition samples) to produce a solution. It would also be possible to write a suitable fitness function to search for optimal filters by optimising the extremal frequencies. Window method filters are less suitable as they require the optimisation of mathematical functions, which is not easily achieved by the methods under investigation. As floating-point GAs have been proved successful in other areas, it appears to be a simple matter to list the transition samples

in a chromosome for the GA to optimise, and to produce a fitness function which returns how close the frequency response is to a given desired response.

4.2 Use of the GA for FS filter Design

At this stage, both the problem, namely the optimisation of FS FIR filters, and a solution, the GA, have been proposed. The combination of the two to form a useful optimisation technique will now be examined. The following discussion covers the optimisation of lowpass filters, but is applicable to all the standard frequency-selective filter types.

When designing FS method filters, the values which need to be optimised are the magnitudes of the transition samples. For a magnitude response with normalised units rather than decibels, these samples have a range of 0–1 inclusive. These therefore need to be encoded into a suitable chromosome for the GA to work with.

The first filter type selected was non-recursive filters using the DFT interpolation method, and since the values to be optimised were floating-point and were used in a full-precision calculation, a real-valued chromosome and GA were chosen. The chromosome consisted simply of the transition sample values in order from pass- to stopband. The GA used was derived from that of Janikow and Michalewicz [59] as described in Chapter 3, but with some alterations and improvements which will be described later.

For a filter response to be acceptable, its magnitude response in the transition band should decrease monotonically from pass- to stopband. To help the GA maintain this, before each fitness calculation the transition samples are ordered so that they decrease monotonically. This is necessary at each fitness calculation because of the disruptive action of crossover and mutation, which can easily alter gene values and hence alter the ordering of the transition sample values. The fitness function passes the re-ordered chromosome back to the GA so that all of the chromosomes have the same transition sample at the same gene position. This also means that the total number of solutions which the GA has to search is reduced. For a filter with two transition samples, the

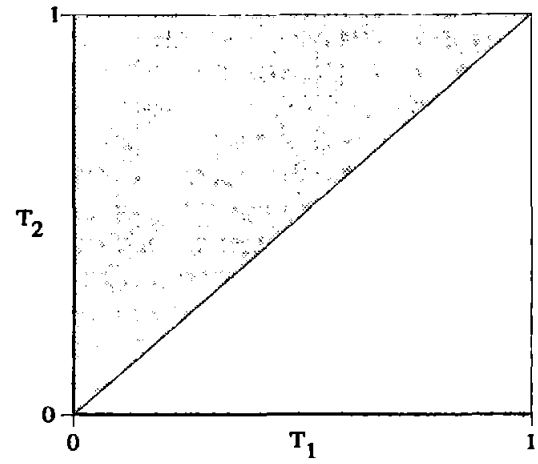


Figure 4.1: The shaded area is the disallowed region of the search space for a two transition sample filter.

limitation that the second sample must be less than the first means that only half the space is valid, as shown in Figure 4.1. In general, the proportion of the total space which gives allowable filters is given by:

$$p = \frac{1}{N_t!} \tag{4.1}$$

where N_t is the number of transition samples. For $N_t = 4$, $p = 1/24$ or only 4.2% of the total space, dropping to just 0.83% when $N_t = 5$. Ordering the samples therefore has the advantage of constraining the search to the region of the optimum, which is increasingly attractive as the number of transition samples rises, although the corresponding increase in difficulty in finding the optimum means that the problem still gets harder overall.

4.2.1 The Fitness Function

The performance measure adopted initially was that used by Rabiner, Gold and McGonegal [5] when producing their tables of transition sample values, namely the maximisation of the minimum stopband attenuation. The fitness was simply defined as:

$$fitness = -20 \log_{10}(ds) \tag{4.2}$$

where ds is the stopband ripple and $fitness$ is the magnitude of the attenuation in dBs.

4.3 Extensions to the Floating-Point GA

The specialised floating-point crossovers mentioned in Section 3.11 are not able to produce offspring whose genes have values *outside* those of their parents. This means that genes will always tend to move towards the centre of the search space after repeated crossovers. This effect will be particularly damaging for those genes whose optimum values lie towards the edges of the allowable range, namely 0–1. To counter this, the crossover action was altered slightly to allow the genes to move apart on crossover as well as together. If the chromosome in Equation 3.6 is crossed, instead of the results shown in Equation 3.7, the new crossover produced:

$$\begin{aligned} c_i^* &= \{a_1, a_2, \rho.a_3 + (1.1 - \rho).b_3, \ldots, a_n\} \\ d_i^* &= \{b_1, b_2, \rho.b_3 + (1.1 - \rho).a_3, \ldots, b_n\} \end{aligned} \tag{4.3}$$

where ρ is a uniform random number in the range 0–1. This extension made it easier for the GA to find solutions which contained at least one transition sample with a very high or low value. This becomes more of a problem as the number of transition samples increases and the samples have to form a smooth transition from pass- to stopband.

4.4 Simplex method hybrid hill-climber

Early runs showed that the GA was only able to approach the optimum filter specification for filters with a few transition samples, and that for filters with four transition samples or more, the best solution found by the GA became progressively poorer. To help combat this, a hybrid hill-climbing search was added to the GA to perform the final optimisation.

The local search technique which was adopted for the FS filter design GA was the Simplex method hill-climber [33], described in Appendix A.2. This method was chosen

because it does not require gradient information, but, like the GA, it simply takes point samples of the space. This means that it can be used to perform numerical optimisations of complex problems which are hard to analyse mathematically. Unlike many other methods of hill-climbing it does not make use of any form of curve-fitting, such as parabolic interpolation. To make use of parabolic interpolation, it must be known beforehand that the space is at least a reasonable approximation to a parabola, which implies some prior knowledge of the structure of the space, which is not necessarily available. The Simplex method does not require any pre-knowledge of the structure of the space, and can therefore be used on any problem without having to perform any analysis of it to determine its suitability.

The Simplex method of hill-climbing uses a number of point samples equal to one greater than the dimensionality of the space being searched (i.e. three points in 2-D space etc.) These are initially set up at random around the best point found so far by the GA, which is itself included in the set. These points now bound a solid shape, which is analysed to determine the worst point and best face. A number of attempts are then made to find an improved point in the space by performing a series of geometric transformations on the worst point in the shape until it is improved or cannot be improved. This process is repeated until either a fixed number of function evaluations have taken place, or the relative difference in performance between the best and worst points drops below a predefined level.

In practice it was found that once the Simplex local search had been used once, it made such an improvement that the best solution then lay in a very small, highly-fit region, and crossover and mutation almost always moved the offspring to poorer solutions. This often resulted in the GA being unable to perform satisfactorily once the local search had been called on once, and the local search was then generally relied on to perform the final optimisation.

4.5 Extensions to the crossover selection scheme

The five floating-point crossover types described in Chapter 3 fall into two categories: those that simply exchange or swap genes between parents to produce the offspring, and the three forms of Arithmetic crossover, which actually combine the parent chromosome's gene values in some way to produce offspring. The latter forms introduce new gene values, while the former ones do not. In order to allow the GA to determine which of the various types of crossover were of most use at each stage of the run, a scheme was devised to dynamically adjust their selection probability according to their performance as the run progressed. This replaced the original selection mechanism, where the crossover type used was chosen using a uniform random scheme.

Under the new scheme, the initial selection probability of each type was set equal. The total number of times each type of crossover was performed was then stored, along with the number of times each produced at least one offspring with a higher fitness than its parents'. The probability of each crossover type being selected was then given by:

$$p_x = \frac{d_x}{\sum_{l=1}^{5} d_l} \tag{4.4}$$

where p_x is the probability of choosing crossover type x, d_x is the proportion of calls to crossover method x that produced a fitness improvement over the parent chromosomes, and the summation is over the five types of crossover. The sum of all five p_x probabilities equals one, and this allows the best crossover method to be selected dynamically during the run by a roulette wheel selection. A scale is drawn up between zero and one, where the gap between the divisions is proportional to each p_x, as illustrated in Figure 4.2. A random number is selected between zero and one, and depending on which division it lands in, that crossover method is used. Here a random number of 0.4 selects crossover method three. This method is similar to that of Davis [75], but does not include the explicit hierarchical probability allocation for the crossover methods which produced the parent chromosomes, and to those which produced their parents, and so on. The approach used here was much simpler to implement but was found to react satisfactorily

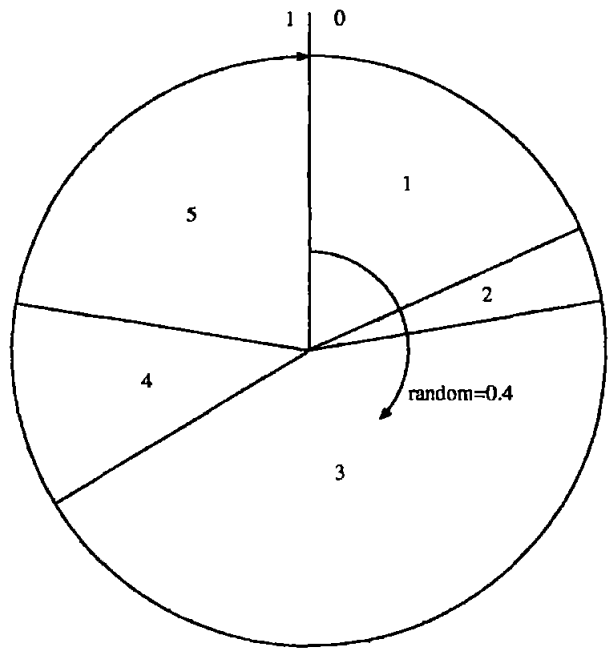


Figure 4.2: Method used to dynamically select the crossover method to use.

to the changing crossover performances in this application.

When the number of transition samples (N_t) is small ($< \sim 5$), the Arithmetic crossover types, which actually alter gene values, perform best early in the run. Later in the run the other types (which simply exchange genes unchanged) and pure dynamic mutation became dominant. This occurred because the GA is initialised with just a subset of the total number of possible gene values. In order for the GA to perform useful optimisations, it must use these relatively few values to generate a wide range of others in between to allow it to search the space effectively, so the crossovers which perform best initially will be those which introduce new genes into the population. Once the population has converged to the region of the optimum, it mainly contains genes which lie in the region of the optimum, so the crossover methods which only seek to combine the existing gene values will increase in usefulness. After the population has converged, crossovers which alter gene values, particularly those between dissimilar parents, will often move the resulting chromosomes away from the region of the optimum, so their offspring will have a poorer performance than their parents, and their selection probability will decrease.

Since mutation occurs at a much lower rate than crossover, if crossover and mutation both occurred on a single chromosome and the offspring improved on their parents' fitnesses, then the credit for the improvement was given to the crossover. The mutation selection probability increased once the population had converged because of the disruptive nature of the Arithmetic crossovers. Once the selection probability favoured the non-Arithmetic forms of crossover, mutation became the main gene-altering action, which was then able to apply small changes to the genes and so gradually improve their fitnesses. The dynamic form of mutation, which reduced its disruptiveness as the run progresses was used here, as it enabled very small changes to be applied after the population had converged, which were then not very disruptive.

When N_t is larger ($> \sim 5$), the GA found that all of the crossover types initially have a similar performance, but the same crossovers and mutation take over later on as became dominant for lower N_t. The gene values in the initial population are available to more than one gene position due to the gene reordering which can occur. This effect increases as N_t rises and the transition sample values get closer together. Since the population therefore effectively contains more gene values than for smaller N_t, the simple action of swapping them will be more productive than before, and allows the better initial performance of the non-Arithmetic gene-swapping forms of crossover.

These results have a parallel with the predicted action of the GA, where at the initial stages of a run, the search should cover a wide range of gene values, while at the end, once the search has converged to a good region of the search space, the emphasis shifts to trying to bring good building blocks together to form a near-optimum solution.

The inclusion of this selection process allowed the Genetic Algorithm to make more regular improvements in fitness, although it was still unable to find very good solutions for filters with more than three or four transition samples as the optimum region was so small that it was hard to hit it when performing crossover and mutation. As the optimum was already being found, the best solution found overall did not improve under this scheme, but more work was done by the GA for some filters.

Filter type	N	Band edge sample no.	Transition samples	Attenuation (dB)	Passband ripple (dB)
Type-I, highpass	49	18	4	128.026	0.175
Type-II, lowpass	48	8	4	124.041	0.099
Type-I, bandpass	128	20,15	3	85.526	0.105
Type-II, bandstop	65	10,8	4	109.568	0.051

Table 4.1: Results for recursive-form FS FIR filters designed by GA. N is the number of filter coefficients.

4.6 Results for FS filters

Some example filters designed by the hybrid GA-Simplex method are given below, in Figures 4.3–4.6. Information about these filters is summarised in Table 4.1. The GA was able to find filters with a performance which at least equalled those tabulated by Rabiner et al [5]. The crossover probability was 0.7 and the mutation probability was 0.01, the population size was 30, and the GA was run for 1,000 generations. This relatively high mutation rate did not cause excessive damage to the solutions found as the effects of the dynamic mutation method decreased rapidly as the number of generations increases.

Further results are given in Tables 4.2–4.6, comparing results from the hybrid GA with those of Rabiner et al [5] for a range of lowpass filters. As before, all runs were for 1,000 generations, with mutation and crossover probabilities of 0.01 and 0.7 respectively, and a population size of 50.

The improvement of the maximum and average fitnesses with generation can be seen in Figure 4.7 for a typical run to design a Type-II highpass filter with $N = 89$, a narrow stopband of three samples, and five transition samples. The run of 1000 generations was completed in around four minutes, although a near-optimal solution was found after about generation 400. The regularly-spaced peaks in the latter three-quarters of the graph appear when the local search routine was called after the Genetic Algorithm had failed to improve the best fitness for 20 generations. The Genetic Algorithm is

Source	Attenuation	TS 1	TS 2	TS 3	TS 4
Rabiner	127.367	0.71883166	0.25469056	0.03717696	0.00131836
GA	146.099	0.703932	0.233668	0.030297	0.000877

Table 4.2: Comparison of results for a Type-I, $N = 16$ filter, with a passband width of 1 sample and 4 transition samples.

Source	Attenuation	TS 1	TS 2
Rabiner	67.131	0.59911696	0.0937500
GA	67.204	0.599416	0.109632

Table 4.3: Comparison of results for a Type-I, $N = 33$ filter, with a passband width of 3 samples and 2 transition samples.

Source	Attenuation	TS 1	TS 2	TS 3
Rabiner	88.256	0.72436684	0.25203440	0.02576904
GA	89.591	0.723101	0.249866	0.025017

Table 4.4: Comparison of results for a Type-I, $N = 65$ filter, with a passband width of 8 samples and 3 transition samples.

Source	Attenuation	TS 1	TS 2	TS 3
Rabiner	94.764	0.67475127	0.19093541	0.01556396
GA	94.994	0.664858	0.177726	0.012207

Table 4.5: Comparison of results for a Type-I, $N = 125$ filter, with a passband width of 1 sample and 3 transition samples.

Source	Attenuation	TS 1	TS 2	TS 3	TS 4
Rabiner	108.297	0.82096794	0.40820056	.09324160	0.0606079
GA	111.829	0.81153	0.392362	0.085818	0.005236

Table 4.6: Comparison of results for a Type-I, $N = 128$ filter, with a passband width of 16 samples and 4 transition samples.

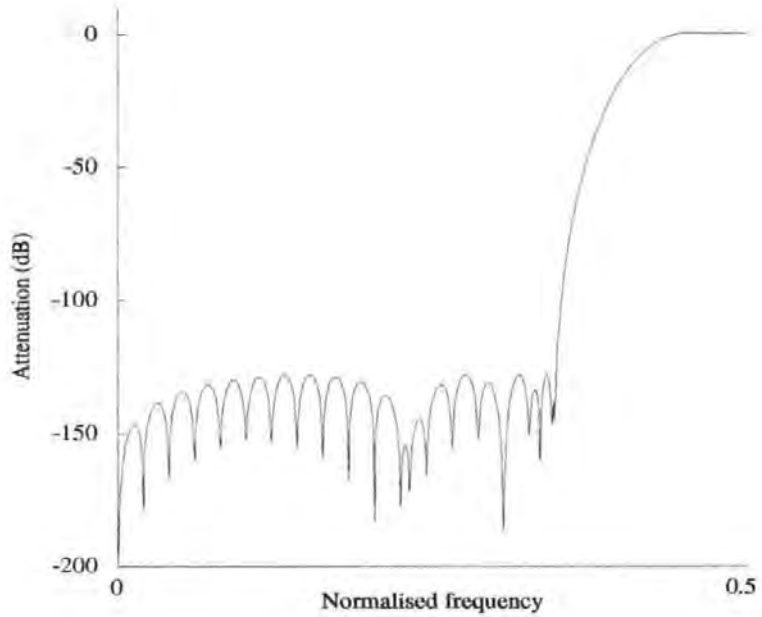


Figure 4.3: Type I, Highpass FIR filter with four transition samples.

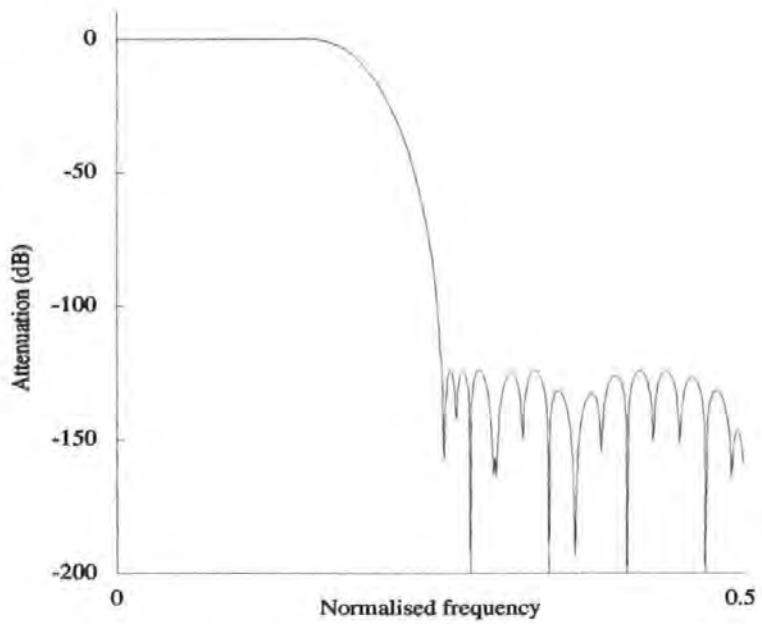


Figure 4.4: Type II, Lowpass FIR filter with four transition samples.

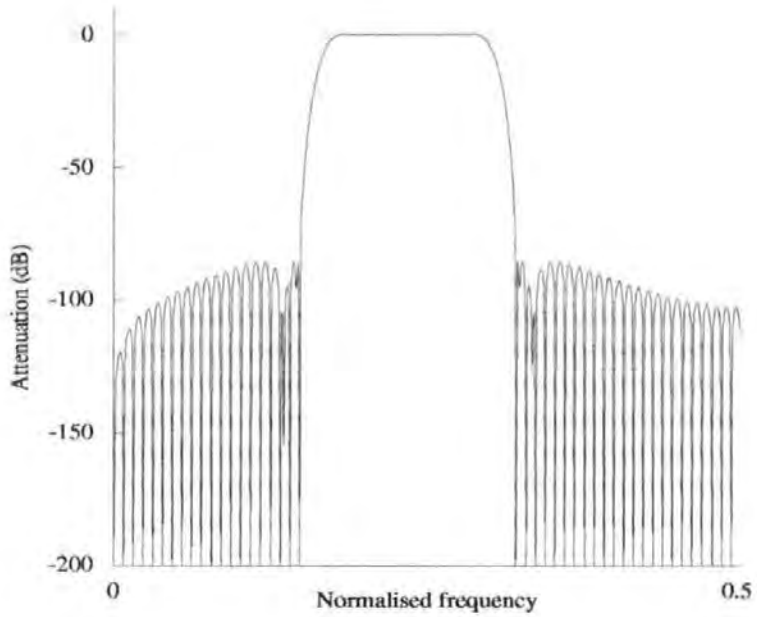


Figure 4.5: Type I, Bandpass FIR filter with three transition samples.

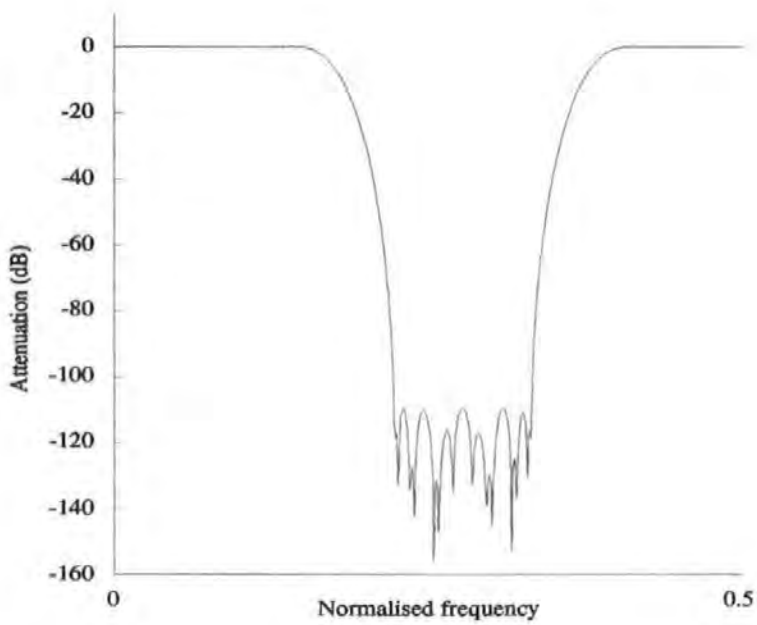


Figure 4.6: Type II, Bandstop FIR filter with four transition samples.

able to find the area of the optimum fairly quickly, but has difficulty finding very high fitness solutions within the very small area of the peak, as crossover is likely to throw the offspring into comparatively very poor regions. The Genetic Algorithm is therefore used alone for the first quarter of the run, after which the hill-climber is able to make substantial improvements.

It has been found that this technique is generally very robust for filters with up to six transition samples, taking an increasing length of time as the number of transition samples rises. For filters with up to around four transition samples, the Genetic Algorithm is able to find good solutions very quickly, without the intervention of the local search. For five or six transition samples, the Genetic Algorithm performed a useful amount of improvement, although not to a good performance, while the local search was able to complete the optimisation. For more transition samples (up to ten were used), the hybrid Genetic Algorithm was only able to perform a small amount of optimisation, from which the local search was also unable to find the optimum within a reasonable time. Improvements in the speed of personal computers may, however, make the technique more viable.

The hybrid Genetic Algorithm has been able to produce results which range from equalling the performance of those in the literature, to improving on them by up to around 20dB minimum stopband attenuation, as tabulated on page 61. However, its main strength lies in the fact that it can quickly produce untabulated filter coefficients, which are much more useful and will have a better performance than those found by interpolation of the published results. It is also able to design filters with more transition samples, showing that the hybrid Genetic Algorithm is a suitable technique to use for designing this type of Finite Impulse Response filter, although the GA alone is of little value, and a straightforward hill-climber could be used alone with equally good results.

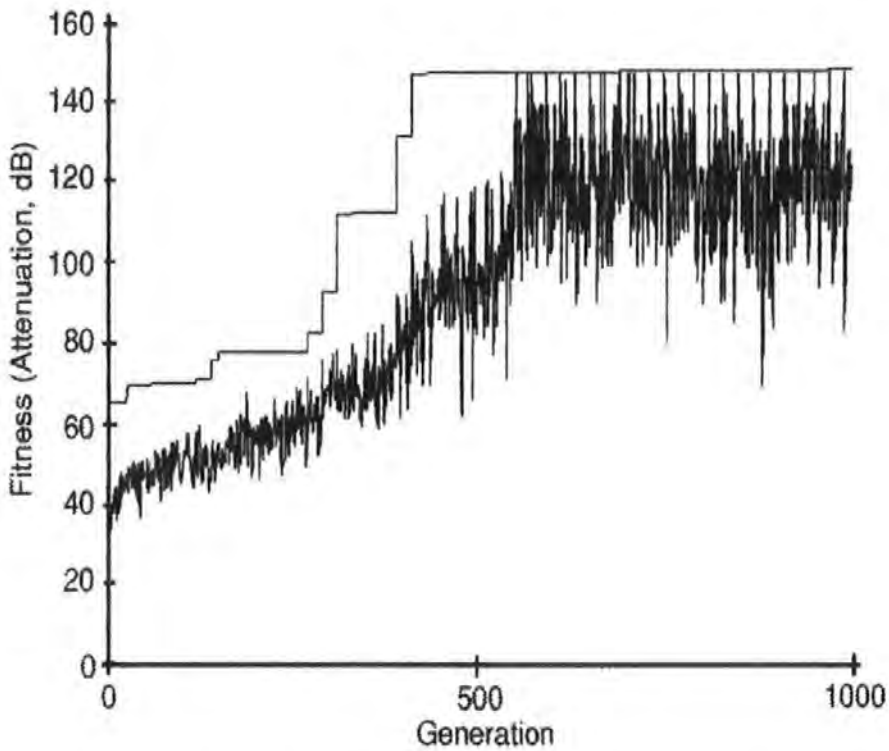


Figure 4.7: Improvement of the maximum (upper line) and average (lower line) fitnesses with generation for a non-recursive Type-II filter.

4.6.1 The FIR Filter search space

In order to understand the problem more fully, and to enable a better understanding of the GA's performance, data was collected from the search space that the GA was searching. Since we are using a real-coded chromosome, the search space and parameter space are actually the same for this problem. As has been shown, the actual proportion of the total space that the GA is restricted to shrinks quickly as the number of transition samples rises, although it was discovered that the search space becomes more difficult, as the high-fitness region becomes proportionally smaller.

For a single transition sample, it can be seen (Figure 4.8) that the general shape of the search space with the chosen fitness function is a concave peak, a characteristic shape which was to recur in the structure of the higher-dimensional surfaces. In the case of a two transition sample filter, the surface is given in Figure 4.9. The peak can be seen to lie on a straight concave ridge, whose cross-section is also concave. The true optimum point on the ridge can be seen to lie close to the edge of the search space, a property which initially caused the floating-point GA some problems, as the original crossover techniques tended to move offspring towards the centre of the space, and not towards the edges, as shown previously in Figure 3.2.

When the three-dimensional case is examined, it can be seen that there is a high-fitness plane running through the permissible area, with a concave cross-section; within this 2-D high fitness plane is a high fitness line, along which lies the optimum solution; again, both have the same shape cross-section. The unimodal nature of this space indicates that a pure hill-climber should be able to find the optimum. The hill-climber used by Rabiner, Gold and McGonagal optimised each parameter in turn, holding the others fixed, but the angle between the high-fitness ridge and the axes means that many small steps parallel to each axis in turn will be required to reach the optimum. Other similar techniques which rotate the axes to align them with the best direction to climb in require the calculation of gradients which may be difficult or time consuming for complex models. Curve fitting methods such as parabolic interpolation fail for this

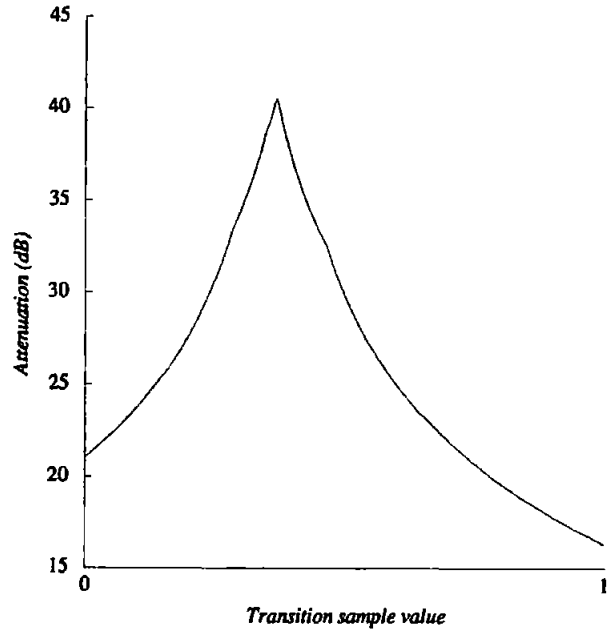


Figure 4.8: Search space for a one transition sample FIR filter.

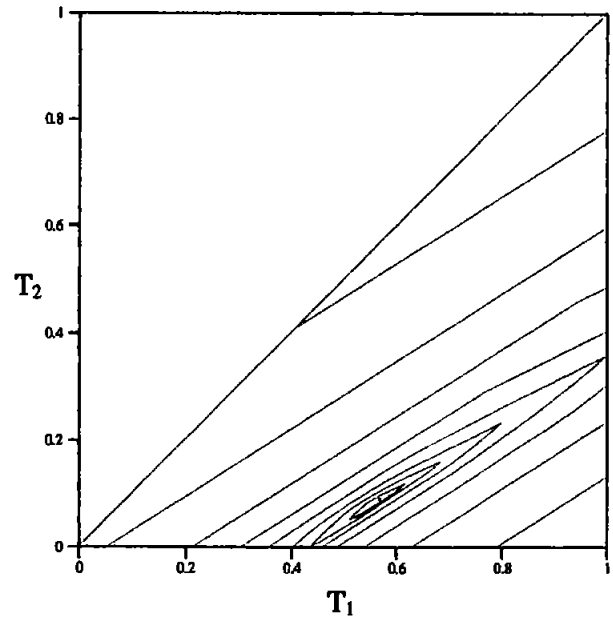


Figure 4.9: Search space for a two transition sample FIR filter.

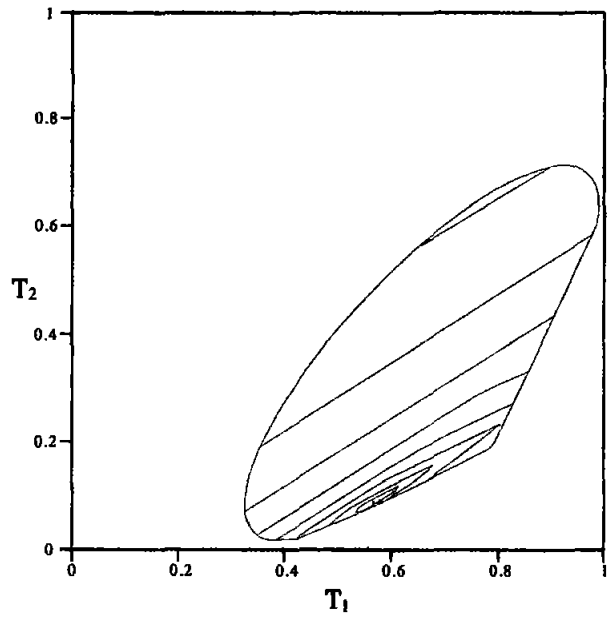


Figure 4.10: Search space for a two transition sample FIR filter with additional constraints.

problem since the space becomes an increasingly poor approximation to a parabola as the optimum is approached. The Simplex Method makes no such assumptions and can move in any direction, so was able to work successfully in this type of space.

In order to constrain the GA still further, filters in which the *interpolated* response did not change monotonically between the pass- and stopbands were also rejected. This reduced the allowable region still further (Figure 4.10), but this did not help the GA at all, as the optimum is now on the very edge of the allowable region. The action of the standard form of floating-point crossover was to move points together, which makes it difficult to find such points. The extended form of crossover used here was able to move the offspring to regions outside those bounded by the parents, so it should have been more able to find points on the edge of the allowable region. This is however still not able to perform well, as the optimum lies so close to the edge of the allowable region that it is hard for crossover to hit it and not go too far and into the disallowed region. Figure 4.11 compares the increase in fitness with generation for typical runs of constrained and unconstrained optimisations of an 68-coefficient, six transition sample lowpass filter, showing that the constrained optimisation has a lower performance.

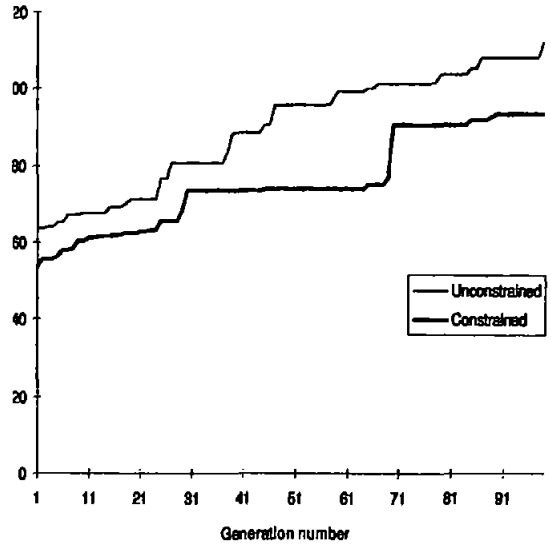


Figure 4.11: Improvement of the maximum constrained and unconstrained attenuations with generation, for a non-recursive Type-I lowpass filter.

4.6.2 Concurrent Optimisation of the Wordlength

In order to integrate a further filter design step, the fitness function for optimising recursive filters was extended to incorporate the finite wordlength effects of coefficient quantisation by including an extra gene in the chromosome which determines the wordlength at which the genes will be decoded. This allows the Genetic Algorithm to search for the minimum wordlength necessary to achieve a given filter specification simultaneously with its coefficients. Mixed-integer programming has proved successful at optimising quantised coefficients [76], although separate runs are required for each wordlength under investigation. Stubberud and Leondes [77] have devised a Lagrange multiplier-based method for designing Frequency Sampling filters which also accounts for finite wordlength effects, but only approximates linear phase. Our approach allows the Genetic Algorithm to search for the minimum wordlength and optimum coefficients for linear-phase filters simultaneously, with no user intervention.

Since we are using a real-coded chromosome with genes in the range 0–1, the wordlength gene must be decoded to give an integer wordlength, which is performed by scaling it up to 0–24 and taking the nearest integer. The real-coded genes are then

quantised to this wordlength before being used to calculate the filter response and then fitness. This simplifies the search by limiting the number of points the Genetic Algorithm has to examine. It has the disadvantage that the Simplex hill-climber is less effective, because the search space is now made up of a large number of flat regions. This is due to the quantisation of the coefficients causing finite ranges of the floating-point gene values to be interpreted by the fitness function as having the same value, so all coefficient values in this range will have the same effect on the filter response. At the beginning of a search, the hill-climber is able to perform well because from a large scale perspective the surface is smooth, however once the search contracts around a good region, the small plateaux become increasingly apparent, and eventually the search cannot gain any information about the direction of the optimum and so is unable to reach it. The Genetic Algorithm is still able to perform successfully in such a space (which resembles the de Jong Genetic Algorithm test function f3 [48]), as it only relies on point fitness samples and is unaffected by discontinuities or perfectly flat areas. This implies that more reliance will be placed on the Genetic Algorithm to perform a good optimisation since the Simplex will be less effective here.

When calculating the filter response, in order to maintain filter stability, the radius was reduced to one less the quantisation interval:

$$r = 1 - \frac{1}{2^B} \tag{4.5}$$

where B is the wordlength. The fitness now has to take account of both the magnitude response and the wordlength, with the emphasis on the former since this constraint should be satisfied regardless of the wordlength. To this end, the following scheme was devised.

Firstly, the normalised magnitude response is examined in both pass- and stopbands to see if it fits within the desired limits. If it does to within 10^{-5}, which corresponds to a deviation of only around 0.3dB from a desired attenuation of 70dB, then the basic fitness is set to 10^5, otherwise it is set to the reciprocal of the normalised deviation. This gives a main fitness range of 0–10^5, for the magnitude response, and is flat (at

10^5) for all filters fitting within the desired specification. By having all satisfactory solutions return the same fitness it means that the Genetic Algorithm is free to return a solution which only *just* fits the design specification, leaving it more freedom to reduce the wordlength. To account for the wordlength, a further term is added to this, consisting of 25 minus the wordlength. This overall fitness function therefore has extra structure, especially within the optimum peak region, which allows the Genetic Algorithm to search for the minimum wordlength. The overall fitness function can be written:

$$f(x) = \begin{cases} (25 - B) + 1/e_{max} & e_{max} > 10^{-5} \\ (25 - B) + 1/10^{-5} & e_{max} \leq 10^{-5} \end{cases} \tag{4.6}$$

where B is the wordlength, and e_{max} is the maximum absolute error between the normalised filter response and the desired response in the pass- and stopbands.

This approach places the major emphasis on the optimisation of the magnitude response, and once this has been achieved, the effect of the wordlength dominates (within the optimum fitness 'plateau'). Other weightings have been tried but these were found to allow the Genetic Algorithm to perform efficiently, without the intervention of the Simplex local search.

Results for a typical test run are given below in Table 4.7, for a 49-th order, four transition sample lowpass filter, with a bandwidth of 0.25. The Genetic Algorithm was able to fit to the desired specifications with coefficients quantised to a wordlength of only six bits. The full-precision fitness function (in which the Genetic Algorithm only seeks to minimise the stopband ripple) was able to find a solution with much greater stopband attenuation as the last transition sample was able to have a much smaller non-zero value, as shown in Table 4.8. The full-precision fitness function used the same radius as the 6-bit solution, and the frequency responses of the quantised filter is shown in Figure 4.12. Advances in the speed of personal computers means that an exhaustive search is now feasible even for a 4-transition sample, 16-bit filter, with a total search space size of $4 \cdot 2^{16} = 262,144$ filters.

	Desired	Quantised	Full-precision
Passband ripple (dB)	0.1	0.058	0.129
Stopband attenuation (dB)	77	82.96	117.26
Wordlength	-	6	-
Radius	-	0.984375	0.984375

Table 4.7: Desired and optimised specification for a quantised-coefficient and full-precision filter. The full-precision design uses a maximum-attenuation fitness function.

Quantised	Full-precision
0.875	0.785269
0.515625	0.348808
0.15625	0.065764
0.015625	0.003069

Table 4.8: Transition samples for the filters described in the text and Table 4.7.

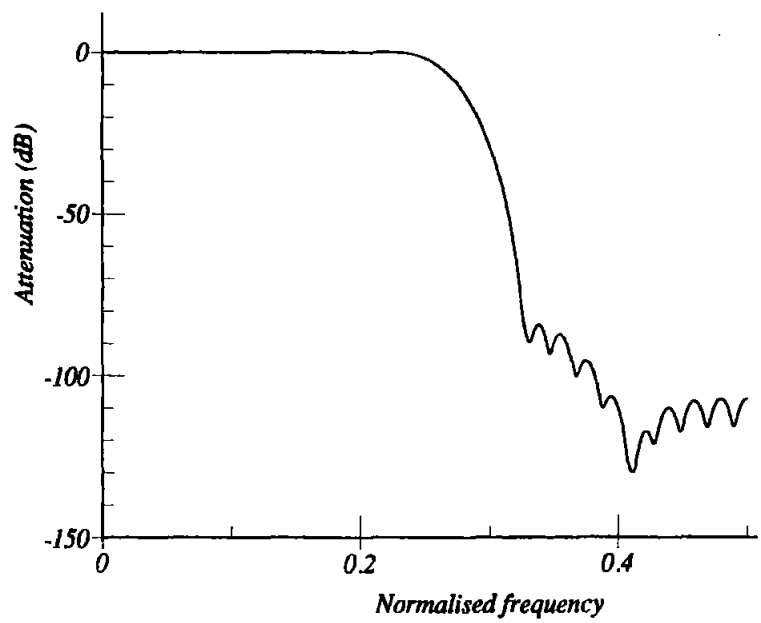


Figure 4.12: Frequency response of a quantised coefficient FS filter designed by GA.

4.7 Conclusions

It has been shown that the GA was able to discover reasonable solutions on its own for filters with a low number of full-precision transition samples, but that for filters outside this limited specification its performance was poor. The underlying problem is that the GA requires a chromosome from which you can take a portion and use that to make a prediction about the fitness of the whole chromosome. In this case, when there are only a few transition samples, it easier to do this as the whole chromosome is small, but when the number of transition samples rises, it becomes increasingly difficult, because the filter response depends on the whole set of transition samples together, and it is not possible to predict the filter response from just one or two adjacent transition sample values.

The addition of a hybrid hill-climber produces excellent results because it is highly suited to this kind of optimisation problem. Since there is only a single peak, the search space is unimodal and running a hill-climber from any point in the search space will always find the optimum. This finding, together with the massive speed increases in personal computers means that it is no longer necessary to rely on the published tables of Rabiner, Gold and McGonagall [5] to find FS filter coefficients, as the coefficients of any Frequency Sampling filter with up to perhaps seven or eight coefficients can be found in a few minutes. However, for the vast majority of applications, four or five transition samples is adequate, so this restriction is not a problem, and the GA/Simplex method is an ideal technique.

The results found by the GA alone for quantised-coefficient recursive filters were reasonable due to the smaller search space, while the Simplex search was less efficient due to the quantised nature of the search space. However, as the calculations used to find the filter responses were full-precision, the coefficients cannot be assumed to be optimal for a system where the calculations are quantised throughout, and a fuller analysis would have to be added to the fitness function before the results were fully representative of the optimisation of a fixed-precision filter. They do however show

that the GA appears to be suitable for optimising the coefficients of this kind of filter.

Chapter 5

IIR Coefficient Optimisation by GA and SA

5.1 Introduction

Having shown that the GA is able to optimise a limited range of quantised-coefficient FIR filters, the next step taken was to select a further optimisation task, in order to extend and expand the investigation into the capabilities of the GA with respect to the design of digital filters.

It was decided to examine an IIR filter, as these have a more complex search space and would provide a harder test of the GA's abilities to perform suitable optimisations. The IIR not only provides the opportunity to optimise the coefficients of a different type of filter, but also the finite wordlength effects, which have a far greater effect on IIR filters than FIR.

The filter structure first selected for optimisation was cascaded second-order canonic (or 'direct form 2') sections, as in Figure 2.6. First-order sections could be obtained by setting $a_2 = 0$ and $b_2 = 0$, but these were not included explicitly as a separate section type for simplicity.

5.2 Use of the GA

The original floating-point GA, that had been used to optimise the coefficients of the FS FIR filters, was used to optimise the coefficients of IIR filters made up of cascaded second-order canonic sections as shown in Figure 2.6. By choosing this IIR filter structure, with complex pole-zero pairs, the original floating point GA only needed to optimise four numbers per section, namely the radius and positive angle of one pole and one zero, as the complex conjugate pole and zero could be easily obtained from these. The chromosome used genes in the range 0–1, which were scaled to the required ranges, namely 0.5–1 for the radii, and 0–π for the angles.

The results from this approach were unsuccessful, due to the nature of the search space. This only contains useful information which can guide the GA around the optimum regions themselves, so a random initialisation and initial selection would produce a population with no coherent good genes. It was also found that the high fitness region around the optima only covers a small proportion of the total space, as the fitness of a solution drops rapidly as a good solution is perturbed. This proportion also decreases as the number of sections increases. This means that the initial population of random points contains little coherent information about the good regions that the GA should be exploring, so it cannot perform adequate optimisation. The inclusion of the Simplex method hill-climber was successful in improving the best solution found if the GA was very close to the optimum, but cannot be used on its own as the search space has multiple peaks. The inclusion of on-the-fly quantisation of the coefficients was unsuccessful in improving the GAs performance, and reduced the quality of the best solution found as the Simplex search cannot be used with quantised values.

Since the GA was either very slow or unable to reach the region of the optimum, and very fast methods of designing IIR filters, such as the BZT, are readily available [11], it was decided to use a BZT design as a seed for the GA. Initially a single quantised BZT design was used, but it was found that this solution immediately overran the population as its fitness was far higher than any of the random solutions in the first population,

and the GA did not progress. The initialisation was therefore changed to seed the GA with BZT solutions which had been perturbed by a random amount of up to 5%. This led to a much better performance by the GA, even though the final coefficients were often quite different to those it was seeded with. Arslan and Horrocks' [78] similar approach to IIR optimisation also found that it was necessary to initialise the GA with perturbed copies of quantised, full-precision coefficients. A GA-based method for finding the best way of quantising full-precision lattice and direct form coefficients has been developed by Aketa et al [79]. Although this does not extend the search to the total coefficient optimisation investigated here, it does have the advantage of being able to weight the performance deterioration to minimise its effects across a specified region.

It was decided that a better approach might be the direct optimisation of quantised coefficients, as the GA theory Chapter 3 predicted that a binary-coded GA should be most efficient. To this end, a binary-coded GA was written with the same aims as the first, real-coded one. The program was written to explicity accept even-order filters only, although first order sections could be generated by setting a_2 and b_2 to zero.

This binary GA was used to optimise the four variable coefficients (two a and two b, as a_0 is always one when complex conjugate pole and zero pairs are assumed) of each section by using a chromosome of length $4BN/2$ where B is the wordlength (number of bits) of each coefficient, and N is the order of the filter. The fitness function extracts them from this chromosome in order to calculate the filter's response and so the fitness. In order to add flexibility to the design process while retaining the ease-of-use approach we are aiming for, the order of the filter could be specified in two ways: either the user can specify the number of second-order sections to use, or the GA can use the order suggested by the BZT, rounded up to an even number as we currently only use second-order sections. Using the order suggested by the BZT will in general enable the GA to find a solution fitting the design specification, although if the tolerance is very tight, the coefficient quantisation may push the filter into instability. This means that the GA will be unable to find a suitable solution, so the user will have to increase either

the order or wordlength until a solution can be found.

The chosen fitness function was:

$$Fitness = \frac{1}{\epsilon_{max}} + \frac{M}{\sigma_{or}^2} \tag{5.1}$$

where ϵ_{max} is the maximum error in the frequency response across the stop- and pass-bands, M is a scaling factor, and σ_{or}^2 is the roundoff noise gain of the filter. Both the error and noise terms are limited to a value of 10^{-6}, although the noise is never this low in practice. The limiting of the frequency response error means that once a filter has been found with a suitable response, the fitness function is simply dependent on the noise factor. It was hoped that this would make it easier for the GA to find a low-noise filter, but this was not found to be the case, as described below. Filters with any section whose b coefficients lay outside the stability triangle of:

$$\begin{aligned} 0 &\leq |b_2| < 1 \\ |b_1| &\leq 1 + b_2 \end{aligned}$$

were immediately discarded with low fitness.

5.3 Results

It was found that the GA only performs well and finds filters within a given specification for very loose tolerance filters, i.e. filters with wide transition widths or low desired attenuation. This poor performance is discussed further in Section 5.3.1, which looks at the nature of the space that the GA is searching. The connection between the a and b coefficients and the filter response is not straightforward, and there is a high degree of nonlinearity. As the specification becomes tighter, the performance drops off as the search space gets harder.

A second approach to the optimisation problem was used to test that the filters were achievable, namely *Simulated Annealing* (SA) [64]. SA is a hill-climbing technique which also allows occasional moves to a poorer solution, thereby permitting the search

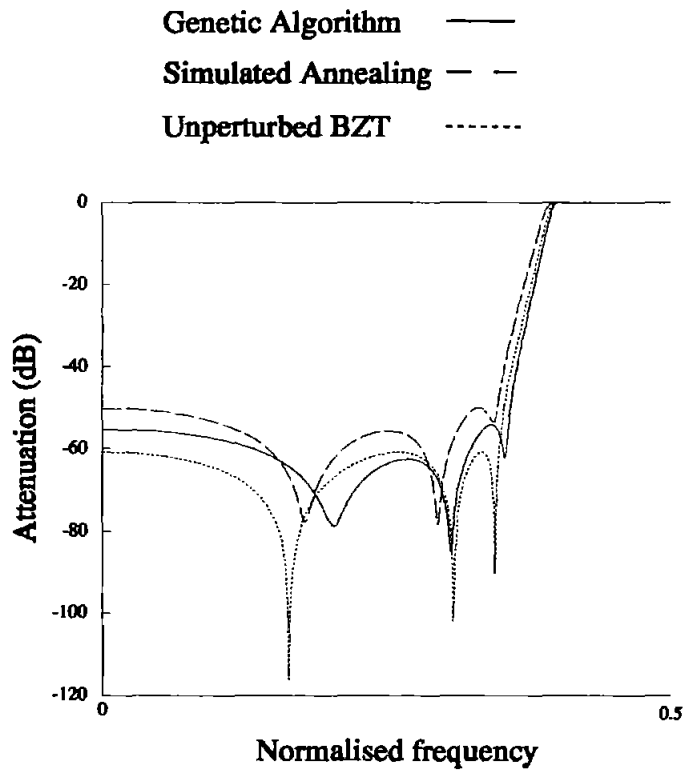


Figure 5.1: Comparison between the frequency response for fixed wordlength filters designed by SA, GA, and BZT.

to escape from local minima. The degree by which the fitness can worsen decreases over time, which allows SA to look widely over the space initially, but confines it more closely to the peak or peaks it finds as time progresses. The implementation of SA that was used is derived from the Simplex Method [33], and used quantised coefficients.

Figure 5.1 shows the filter responses for a fixed wordlength filter designed by SA, GA, and the filter obtained by quantising the coefficients as found by the BZT. The GA was run with a population size of 40, mutation and crossover probabilities of 0.005 and 0.6 respectively, for 3,000 generations. It can be seen that the BZT solution had the highest attenuation, followed by the GA and then the SA, while the GA's passband ripple of 0.27dB was larger than the BZT solution of 0.103dB. In fact, the SA solution was only just within the requested tolerances of 50dB attenuation and 0.1dB passband ripple, which gave it more freedom to reduce the roundoff noise gain, which was an order of magnitude lower than for the other two methods.

Although the results are all similar, the better performance of the SA suggests that

there might be some problems with this GA approach: firstly, the fitness function may not be suitable, or secondly, the GA may not be as suitable for this problem as SA. In order to determine which is the case, a future investigation could alter the fitness function to penalise filters whose responses lie far from the desired frequency response *in either direction*, as this can reduce the performance of the filter in other respects. To remove the dependency on the combined magnitude response/roundoff noise gain fitness function, the use of an extended GA to produce a range of solutions with varying tradeoffs between the two criteria will also be investigated (see Chapter 6).

The poorer performance of the GA is also influenced by the hard search space used by the fitness function, which is derived from the minimax error from the desired filter response. It has been found that the mean square error between the filter response and the desired response is generally used as a measure of performance [21, 80], which is an easier problem as there are more possible filters which give the same mean square error, so finding one with the optimum value is more straightforward.

5.3.1 The IIR Filter Parameter Space

In order to determine why the GA performed poorly far from the optimum, the parameter space was examined to determine its characteristics away from the optimum. The search space for a cascade structure IIR filter is known to be generally multi-modal [32], and Chellapilla et al's investigation [24] showed that higher-order cascaded filters have more local minima, which will hamper the search.

The search space was examined at two points: the BZT solution with parameters moved at random by up to 5%, and purely random genes. Since the IIR chromosomes have many more genes than the Frequency Sampling FIR, it was decided to examine only slices through the search space, by fixing all but two genes and producing a fitness surface by varying the other two.

For the filter with 5% perturbed BZT coefficients (Figure 5.2), the surface has two clear peaks, one narrow, the other wide and flat. The best, narrow peak has a much

lower fitness than for the quantised BZT solution.

When the gene values are completely random (Figure 5.3), the peak fitness drops dramatically, and the surface becomes highly multi-modal, but with little variation in peak size. This means that far from the optimum, the GA has no useful information to go on, so it will be unable to function properly. A more serious problem is that the highest peak for one pair of genes moves as other gene values are altered, e.g. by crossover, so as the GA moves one gene towards its own optimum, the optimum value for others can move. This makes it harder for the GA to operate, as its targets are continuously changing as it runs, and it cannot build up a good set of genes as what defines a 'good' gene is also continuously changing.

For the pole position genes the effects are less severe, although the peak fitness reduces and extra peaks appear as the coefficients move away from their BZT-calculated values.

It is clear from these surface plots why the GA needs seeding with a good solution, such as that from the BZT, in order to reach a region of the search space with enough large-scale structure to allow it to search effectively. In future work it might be more effective to use the GA to optimise the filter structure, and use the BZT to generate the corresponding filter coefficients, from which the frequency, phase, and noise responses can be determined and used to calculate the fitness.

5.4 Discussion and Conclusions

As mentioned above, the randomly-initialised GA was not able to find suitable solutions due to the nature of the search space. In a similar fashion to the FS FIR filters, for the GA to be successful the chromosome representing the filter coefficients must be able to be broken down into smaller parts which can be used individually to predict the filter response. The chromosome used here, although it uses the pole-zero radii and angles, which have a more direct correspondence to the response of their second-order section than its a and b coefficients, still does not fulfil that criterion. This is due to

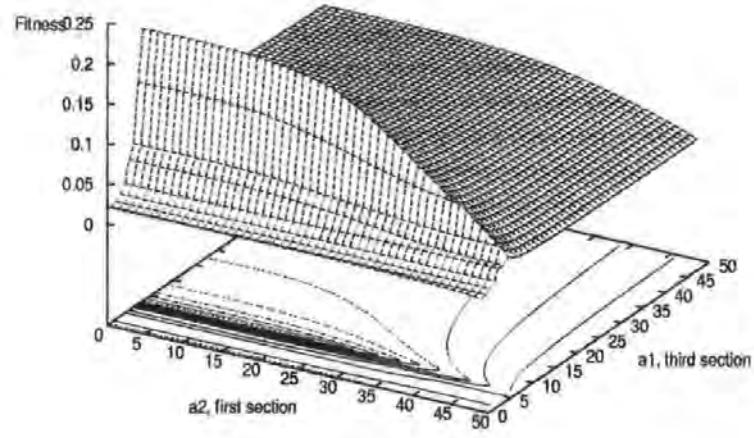


Figure 5.2: An example slice through the search space for a sixth-order BZT solution IIR filter, perturbed randomly by up to 5%.

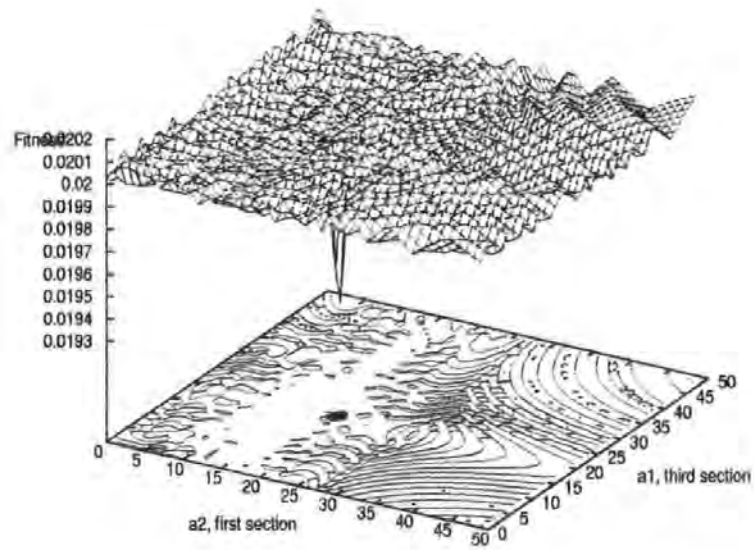


Figure 5.3: An example slice through the search space for a sixth-order random coefficient IIR filter.

the nature of the digital filter itself, which is dependent on its entire set of coefficients, taken together; it is not possible to predict a filter's overall response from a small subset of its coefficients or pole-zero positions, so neither can its fitness be so predicted. This leads to the GA being unable to determine if any part of a particular chromosome is better than the corresponding section of any other chromosome, so it cannot determine which of the randomly-selected initial population has suitable genes to carry on to the next generation, and the results are poor.

When the initial population is seeded with a BZT solution, the GA has a few good chromosomes, and hence genes, to work with, and it is able to make some progress, but again, as it relies on being able to predict a filter's fitness from a section of its chromosome, it is not able to work very efficiently. Although the GA was run until no improvements had been found for several hundred generations, time limitations on the maximum number of generations could have limited the GA's effectiveness, which future increases in computing power could overcome.

SA does not need to be able to perform the same performance prediction and simply looks at the performance of the chromosome as a whole. It is therefore able to perform more effectively in this situation than the GA, and, as was shown in Figure 5.1, is able to find a suitable solution which trades off the frequency and noise responses in the best way—the frequency response only just fits within the desired response, leaving more freedom for the noise to be reduced. As the performance with respect to each criterion has to be traded off against the others, this is the best possible result.

The GA- and SA-based approaches have the advantage of flexibility over a BZT approach, which requires that an analogue filter be found with the desired response, which may not be possible. Using the GA or SA, only a near approximation needs to be found by BZT, which can then be further optimised towards any desired response.

The initial results presented here suggest that, although the GA produced encouraging results, SA is probably the better technique. Further investigations into the use of SA techniques would be useful in determining its suitability over a wider range of IIR filter designs.

Chapter 6

Multi-criterion Optimisation

6.1 Introduction

The GA and SA techniques which have been used up to now have been strictly single-criterion optimisers. Although both the magnitude and noise responses were optimised by a GA in the previous chapter, this was achieved by simply adding the two measures together. The GA is, however, able to be extended in a way which could make it especially useful for filter design, by allowing it to optimise a solution with respect to several criteria simultaneously. As was seen previously, the filter design process involves several interacting steps, each of which can affect the filter's performance with respect to any of the others. It is possible to write the GA in such a way that it examines more than one performance measure at once, and attempts to generate either one compromise solution or a set of solutions with varying trade-offs, from which the user can select one.

6.2 Techniques

A variety of methods of performing multi-criterion optimisation (MCO) will now be examined, and their applicability to filter design analysed.

6.2.1 Weighted sum of fitnesses

This is the simplest form of MCO, and does not require any changes to be made to the standard single-criterion form of the GA, as it takes place wholly within the fitness function. Instead of the usual single performance measure, a number of them are taken in order to determine the chromosome's performance with respect to each criterion. In order to use these values in a single-criterion GA, they are then combined into a single weighted sum:

$$Fitness = w_1.f_1 + w_2.f_2 + \ldots + w_N.f_N$$

where w_i is the weighting and f_i is the fitness with respect to criterion i. This single value is then passed back to the GA, and used in the normal way to decide the individual's fate.

This method has the great advantage of simplicity, but it is not guaranteed to produce the desired solution without a great deal of user intervention. It is necessary to perform an iterative optimisation of the weightings, by running the GA, examining the solutions found, and then adjusting the weightings repeatedly until a suitable set of weightings is found.

This method is feasible for a single-purpose GA, which is only optimising two criteria, but for more complex applications the rapidly increasing difficulty of finding the correct weightings makes it increasingly unsuitable. This is especially true for a problem where the range of fitness measures is not known or is unbounded. While it may be possible to apply some additional meta-optimisation technique to the weightings to find the best set, this would add a large amount of complexity to the optimisation. It also has the disadvantage of only returning a single 'best' solution, with a single performance trade-off, when a variety of solutions with differing trade-offs exist. The following techniques are designed to search for this set of solutions, known as the *Pareto-optimal* or *Non-dominated sets* (POS, NDS).

6.2.2 The Pareto-optimal set

The weighted-sum approach described above finds a desired solution by the repeated adjustment of the weights applied to each criterion, and so only returns a single solution. Since, however, there are a range of solutions with different trade-offs, a better approach might be to find the best *set* of solutions, and allow the user to pick the most appropriate for their application. This not only removes the need for most of the user intervention, but also automates the trade-off process, turning a complex iterative process into a simple single-step one.

When using a model in which several performance measures are being optimised, it will often be the case that these will not be independent, so a change to the chromosome which alters the performance with respect to one measure will also alter the performance with respect to another. An example of this might be a filter in which the order and stopband ripple are both being minimised. A low order filter is not able to achieve such a low ripple as a longer one, so there is a conflict and it is not possible to satisfy both conditions simultaneously. Within the search space, for many, if not most of the solutions, both performance measures can be improved simultaneously, but there is a subset for which improving one *always* worsens the other. This set of 'best' solutions is known as the *Pareto-optimal set* (POS) [81].

The definition of the Pareto-optimal set is based on that of *domination*. A solution is said to dominate another if it has a better fitness in at least one measure, and at least the same fitness with respect to all the other measures. All solutions outside the POS are *dominated* by at least one solution within it, while no member of the POS is dominated by any other solution at all.

The concept of domination allows the mathematical specification of the POS, where a vector of fitnesses is dominated by another if the second is *partially less than* the first. A vector $\mathbf{x}$ of i fitnesses is partially less than vector $\mathbf{y}$ if:

$$(\mathbf{x} <_{\mathrm{p}} \mathbf{y}) \Leftrightarrow (\forall_i)(x_i \leq y_i) \wedge (\exists_i)(x_i < y_i) \tag{6.1}$$

i.e. for all i, $x_i \leq y_i$, and for at least one i, $x_i < y_i$. While the POS is the optimum

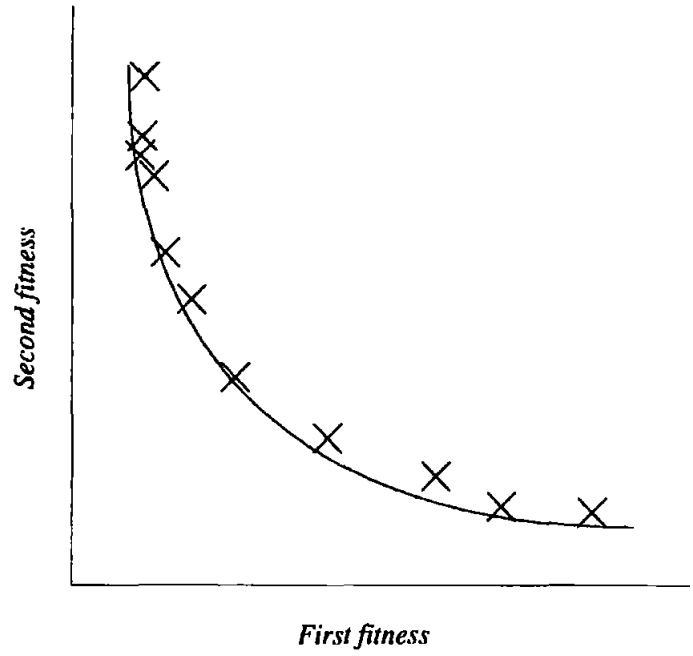


Figure 6.1: Example illustrating pareto-optimal and non-dominated sets.

set of solutions, the GA, by its random nature, is not guaranteed to find it. Within the GA's population, the undominated solutions form the *non-dominated set* for that generation.

An illustrative example is shown in Figure 6.1, for a two-criterion fitness function. The continuous line shows the position of the POS, and limits how far down and left the positions of the solutions found by the GA can go. The line is shown concave, but could be any shape, e.g. linear, convex, or stepped. The crosses on the diagram show the positions of solutions within the NDS found by the GA. Fitness functions for the various criteria must return smaller values for better solutions, and selection techniques developed which favour those solutions which lie closer to zero for each criterion.

Nicolson and Cheetham [82] have proposed a way of finding the POS by using known, good solutions as a seed for a conventional optimisation technique, 'inching' along the POS by changing the weighting of the different performance measures. This approach is very limited and suffers because adjacent solutions on the POS can have very different parameter values, making it very difficult for conventional searches to move between them. GA-based approaches are much more flexible, and can optimise

such 'niches' of dissimilar solutions on the POS.

6.2.3 Vector-Evaluated GA

The Vector-Evaluated GA (VEGA) developed by Shaffer [83], is a simple technique for searching for the POS. It is perhaps not a true multi-dimensional technique as only one criterion is examined at once, but the resulting solutions can be combined to give multi-dimensional results.

In VEGA, the population is split into as many sub-populations as there are fitness measures, and each sub-population's fitness is calculated with respect to a single fitness measure. The population is then recombined before selection and reproduction occur, causing those members of the population which have a high fitness with respect to their single fitness measure to be favoured. Under crossover, members which are highly-fit are combined, potentially producing solutions which have an intermediate performance with respect to a number of criteria, i.e. a trade-off has been performed. Throughout the run, the population is examined and the non-dominated solutions are stored, but these are not used to drive the search.

Although VEGA is simple to implement, it is limited in the range of solutions it produces, and the fitness functions have to be designed to return similar performance measures for what might be disparate aspects of the design. Because the solutions are selected by only a single fitness measure, it tends to find solutions which are clustered near to the axes. These therefore perform well with respect to one criterion, but have little trade-off of performance, unlike those solutions that lie between, which perform reasonably under all measures. This is illustrated in Figure 6.2, where the members of the NDS found by the VEGA GA can be seen to lie mainly close to the edges where they can have a good performance with respect to one criterion.

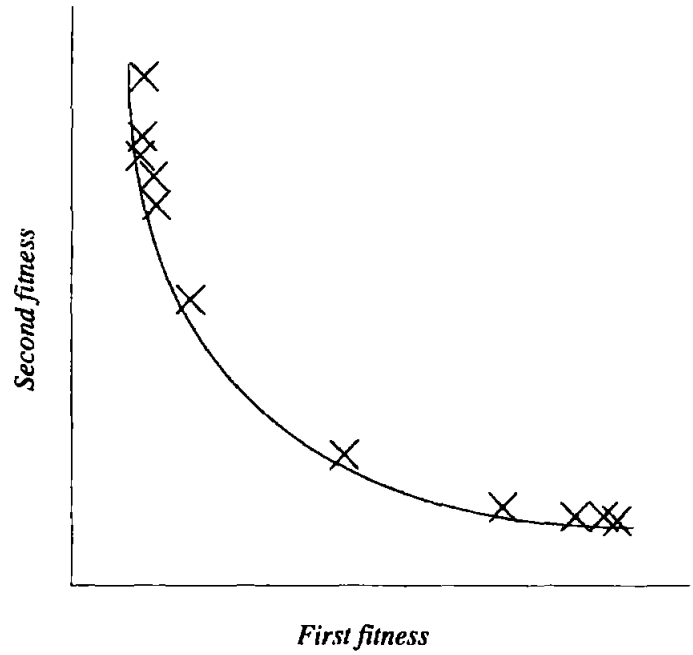
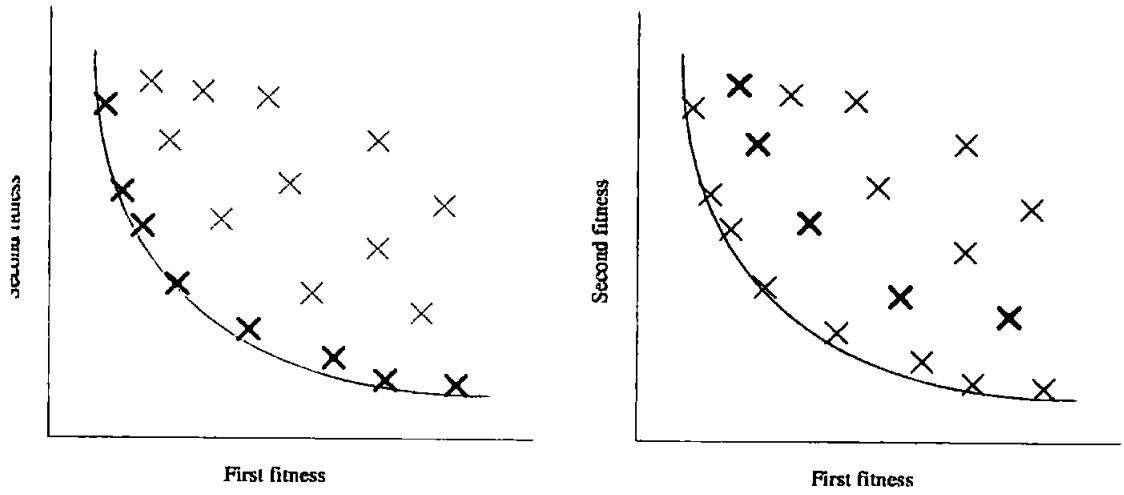


Figure 6.2: Example of a non-dominated set found by VEGA, illustrating the bunching of solutions near the axes.

6.2.4 Goldberg's fitness allocation method

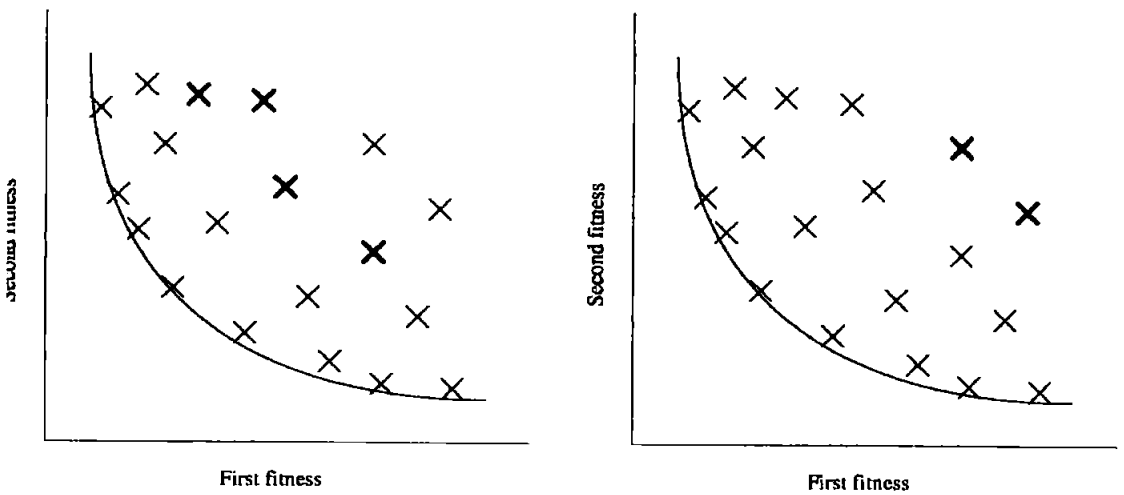
In order to find solutions with a wide range of trade-offs it is necessary to make use of the all of the fitness measures simultaneously. A method described by Goldberg [48] combines this information with the degree of dominance of each member to facilitate a more wide-ranging search.

All of the members of the population have their fitnesses determined under each performance measure, giving a 'fitness vector' which specifies a position in multi-dimensional space. These positions are then examined to determine the NDS within the current population. The members of this set are given a ranking of one, and are then excluded from consideration for the next step, where the NDS of the remaining members of the population is found. These are ranked two, and then excluded themselves. This repeats until all of the population has been ranked. Fitnesses are now allotted by rank, the higher ranks receiving a higher fitness, as illustrated in Figure 6.3. Fonseca and Fleming's analysis of multi-objective natural algorithms [84] has suggested that Pareto-based fitness allocation strategies such as this were the most promising.



(a) First NDS - highest allocated fitness

(b) Second NDS - lower allcated fitness



(c) Third NDS

(d) Last NDS - lowest allocated fitness

Figure 6.3: Example ranking calculation applied to non-dominated sets in the population.

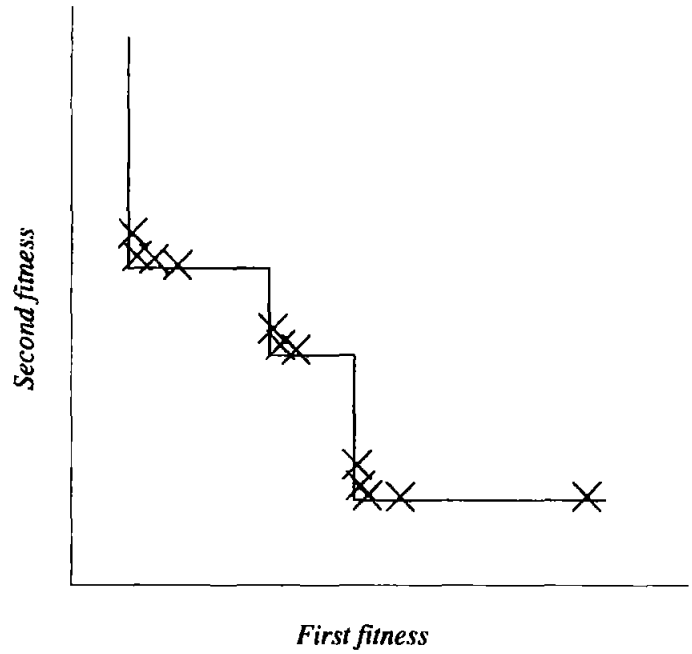


Figure 6.4: Example population distribution between niches.

The fitnesses that the members of the population now have can be used directly in a selection process, but if the first few highly-fit solutions cause premature convergence, by being the only members of the best NDS, then the efficiency of the search process can be compromised. To help combat this, *fitness sharing* is used, whereby the fitness of every member of the population is reduced by a *crowding factor*. This means that the more members of the population are bunched together, the more their fitnesses will be reduced. This has the effect of encouraging the GA to search in the less-densely populated areas, and so the members of the NDS should be spread out more evenly along the 'wavefront' of best solutions and so contain a wider range of solutions. If the POS is 'stepped', as in Figure 6.4, then by encouraging the GA to cover a wider area it becomes easier for the population to find each of the highly-fit regions, which it might not otherwise do due to premature convergence to the first regions it finds. Mahfoud [85] has found that Goldberg's fitness sharing method works well for a wide variety of problems, but may have weaknesses when the search space has many local optima with similar performances to the pareto optima.

The existence of such 'niches' can make a further adaptation of the GA useful. The

chromosomes which exist within each niche will generally be similar to each other, but different to the chromosomes in the other niches. This implies that when crossover occurs between chromosomes from *different* niches, the offspring will tend to be less useful than the offspring of parents from the *same* niche. When the POS is smooth, this technique is still useful since a chromosome which has a good performance with respect to one criterion will probably be quite different to a chromosome with a good performance with respect to another. These effects can be reduced by encouraging crossover between similar pairs of chromosomes. This can be achieved by selecting the first string for crossover, then altering the selection probability of the second according to its relative Hamming distance to the first. This allows the GA to search the niches more effectively for high-fitness solutions.

The benefits of including fitness sharing and crossover restrictions are extremely problem-dependent, and are determined by the shape of the POS, and the variation of the fitness with a change in chromosome with respect to each of the criteria. Similar strategies have been proposed by Horn et al [87] and Srivinas and Deb [88].

6.3 Applications of MCO optimisation to filter design

Standard approaches to MCO filter design generally involve a large amount of constraint-based mathematical analysis. Selesnick's approach [89] requires the solving of a set of non-linear equations, and multiple runs are necessary to obtain a range of solutions with different characteristics. In [27] Lawson designs PCAS filters with approximately linear phase directly, by solving a set of linear equations.

Natural Algorithms have been applied to a variety of MCO filter optimisation problems. Roberts et al [40] and Tang et al [90] use a GA to design FIR and IIR filters respectively, by using a structured chromosome to represent both the filter structure and coefficients. This allows both the frequency response performance and the fil-

ter complexity to be optimised simultaneously. Franzen et al [67] use Evolutionary Strategies and Simulated Annealing, with a weighted sum of performance measures, although this has the disadvantage mentioned above, that manual intervention and repeated runs will be required to obtain a solution with the desired trade-off.

Storn [66] performed a basic form of MCO on IIR filters using Differential Evolution, by giving each solution the poorer of two fitness measures, namely the degree by which the solution violated templates for the frequency response and group delay. This ensured that a filter had to fulfil both criteria to get an optimum fitness. Redmill and Bull [45] have used an MCO GA to produce the pareto-optimal set for low-complexity integer-coefficient FIR filters, thereby optimising both the filter performance and its implementation complexity simultaneously.

6.4 GA difficulty measures and deception

It is often hard to predict whether or not the GA will be a suitable technique to use without trying it in practice [91]. Horn and Goldberg's investigation [92] shows that the number of suboptimal peaks alone does not give an indication of the difficulty of the problem. To combat this, a number of measures have been developed which attempt to give a qualitative, if not quantitative measure of the difficulty of the problem from the perspective of the GA.

6.4.1 Epistasis

In Chapter 3 the schema theory was described, which attempts to explain the underlying mechanism which drives the GA. One of its main assumptions is that it is possible to predict the fitness of a chromosome from just a short section of it. For many problems this is not the case, and the GA is not able to perform well. The degree to which the fitness of a schema is dependent on the values of the other undefined bits (shown by a # in the schemata) is called the *epistasis*. The effect of epistasis is to remove the linear relationship between the gene values and the overall string's fitness. The degree

of epistasis within a string can vary from 0 to 100%.

Davidor [93] has developed a qualitative measure of the epistasis, which is based on the degree to which the actual fitness of strings vary from the fitnesses predicted by their genes. A set of random chromosomes, as large as is possible within storage and/or time constraints, is selected, and their fitnesses found. Each chromosome is then examined, and the average fitness of the strings with bit zero set is found, then the average fitness of those with bit zero clear. The corresponding values for each bit position are also found.

The first of the chromosomes in the set is then examined, and the average fitnesses are summed which correspond to the bit settings at each position. This value is then divided by the string length to give a predicted fitness for the string. The other random strings are also examined in this way, and their predicted fitnesses found. A comparison is then made between the actual and predicted fitnesses, from which a measure of epistasis can be made.

This method has a number of drawbacks, in that the measure of epistasis is qualitative and problem-dependent. It also suffers from a substantial sampling error which becomes increasingly large for small sample sizes. The latter is particularly true for those problems which have a long chromosome, making it impossible to store all the possibilities or to calculate their fitnesses in a reasonable time. However, it does allow a qualitative comparison to be made between the epistasis of different representations of the same problem.

An example of a zero-epistasis problem is that of maximising the numeric value of the binary string. This problem has no gene interaction, because it is possible to say that regardless of the setting of any other bit, if a bit is set then the fitness will always be higher than if it is not. This means that a simple bitwise optimisation can be used to find the optimum. A fully-epistatic problem, in which there is a 100% interdependence of gene values, is a delta function where the fitness is zero unless all the bits are set, when the fitness is one. In this scenario, how a bit setting affects the fitness is totally dependent on the setting of the other bits—if they are all one, then the setting of the

bit will determine the fitness; if they are not, then the setting has no effect at all. For this type of problem, it is generally impossible to predict the chromosome's fitness from a small portion of it, so the GA, and a bitwise optimisation will fail.

The GA is best suited to problems with a reasonable amount of epistasis—too low a value, and the added complexity of the GA is unnecessary, and a simpler optimisation technique is adequate; too high, and the schema theory breaks down and the problem is unsuitable.

6.4.2 Fitness-distance correlation

There are some problems which have a low epistasis, but which the GA still finds hard. Davidor's method above would predict these problems to be suitable for the GA, implying that a better method of calculating the GA-difficulty of a problem is needed.

Jones and Forrest [94] have proposed a different measure, the *Fitness-distance correlation* (FDC), which analyses the *deceptiveness* of the problem. A search space is deceptive in GA terminology if the search space tends to lead the GA away from the optimum, which can happen if the optimum is small and lies within an otherwise low-fitness region.

The FDC is calculated by finding the degree of correlation between a string's fitness and the distance to a global optimum. For a simple unimodal space, there will be a monotonic decrease in fitness with Hamming distance from the optimum, so the correlation will be high. As the search space becomes more complex and hilly, there is less useful information to tell the GA which direction the optimum lies in, so the correlation will be lower, and the problem will be harder for the GA. For a deceptive problem, where the fitness increases as the Hamming distance from the optimum increases, the FDC will indicate a correlation of opposite sign to the unimodal function, meaning that the GA will be led away from the global optimum.

The signs of the FDC values are dependent on whether the GA is being used as a

maximisation or minimisation tool. When minimising a problem, the sign of the FDC should be positive for a suitable problem, since the fitness should increase with distance, and negative for a deceptive problem. When analysing a maximisation problem the signs will be reversed.

The FDC is more reliable than the epistasis measure given above, but it does rely on the user knowing the location of the global optima in the search space. In the absence of this knowledge, the space around local optima can be analysed, but it cannot be assumed that the results can be extrapolated to the global situation.

6.5 Alterations to the GA

In order to extend the GA as planned, to encompass the optimisation of more than one design criterion simultaneously, the GA was extended to include MCO, in the form proposed by Goldberg (Section 6.2.4). The interface was extended to allow the display of non-dominated sets, with the the filters described within it. The limit of 640K of memory imposed by the DOS operating system meant that the number of non-dominated individuals which could be stored within the NDS was restricted. This reduced the effectiveness of the search considerably. To combat this, in the final phase of the work described in Chapter 8, the Borland C++ 3.1 compiler was dropped in favour of the GNU freeware DJGPP compiler for DOS. This compiler has flat 32-bit memory addressing, allowing the full memory of the computer to be used. This meant that the maximum size of the NDS could be greatly increased, and increase the GA's potential.

Chapter 7

An Analysis of the Suitability of GA-based Optimisation for Non-linear Phase FIR Filter Design

7.1 Introduction

In Chapter 4 the design of linear-phase FIR filters by GA was discussed. These filters' linear phase makes them particularly useful in areas such as biomedicine and audio where low phase distortion is of paramount importance. They do however suffer from having a long delay of half the filter length, making them unsuitable for high-speed, real-time applications. If the restriction on linear phase is relaxed outside the passband, then a shorter filter could be designed with the same magnitude response but a lower delay, as shown by Selesnick and Burrus [89], who used standard methods to produce a reduced delay, but with the restriction of having a maximally-flat magnitude response. Some techniques requiring or generating minimum-phase filters exist, approximating an FIR filter with a much shorter IIR filter [95], but this gives no control over the linearity of the phase response.

It was now planned to investigate another area in the design of FIR filters by extending the GA into a true MCO tool, and designing the filters under two criteria

simultaneously. This would not only simplify the design process by combining a number of design steps, but also reduces the degree of user intervention by reducing the number of iterations needed to produce a desired filter. A further analysis step, that of the effect of coefficient roundoff on the filter's response was catered for by using a binary rather than a real-coded chromosome. This approach meant that the coefficients always have quantised values when they are decoded by the fitness function, so the filter's responses intrinsically include the effects of their being represented with finite precision. The use of an MCO GA meant that the designer could be presented with a range of non-dominated solutions from which the most suitable can be selected, rather than having to undertake an iterative adjustment of weightings (Chapter 6) to obtain a suitable solution.

7.2 Non-linear phase FIR filters

It is a necessary and sufficient condition for a filter to have a linear phase response for its impulse response to be symmetric or antisymmetric, although this requirement ties up a large number of degrees of freedom by constraining the coefficients. In order for the filter to achieve a high stopband attenuation or a sharp cut-off, the filter must have a large number of coefficients. Since the FIR filter's group delay is given by half of the filter length, it means that the delay of high-attenuation or sharp cut-off filters is also high. This makes them unsuitable for high-speed, real-time applications where high-speed devices are unavailable, and also means that a filter must be very long in order to achieve that same magnitude response as a filter with complete freedom in the phase domain. Linear phase filters have the advantages that their symmetry means that they only require half the coefficient storage of a non-linear filter, and they can therefore be implemented more efficiently. Their constant group delay means that the signal's components are delayed by an amount proportional to their frequencies, so it is not distorted. This property is particularly important in audio, data transmission and biomedicine, which are especially sensitive to distortion.

In order to reduce the length of the filter, and so also reduce the delay, the restriction on linear phase can be removed. This can allow the design of a filter with the same magnitude response, with as few as half the original number of coefficients for wide passband filters [11]. The class of filter, known as the *minimum-phase*, has the shortest possible delay for a given magnitude response. Although these filters are suitable for phase-insensitive applications, they cannot be used in other situations, so a compromise must be reached between phase-linearity and filter length. One way to do this is to force the phase response of a non-linear filter to be as close to linearity as possible in the passband, where it is important that the signal should not be distorted. In the stopband, the phase response can be left unrestricted, because the signal is attenuated, so any distortions caused by nonlinearities are unimportant and can be discounted provided the attenuation is high enough. As the linear-phase restriction has been removed, the constraint on the coefficients being symmetric no longer applies and they can describe a wider range of filters, and the filter length can be reduced while maintaining the magnitude response.

In order to design such a filter, the simple optimisation approach used in Chapter 4 is no longer suitable, as there are two performance measures which need to be examined: the performance of the magnitude response with respect to the desired response template, and the phase response with respect to linearity over a chosen region of the response. Since these have different ranges of unknown magnitudes, several iterations would be required to find the correct weights to use in a weighted-sum fitness function in a standard GA. A better approach is to use one of the multi-criterion optimisation (MCO) approaches detailed in the previous chapter. These allow the GA to perform the trade-off automatically without user intervention, producing a range of solutions from which the most applicable may be selected.

A major objective of this work is the simplification of the filter design process, so to this end a multi-objective Genetic Algorithm was developed to optimise the filter coefficients with respect to both the filter's magnitude response and its phase response in a region of the passband. To increase the number of filter design steps being under-

taken simultaneously, the Genetic Algorithm used a binary chromosome, containing a concatenated list of the coefficient values, which therefore intrinsically accounted for coefficient quantisation effects. Since the impulse response has the same values as the filter coefficients, the quantised coefficient values decoded from the chromosome can be used to find the responses by zero-padding them to a length of 1024, and taking the FFT.

The fitness function with respect to the magnitude response was the maximum error from a desired response template (such as that shown in Figure 7.1), while that for the phase response was the sum of the squared differences between the response and an LMS straight line fitted through the response, in a selected region covering most of the passband, thereby giving a measure of its linearity.

7.2.1 Effects of coefficient quantisation

A further important factor which should be taken into consideration is that of coefficient quantisation. In a practical application, the coefficients will be stored in a quantised form, which means that they will only be able to take a certain number of values, and that consequently there are only a finite number of possible filters for any given order. Standard methods of FIR filter design rely on full-precision maths, e.g. the hill-climber which requires a continuous surface to perform effectively with small-scale movements, such as those that occur when the search has converged on a peak. When the coefficients are stored in a quantised form, small ranges of their continuous values are stored with the same quantised value. This results in small areas of the search space, with dimensions of the quantisation interval, having the same parameter values, and so the same fitnesses. This is illustrated simply in Figure 7.2.

When a hill-climbing search such as the Simplex method (described in Section 4.4) is initialised, it looks and moves around the search space in large steps. Since it only takes point samples, the stepped nature of the search space is masked and the optimisation can proceed effectively to the higher-fitness regions of the space. When

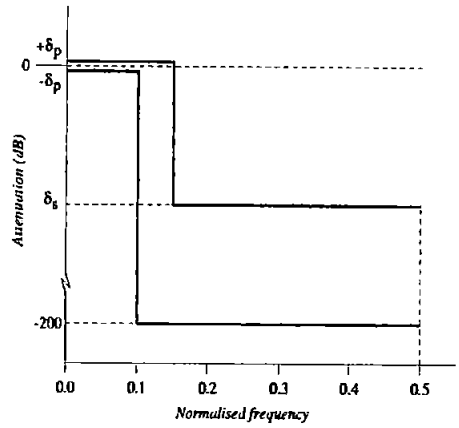


Figure 7.1: Desired response template used to calculate the fitness with respect to the magnitude response.

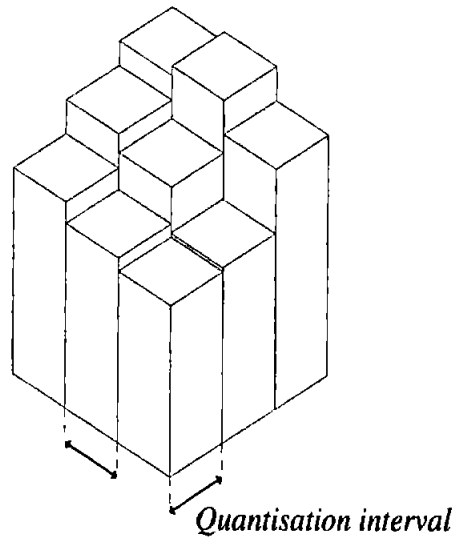


Figure 7.2: Illustration of the effect of coefficient quantisation on the search space.

the Simplex begins to converge, however, it shrinks, and takes smaller and smaller steps. This means that it only looks at the immediate vicinity of its current position, and the flat regions are increasingly apparent. The final optimisation by the Simplex method relies on being able to take ever-smaller steps, which is not possible with a quantised space, and the search also fails if two of the points in the Simplex occupy the same position in space. This is very unlikely to happen in a full-precision system, but is bound to eventually in a quantised system when the step size drops to the order of the quantisation interval.

To overcome these difficulties, a bitwise form of hill-climbing must be used. The method chosen was to flip each bit in the string in a random order, calculating the new fitness each time. If flipping the bit improved the fitness with respect to at least one criterion, then the change was kept, otherwise the original bit was restored. This was applied to the members of the NDS in the current population only, once every 50 or 100 generations.

7.3 Use of the GA

An MCO GA was set up to optimise the quantised coefficients of non-linear FIR filters. A binary approach was used so that the effects of coefficient quantisation on the filter performance were accounted for intrinsically. The fitness function calculated two performance measures, firstly how much the magnitude response deviated from a desired template, and secondly the mean-squared error from linearity of the phase response over a specified region covering most of the passband. This approach was intended to allow the GA to trade-off the performance of the quantised coefficient filters, and to return a number of solutions to the designer, thereby performing the two optimisations simultaneously while intrinsically taking account of coefficient quantisation. This combines a number of the traditional design steps into a single operation, while retaining the freedom of the designer to select the most applicable design for their application.

7.3.1 Design performance

Initial runs were performed with randomly-initialised chromosomes, but proved unable to find solutions which fitted the design templates satisfactorily, or even to within a magnitude response error of 10–20dB, although the optimisation of the phase-linearity was generally more successful. The magnitude response fitness function minimised the maximum deviation from the supplied template; all solutions which fell fully within the template boundaries were given the same fitness as all were taken to be equally suitable.

Since reducing the minimax error is a hard problem, the fitness function was changed to optimise the RMS error in the deviation of the magnitude response from the template. Using the RMS error is more forgiving of outliers and 'rogue' points, and so should result in an easier optimisation for the GA. This proved more successful in that the GA was able to reduce the RMS error more than when using the minimax error, although as it then resulted in a *greater* minimax error in the magnitude response, the technique was still not satisfactory.

To determine if the problem is simply too difficult for the GA to solve from randomly-initialised positions, solutions close to a known good solution were used to 'seed' the population. If the GA is able to perform the optimisation, it should then be able to improve on those solutions. The first population was therefore filled with perturbed copies of a Remez exchange solution, where a chosen percentage of the bits in the good chromosome were flipped. The first trials added a single unperturbed copy of the good solution to the population, but this caused premature convergence as the seed was a much better solution than the randomly-chosen ones and therefore quickly overran and dominated the population.

To combat this problem, the population was seeded entirely with perturbed copies of the Remez exchange solution. This allowed the GA to perform more effectively, but it was never able to find a solution even as good as the original quantised Remez exchange design. It had been expected to improve on this design as the linear phase constraint

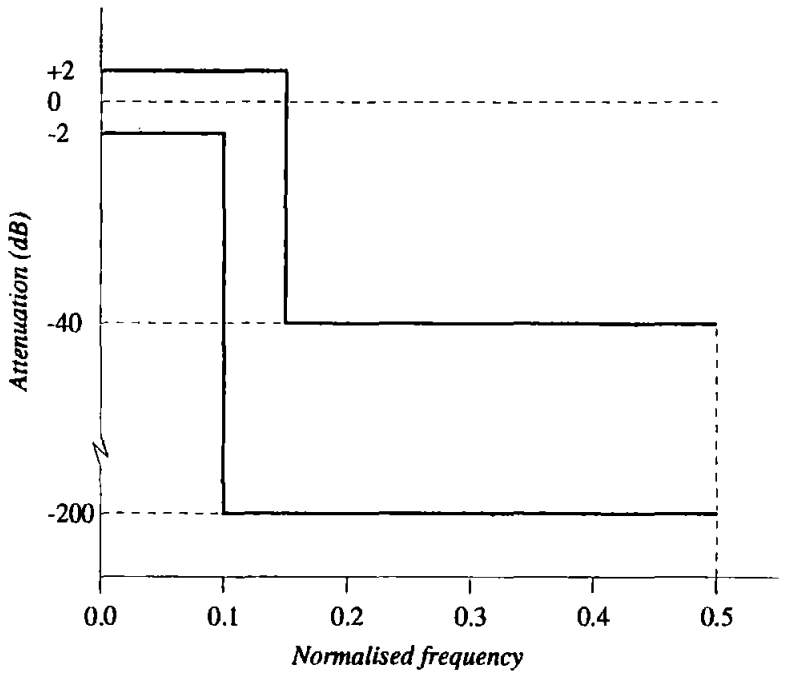


Figure 7.3: Loose-tolerance template used to test the non-linear FIR design technique.

had been relaxed without reducing the order, and by quantising the coefficients, they are moved from their original positions and are no longer optimal.

Since the Genetic Algorithm had not performed well, a loose tolerance filter template as shown in Figure 7.3 was used as a test problem. The order (as determined by Matlab) was set to 25, and the wordlength to eight. As the order was determined for a linear phase filter, and the filter being designed was not linear phase over the whole response, it was anticipated that the Genetic Algorithm should be able to find an acceptable solution to this problem, as there were fewer constraints on the coefficient values. However, even for this simple problem, the Genetic Algorithm was only able to improve a little on the initial best fitnesses.

A typical best magnitude-response solution found by the GA is given in Figure 7.4 for a lowpass, $N = 40$, Type I filter with band edges at 0.1 and 0.175. The filter is shown with the nearest equivalent Frequency Sampling FIR filter found by hybrid GA. The non-linear phase GA was run with a population of 100, crossover and mutation probabilities of 0.6 and 0.005 respectively, for 8,000 generations. The FS filter was found in under 30 seconds, while the non-linear phase GA ran for almost two hours.

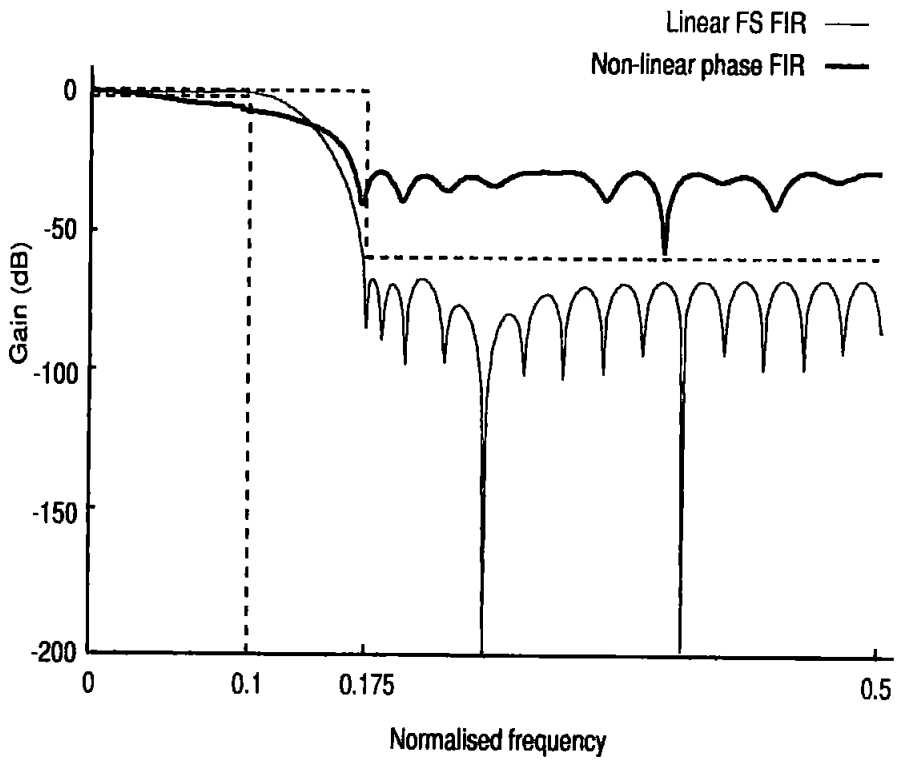


Figure 7.4: A typical non-linear phase FIR filter compared to the nearest equivalent Frequency Sampling linear phase filter, both found by GA.

The filter shown had the best magnitude response, with a maximum error from the template of 32.569dB, and a sum-of-squares error from phase linearity between 0–0.09 of 0.0082. The filter with the worst magnitude response error (59.908dB) had a phase response with a sum-of-squares error from linearity equal to zero to six decimal places. The template was ±0.25dB over the passband, +0.25–-200dB over the transition band, and -60–-200dB in the stopband. The filters' characteristics are compared in Table 7.1.

At this point, investigations were undertaken to determine why the Genetic Algorithm was not performing well, and to analyse the difficulty of the problem. The analyses, which were performed for the filter design template in Figure 7.3, will now be detailed.

Value	Linear phase FIR	Non-linear phase FIR
Wordlength	full-precision	16
Generations	1,000	8,000
Time	<30 secs	~2 hours
Best attenuation	66.567dB	27.431dB
χ^2 error in linearity	0	0.0082

Table 7.1: Comparison between a linear-phase FIR filter found by the technique described in Chapter 4 and the best magnitude response filter found by MCO GA.

7.4 Analysis of non-linear FIR filter design

7.4.1 The parameter space

The parameter space and the search space are related but different ways of looking at the problem space. The search space is the space as seen by the GA, which in this case will be examined by altering chromosome bits and finding the fitness of each solution. Although this gives the truest picture of what the GA 'sees', it is hard to visualise or display. The parameter space, on the other hand, changes the viewpoint on the problem by altering the model parameters directly, so they can take any values. This makes it easier to plot slices through the space by fixing all but two parameters and altering the others to give a 3-D search surface of the fitnesses. This surface is clearly different to the search space, but can be made most similar to it by plotting it with a number of divisions equal to the number of divisions in the binary-representation in the chromosome.

It is possible to build up a picture of the nature of the parameter space by taking a number of slices through the space by fixing all but selected pairs of parameters, and varying those over their allowed range in a finite number of steps. By selecting pairs which are held at different locations in the chromosome, the effects of short- and long-range interactions can be discovered. The fitness function used was the magnitude response error, as this was the fitness measure that the GA had had the most problem with. A lower fitness value therefore means a better filter. A good solution was

required, so in the absence of one found by the GA, coefficients calculated by the Remez exchange method were used to seed a bit-flipping local search algorithm, and the solution this found was assumed to be a near-global optimum.

To draw each slice, all but two of the filter coefficients were fixed, and a surface plotted from the fitnesses found by varying the others. For the 25^{th} order filter, the slices were produced by varying the following pairs of coefficients: 1 and 2; 1 and 13; 1 and 24; 12 and 13. The rest of the coefficients were fixed, firstly at the values found by the local search optimisation; secondly at the local search values perturbed by 5% of their value, and finally at random values. For simplicity these data sets will be referred to as LS (local search), P (perturbed local search) and R (random), for example 1,24,P and 12,13,R. It should be noted that the changes in the appearance of the parameter space described below were similar for all the data sets. It should be noted that these surfaces represent the maximum error in decibels between the filter and the design template, so a smaller error means a better solution.

Figure 7.5 shows the 1,13,LS data slice. It shows that the parameter space consists of smooth intermediate-error regions separated by narrow high-error ridges. The low-error region around the optimum (which lies in the depression at the front of the figure) is clearly small compared to the total area of the slice. When this is extrapolated to the full 26-dimensional volume, the proportion of the total volume with a similarly high fitness will be extremely small. The other unperturbed local search slices have a similar appearance but with different numbers and orientations of the ridges, which are mostly straight like the 1,13,LS slice, while for the 1,24,LS slice, some of them were curved. The optimum region in all the slices is very small, suggesting that perturbing any coefficient will quickly lead to a large drop in fitness and therefore performance.

In Figure 7.6, the same slice is shown, but with the fixed coefficients perturbed by 5% from their original, near-optimal values (the 1,13,P slice). The surface has become more multi-modal, with a greater number of low-error regions than before. The characteristic appearance of the slice is, however, the same. The small low-error region around the local search solution has disappeared, leaving a comparatively poor

best solution. The position of this best remaining solution moves in all the slices examined, sometimes considerably, so seeding the Genetic Algorithm with a perturbed solution will not generally start it off in the region of the true optimum.

The same slice was also obtained for a filter with the fixed coefficients selected at random. As can be seen in Figure 7.7, the space is very flat, with very poor filter performances, and no indication as to the location of the original optimum solution. This has the implication that a GA seeded at random will contain no useful information as to the best regions to search in, so will be unable to proceed effectively.

7.5 Measures of GA-difficulty

As with all design techniques, the GA has its advantages and disadvantages which make some problems easier and some harder for the technique to solve. For the GA, although it has several advantages over 'standard' optimisation methods, its success is highly dependent on the structure of the chromosome, the search space, and how they interact, which in turn depends on both the problem itself and the way it is stated.

The *deceptiveness* of a problem in a GA-sense can have a number of causes, but its effect is to tend to lead the GA *away* from the global optimum. This can occur if highly fit, short building-blocks combine to give longer blocks with a *lower* fitness. If this occurs, it means that although selection may pick those strings containing high-fitness building blocks, when crossover combines them they produce poorer offspring, and one of the basic premises behind the GA breaks down. Other types of deception can occur when the structure of the search space 'misdirects' the search. For example, if the global optimum is small in area and is surrounded by the worst points in the space, then in almost all areas of the space the direction towards the local optima will be different to the direction of the global optimum. This is illustrated in Figure 7.8, where it can be seen that apart from in the narrow shaded region, the direction towards the nearest optimum leads away from the global one, and the problem is deceptive. If the deception occurs in the search space, as in this example, then the search will be difficult for any

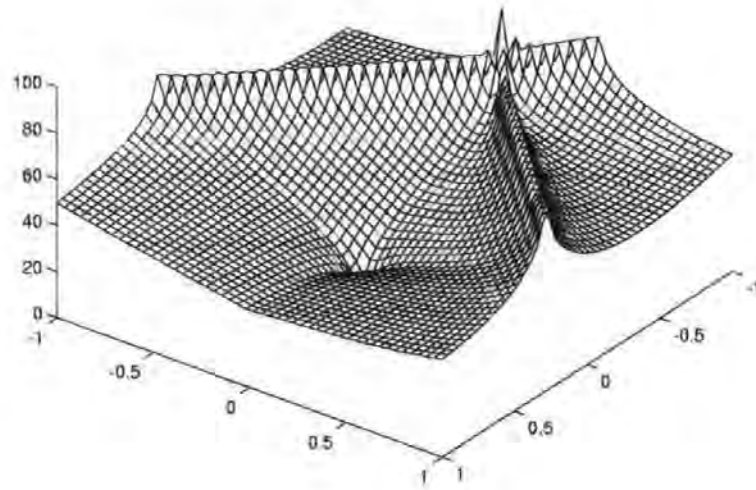


Figure 7.5: Slice through the 1,13,LS data set, with a best fitness of 11.3dB.

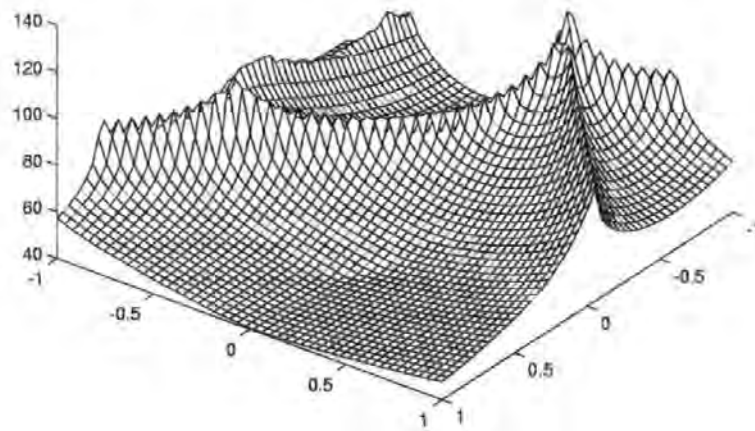


Figure 7.6: Slice through the 1,13,P data set, which has has a best fitness of 40.5dB.

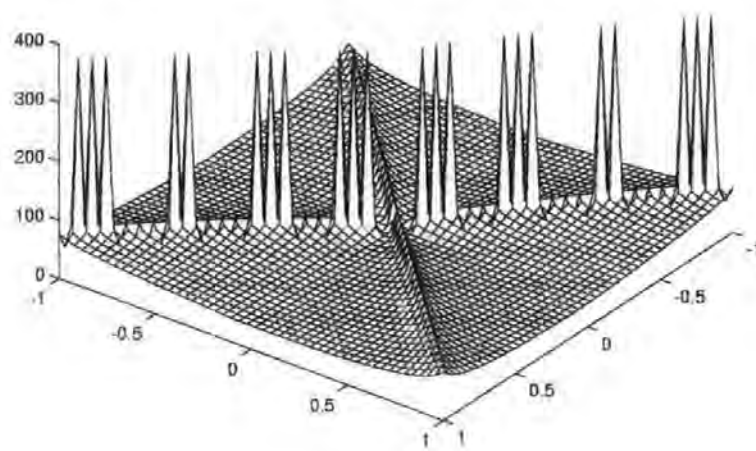


Figure 7.7: Slice through the 1,13,R data set, with a best fitness of 32.2dB.

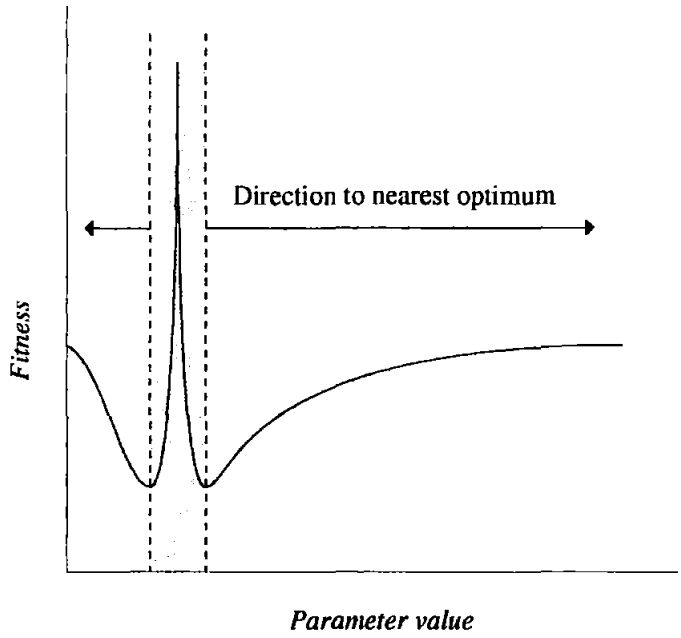


Figure 7.8: Illustration of deception, where only points in the shaded area indicate the location of the optimum.

technique which uses hill-climbing or gradient information or which performs a random search, while the deception in the chromosome is clearly a GA-specific problem. The level of chromosome-based deception can be reduced by changing to a higher-cardinality alphabet, but this is not always possible or desirable, for example if the problem fits naturally into a binary representation.

7.5.1 Epistasis

One of the causes of chromosome-based deception is a breakdown of the linear relationship between gene values and the fitness of the solution they represent. It should be possible to predict the fitness of a chromosome from a short portion of it, but this breaks down if solutions containing good genes combine to give a poorer solution. When this occurs, it implies that there is a great degree of interaction between different genes' values and the fitness of the chromosome as a whole. The degree to which the fitness contributed by a particular gene is dependent on the values of the other genes in the chromosome is called *epistasis*.

It was now proposed to investigate the epistasis of the non-linear FIR problem, using

	Random	10% perturbed LS
σ_ϵ^2	116.75	363.19
σ_v^2	132.25	112.26
σ_A^2	17.73	631.57
$\sigma_v^2 - \sigma_A^2$	114.52	-519.31

Table 7.2: The Epistasis variance (σ_ϵ^2), calculated for regions of the search space.

the qualitative *epistasis variance* measure of Davidor [93]. These epistasis measurements were performed on two 5,000-point data sets, one of randomly-selected points, and the other for the local search solution perturbed by a random amount of up to 10%. The results are tabulated in Table 7.2.

The fitness variance σ_v^2 is a measure of the spread of the sample fitnesses around the average sample fitness, while the genic variance σ_A^2 is a measure of the spread of the predicted fitnesses around the true fitness sample average. It can be seen that the genic variance is very low for the random data, which can be explained by the relative flatness of the parameter space as shown above in Figure 7.5. This flatness means that regardless of the gene values, the fitness will not change greatly, so it can be predicted with reasonable accuracy.

On the other hand, the genic variance of the predicted fitnesses is comparatively high for the perturbed local search samples. As can be seen from the example parameter space surface in Figure 7.6, the space is highly variable, and there is a far greater correlation between *both* genes' values and the overall fitness of the solution. This need for both genes to have the correct values to produce a high-fitness solution means that there is a high degree of epistasis in the problem. Another important result from these calculations is that the problem becomes more epistatic as the optimum is approached, which implies that it also becomes harder for the Genetic Algorithm, due to the effects described above.

In the context of the non-linear FIR filter design problem, this implies that to

obtain a good solution it is necessary for the coefficients to *all* have the correct values, and that 'partial' good solutions do not exist in that it is impossible to say if half a chromosome gives a good or bad solution, since the quality of solution is also dependent on the values of the genes in the rest of the chromosome.

The impulse response of an FIR filter (which has the same values as the coefficients) has to have a specific shape to make the filter operate in the desired way, and although this has different characteristics in different types of filter, e.g. symmetry for a linear phase, or with its peak amplitude early on for a minimum-phase filter, the underlying appearance is similar, with well-defined ripples, whose amplitude rises and falls within a smooth envelope. It is clear therefore that although there may be a large number of sets of coefficients which give the same magnitude response, within each solution, the value of each coefficient is highly dependent on the value of the others to produce the desired response. Should mutation or crossover alter the value of a single coefficient in a good solution, then all of the other coefficients need to change to preserve the quality of the filter. This means that crossover of two good but dissimilar chromosomes will lead to poor offspring. It is also clearly impossible to predict the response of a filter from just a few coefficients, so there is a large non-linearity between gene values and the corresponding fitnesses, and the problem is not suited to solution by the GA.

7.5.2 Fitness-distance correlation

A better measure of GA-difficulty is claimed to be the fitness-distance correlation [94]. The FDC examines the difficulty of the solution in terms of the variability of the search space, and its degree of deceptiveness. The calculations were performed for three data sets of 4,000 points, and averaged over ten runs. Two of the sets were made by perturbing 1% and 5% of the bits in the assumed optimum solution used in the epistasis calculations above, and one set contained randomly-generated points. The results are shown in Table 7.3.

One random data set is shown in Figure 7.9, which plots the magnitude response

	FDC
Random	0.0032
5% perturbed LS	0.3199
1% perturbed LS	0.6466

Table 7.3: FDC calculations for various regions of the search space.

error against the Hamming distance between each random solution and the optimum one. The Hamming distances have been perturbed slightly to show the distribution of points more clearly. The FDC for the random data set was almost exactly zero, bearing out the distribution of points in the figure, which contains no clear correlation between fitness and distance, and therefore usually no indication of the direction of the optimum from a randomly-selected point.

When the same calculations were repeated for data sets taken around the assumed optimum, obtained by perturbing an average of 5% and then just 1% of the bits in the assumed optimum solution. In these regions, the FDC was calculated to be 0.32 and 0.65 respectively. A typical set with 5% perturbation is shown in Figure 7.10, which clearly has more structure, in that the points are constrained in a narrower band. This means that the search space closer to the optimum behaves in a more predictable manner, so it should be easier for the GA to optimise the solution. However, the spread of fitnesses is still large and the optimisation is unlikely to succeed.

For the 1% set, as shown in Figure 7.11, there is still more structure, and the fitnesses are constrained to an even narrower region than before. For the solutions with a Hamming distance of just one bit, there are clear bands indicating that only a few different fitnesses can be generated. When the number of bits difference rises, more values can be obtained, affecting more genes, so the number of possible fitnesses rises and the bands overlap and can no longer be resolved. The range of possible fitnesses still rises very quickly, showing that changing even a few bits can have a profound effect on the quality of the solution. The poorer solutions will be caused by a perturbation

of the bits in the chromosome which represent the most significant bits of coefficient values as these will cause the greatest perturbation in the solution. When changing one of the least significant bits the change will be much smaller, and possibly insignificant. This implies that changing even a few bits *can* (but is not certain to) lead to a dramatic decrease in performance. The most dramatic example of this is the point indicated in the figure, which has only two bits difference in its chromosome, but a magnitude response error of over 130dB.

7.6 Results

The slices taken through the parameter space show that it is multi-modal and therefore also unsuitable for hill-climbing methods, while the small size of the region around the optimum means that a random search will take an excessive amount of time to find a good filter. The GA, although appearing to be a suitable optimisation technique for this type of space, was not found to be a suitable technique for directly optimising the coefficients of non-linear FIR filters.

GA theory suggests that, for success, there should not be an excessive interdependence between the gene values to disrupt the linear relationship between gene value and fitness. The FDC calculations suggest that not only does the search space become more suited to the GA as the optimum is approached, but it is *only* suitable around these regions. This means that at the start of a run when the population contains widely spaced points, it is unlikely to contain information about the location of the optimum. As the optimum is approached, the FDC suggests that the space becomes more ideal, although the epistasis increases as the fitness becomes ever more sensitive to changes in coefficient values and the non-linearity of the gene-fitness correspondence rises.

One cause of failure is the initialisation of the population, since it has been shown that the space far from the optimum contains no useful information. Since non-linear filters have non-symmetric impulse responses and hence coefficients, initialising them

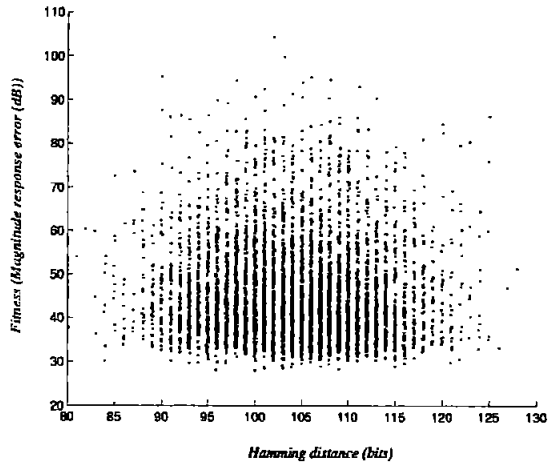


Figure 7.9: Scatter plot of fitness against Hamming distance for a random set of non-linear FIR filter chromosomes.

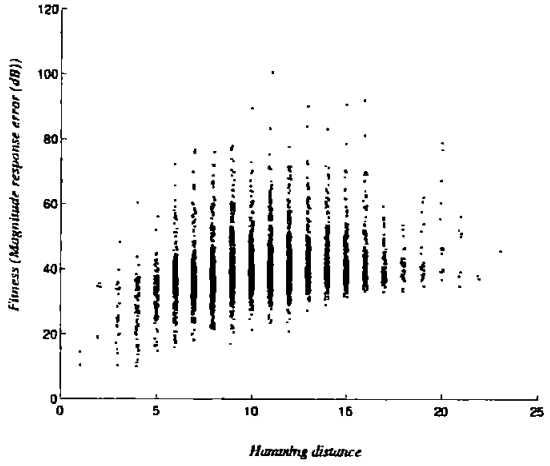


Figure 7.10: Scatter plot of fitness against Hamming distance for a 5% perturbation, non-linear FIR filter data set.

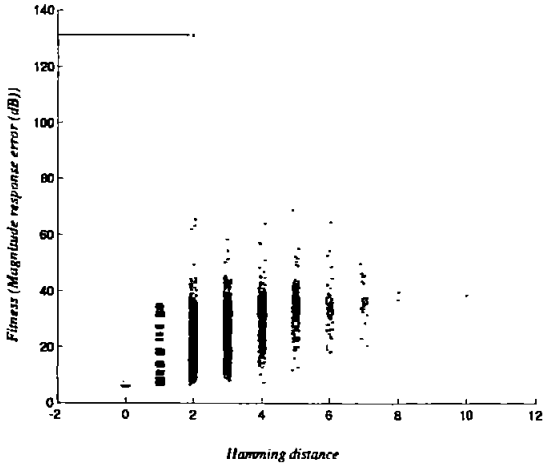


Figure 7.11: Scatter plot of fitness against Hamming distance for a 1% perturbation, non-linear FIR filter data set.

with linear-phase solutions, even perturbed solutions, may be far from ideal. Minimum-phase filters have the peak amplitude of their impulse responses early on, so initialising the population with perturbed linear phase solutions which have been shifted or 'rotated' might be a better technique. The success of the GA with this problem is still not guaranteed however, due to the high epistasis of the search space and its unsuitable structure away from the optima.

The optimisation of the phase was more successful as the GA was able to move from its initial perturbed linear-phase solutions back towards fully-linear phase filters as the optimisation required is far simpler. The requirements for linear phase are that the filters are symmetric or antisymmetric, but otherwise places no restrictions on the coefficients' values. This means that there are many solutions with linear or near-linear phase, which are therefore easier to find, but to produce a filter with a desired magnitude response requires that *all* of the coefficients have good values collectively, so there are comparatively few good solutions in the phase domain.

It might be possible to reduce the degree of epistasis by re-stating the problem so that the chromosome stores information about the filter coefficients in a different way, or stores entirely different information which is can be used to adjust the filter's responses. Methods for reducing epistasis such as the 'Expansive Coding' at of Beasley et al [96] will not work in this case because it is not possible to break down the problem into suitable sub-problems—all of the coefficients must be used at once to determine the filter's fitness.

Since the completion of these investigations, Lu and Tzeng [97, 98] have successfully used a GA to produce non-linear optimal filters. Their approach is to use a guided crossover technique which biases poorer solutions towards better ones, and a fitness function which uses a least-squared error measure, which is an easier problem than the minimax one attempted here. They achieve an Optimal (equiripple) response in both phase and magnitude responses, by using a dynamic weighting function, which is updated to give a stronger weighting to those areas of the space which have the greatest error. This allows the GA to concentrate its search on those areas which most

need improvement. No attempt was made to reproduce linear-phase filters with fewer coefficients, but it is likely that a modified version of this technique, which ignored the phase response in the stopband, would be successful at producing lower-delay, near-linear optimal FIR filters.

Chapter 8

An Extended Multi-objective GA for IIR Filter Design

8.1 Introduction

In Chapter 5, a GA was used to design an IIR filter using a chromosome which optimised the filter coefficients directly. This was found to have a limited success, due to strong coefficient interdependence making the problem less suitable for a GA-based optimisation. In the light of the analysis presented in the previous chapter, it was decided to repeat the investigation using a true MCO GA. This would optimise the pole-zero positions instead of the a and b coefficients, as perturbing a pole or zero slightly has a more predictable effect on the overall response than altering a coefficient. A suitable GA could search for an NDS containing a range of solutions with different performance tradeoffs, from which the most appropriate can be selected.

To enable the GA to cover more filter design steps, the fitness function can be extended to examine how close the phase response is to linearity over a region of interest within the passband. The noise performance of the filter can also be calculated, which will allow the GA to look for the best pole-zero pairing and ordering, while a binary-coded chromosome can allow the effects of coefficient quantisation to be included implicitly in these calculations. Lastly, additional bits could be included to specify the

type of filter section to use for each stage, which would give the GA more freedom to pick lower-noise structures, or even to allow it to pick between a parallel or cascade structure.

The intention is that it should ultimately be possible, under such a scheme, to design filters with quantised coefficients, with the GA returning a range of non-dominated solutions which trade-off the performance between magnitude and phase responses and roundoff noise gain, while using the filter structure giving the best performance, and with the best pole-zero pairing and ordering. It was planned to extend the fitness function in stages to incorporate these optimisations.

8.2 Chromosome design

In the light of the inability of the GA to optimise the non-linear FIR filter coefficients, it was clear that to successfully perform this multi-criterion optimisation, the chromosome would have to be designed to have a lower epistasis. This in turn implies that the problem would have to stated in such a way that a suitable chromosome would be used.

The original IIR optimisations were performed on the coefficients of canonic sections, cascaded to give an overall filter response. This approach suffers because the coefficients are used in a polynomial expression, and these are very sensitive to changes in coefficient values. It is also hard to predict changes in the response of each section as each polynomial coefficient is changed, since small changes in the coefficients can move a filter from being stable to unstable, or from having real to complex poles, as is shown in the stability triangle in Figure 8.1. This clearly shows that if b_2 is zero, then a change in b_1 from 0.99 to 1.01 would result in an unstable filter.

The GA used to optimise IIR filters in Chapter 5 was constructed in such a way that the value of b_1 was constrained to lie in its valid range of -2–2, and that of b_2 to within the range -1–1. Since it was still possible for the solution to lie outside the stability triangle when both values were taken together, every solution had to be checked for

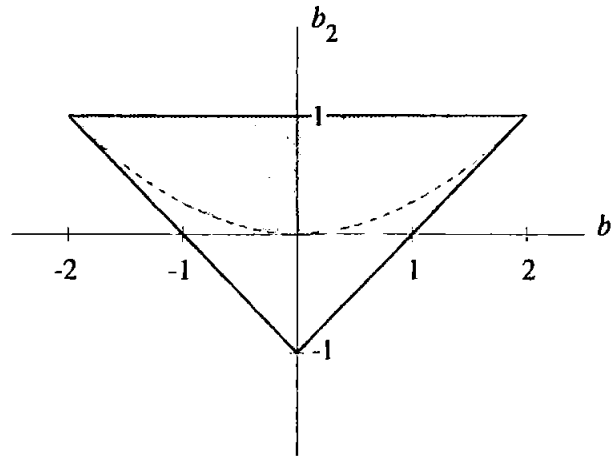


Figure 8.1: Triangle bounding the stable values of b_1 and b_2 for a second-order IIR section. The shaded area contains complex-valued poles.

stability and moved back into the stable region if it lay outside it.

This approach is not ideal as it changes the information contained within the chromosome, and so will affect the data which the GA is accumulating within the population. Although the information is being moved to a valid solution, it also means that the stability of *every* second-order section must be examined at *every* fitness calculation, because of the damaging effects of crossover and mutation. A better approach would be to use a representation which *only* returns valid solutions, and so never requires checking as crossover and mutation cannot result in an unstable solution.

A suitable representation with these characteristics is to use the radius and angle of each pole and zero. Assuming that the poles and zeros exist as complex conjugate pairs, four values are required to describe each second-order section, namely the radius and angle of the positive-angled pole and zero of each pair, although this was simplified to three in this work by fixing the zero radius at unity. This representation contains intuitively short building-blocks, describing the position of a single pole or zero. It is also clear that since each pole and zero has a known effect on the response as a whole, changing the values of a pole or zero's radius or angle changes the response in a more predictable way, never producing an invalid solution.

While the GA still needs to find a complete good chromosome to produce a good filter, as the response depends on all of the coefficients, moving a pole or zero slightly

has a more controlled effect on at least some aspects of the filter performance, such as the magnitude response, than moving an a or b coefficient, so this representation should be more suitable for the GA to optimise.

To specify the pole and zero, their positions could be specified by either Cartesian (real and imaginary) or polar (radius and angle) coordinates. If the chromosome used to store the values uses a binary alphabet to intrinsically account for coefficient quantisation effects, then not only will changing gene values affect the response in different ways, but a different set of possible positions can be represented. This is illustrated in Figures 8.2 and 8.3, for the two representations at the same resolution. It can be seen that although the points specified by using the Cartesian coordinate system are spread more evenly over the unit circle, there will be some points which lie outside the unit circle and which therefore specify illegal positions for poles and must be checked for. The Cartesian approach also has the disadvantage that there are relatively few available positions close to or on the unit circle. This means that the zeros, which are generally found *on* the unit circle, can only lie close to their optimum positions, while for the radius/angle representation, the full radius is available, albeit at fixed angular positions. Other representations and topologies will of course have other distributions [99, 39].

The proportion of the Cartesian positions which lie outside the unit circle is of the order of $1 - \pi/4 = 0.215$, the exact value being dependent on the quantisation interval. This means that around one-fifth of the positions the chromosome can represent are illegal and cannot be used, so the full range of solutions described by the chromosome cannot be exploited, unlike the polar chromosome, where every position it can describe is valid. Since this also means that no checking on the validity of the solution needs to be performed at all, the polar description chromosome was adopted.

8.3 The fitness function

As has been discussed before, the fitness function is the only view the GA has on the search space it is investigating. The fitness function was initially constructed to

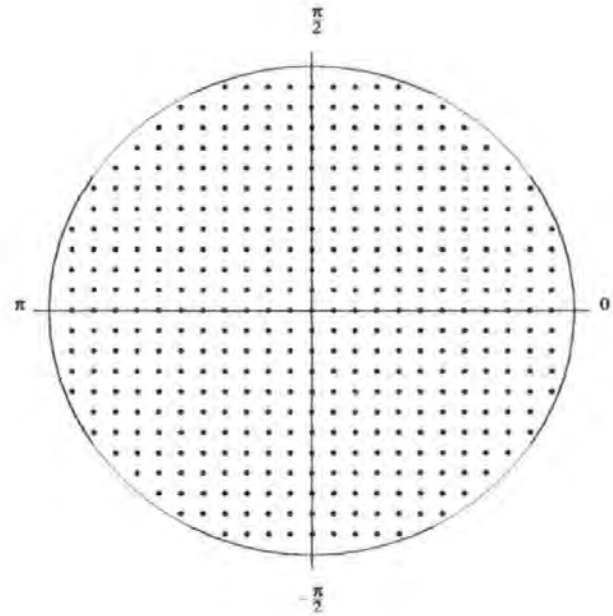


Figure 8.2: The possible pole-zero positions for a second-order section with quantised real and imaginary positions.

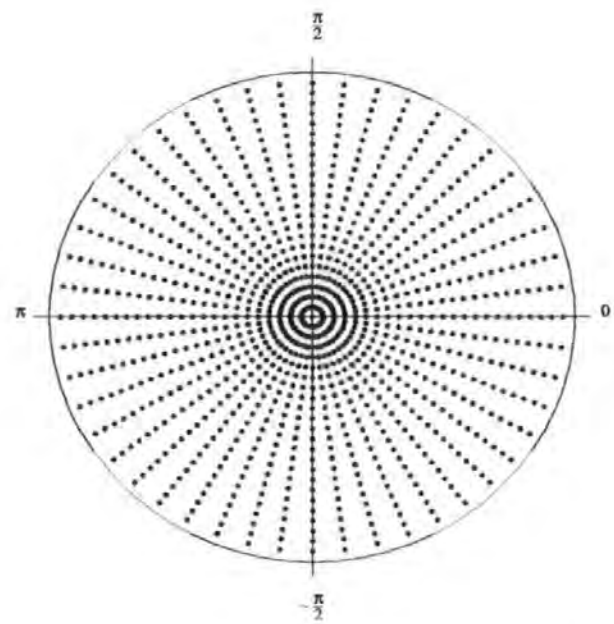


Figure 8.3: The possible pole-zero positions for a second-order section with quantised radii and angles.

return two performance measures, namely the error between the filter's magnitude response and a desired response template, and also the error from linearity of a region of the phase response. The latter was determined by fitting a straight line through the phase response in the specified region (covering most of the passband), and finding the mean-squared error between this line and the actual response. For simplicity, the chromosome described a filter of cascaded second-order sections, and was initialised, not with a perturbed BZT solution, but with randomly-selected poles in the passband region and zeroes in the stopband. Directing the initialisation in this way meant that the solutions in the first population contained a wide range of pole positions which were in the appropriate regions of the search space, ensuring that the population is diverse enough to ensure efficient searching.

A Matlab script (Appendix A.3) was developed to generate the initialisation files for the GA from a desired filter specification, and to display the response for a traditionally-designed filter of the same specifications, with full-precision coefficients. The response of the filter was determined in the fitness function by passing an impulse through the filter, followed by 1023 zeros, to produce the impulse response. An FFT was then taken of this to give the complex frequency response, from which the magnitude and phase responses were then extracted. The script generated a 512-point template file from the requested pass- and stopband widths, ripple and attenuation, which the GA read in and used to determine the filter's fitness with respect to the frequency response.

8.4 Effects of quantisation

When designing digital filters with standard methods, full-precision maths is used to produce filters with full-precision coefficients, internal calculations and storage. In practice, however, all digital systems employ a finite wordlength for both data storage and calculations. This has the effect of limiting the maximum attenuation of the filter, and also introduces noise into the filtered signal. The effects of both of these factors need to be considered when designing a practical system, but in 'traditional'

design methods they are investigated one at a time, after a full-precision filter has been designed.

While these quantisation effects may not matter for loose tolerance filters, when the constraints become tighter, they become significant, and may make it impossible for a solution to be optimal with respect to both criteria simultaneously. This would lead to a system which is sub-optimal with respect to at least one design criterion—the final filter can either have a good magnitude response, or a good noise response, but not both. A major aim of this section was to integrate finite wordlength effects analysis into the design process so that all of the criteria can be traded against each other by the GA, allowing it to produce a selection of quantised coefficient filters, with a range of tradeoffs, in a single design step.

8.4.1 Coefficient quantisation

The effect of quantising the coefficients is generally to alter them from their optimum full-precision values to sub-optimal values. There exists an optimum set of coefficients for each wordlength with respect to each design criterion, but this will usually not be the same as the full-precision set, so quantising the full-precision results will not give as good a filter as the optimum for that wordlength. Since shorter wordlengths can only represent a smaller number of values, they will cause a greater perturbation of the full-precision values, while for longer wordlengths of 16 bits or more the quantised filter might still be of sufficiently high performance to be used directly. Generally, however, since the optimum finite-precision filter is different to the quantised optimum full-precision filter, some additional form of optimisation will be necessary to raise the performance of the quantised full-precision solution. For some filters, the two sets of values will be very similar, so little further optimisation is required, while others, particularly IIR filters, are more sensitive to changes in wordlength, and may have very different coefficients.

8.4.2 Noise

A further complication is introduced when using a finite wordlength system in that the results of calculations are also quantised (see Section 2.3.3). This has the effect of introducing noise into the system as the values returned from calculations are not their true values, which can in turn affect further calculations. FIR filters are relatively unaffected by this, having a noise factor which is purely dependent on the wordlength of the system, while IIR filters' recursive nature means that the effects of quantisation noise are much greater and can lead to lower stability, or even instability.

Many modern DSP chips have long wordlength accumulators, so calculations only have to be quantised down to the system wordlength when intermediate results need to be saved back to memory, which reduces quantisation's deleterious effects on the filter response. This model was used in the fitness function to approximate the effects of having a finite wordlength system, by quantising the results of calculations when they were being stored.

8.5 Filter structure

A further aspect of IIR filter design, and the final one which was examined, is that of finding the best filter structure for the desired filter characteristics. IIR filters can be designed with a structure other than the cascade used in the previous optimisations (Chapter 5). Since the different structures and second-order sections have different noise characteristics, there will be a different optimum structure depending on the pairing and ordering of the coefficients, and a different optimum pairing and ordering depending on the filter structure. Under the standard design methodology, determining which of these structures is the best is another separate optimisation step, whose results may indicate the need to iterate back one or more stages and repeat previous optimisations or even completely re-design the filter to improve on its performance with respect to one or more of the design criteria. This approach makes the design process complex and requires the repeated sequential examination of the filter's per-

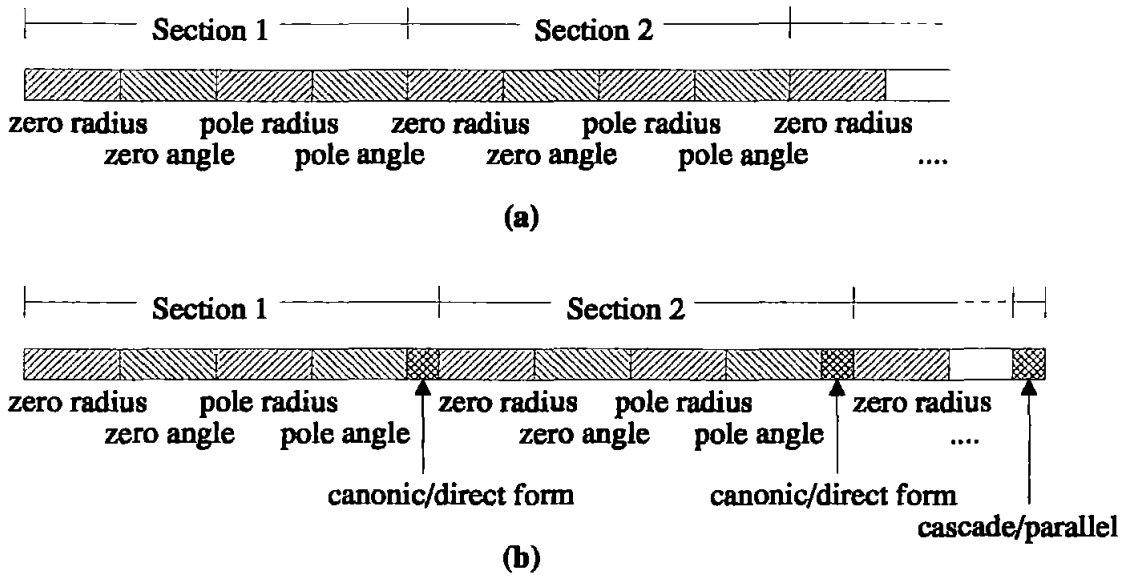


Figure 8.4: The structure of the chromosomes used to optimise (a) the coefficients and (b) the coefficients and structure of IIR filters.

formance with respect to different design criteria. It should be possible, by extending the chromosome to include bits which determine the structure of the filter, to allow the GA to search for those structures which give the best filter performances with respect to each of the design criteria. The inclusion of this ability will permit the GA to use whichever structure gives the best performance, and to search for this best structure simultaneously with the coefficients. Although the structure is not examined explicitly within the fitness function, its inclusion in the calculations means that the GA will also automatically search for the structure which gives the best performance for each set of coefficients.

The binary chromosome, which initially contained a simple list of the radii and angles of the poles and zeros of each second-order section as shown in Figure 8.4a, was extended to include bits to control if each section had a canonic or direct form, and also one bit to determine the structure of the whole filter, specifying whether it had a parallel or cascade structure, as shown in Figure 8.4b. This was later simplified by removing the zero radius and setting it to unity within the fitness function.

This type of *structured* chromosome allows the same chromosome to describe a number of different filters, all of which are valid. The control bits simply tell the

fitness function how to decode the information in the chromosome. Storing the data in the chromosome this way means that all of the filters described by the chromosome are valid. Tang et al [90] have shown structured chromosomes to be suitable for optimising IIR filter structures and coefficients simultaneously by GA. If the chromosome shown in Figure 8.4b was used to optimise a filter of S second-order sections, there would be a total of $2.S! + 1$ possible filter structures, comprising $2.S!$ cascaded structures and one parallel one. When the cascade/parallel bit is set, telling the fitness function to use a parallel structure, then the bits which determine whether each section has the canonic or direct form are ignored, and the pole and zero positional information would be used to calculate the equivalent parallel filter coefficients.

8.6 Use of the GA

The GA used for this optimisation was the same as was used for the non-linear FIR optimisations, using an MCO approach to search for the POS of non-dominated solutions. The binary chromosome used initially contained a list of the positions of the poles and zeros of second-order sections, which were taken to exist as complex conjugate pairs.

The fitness functions used were simply the maximum error between the magnitude response of the filter and a given desired response template, and the mean squared error between the phase response and a best-fit straight line over a specified region of the passband.

8.7 Results

The GA was initially run with fitness functions designed to optimise and trade off the magnitude response performance against the linearity of the phase response over a given region of the passband. Initial results showed that the solutions within the GA's NDS were strongly biased towards solutions with near-linear phase in the passband, rather than with a good magnitude response. The range of fitnesses returned by the

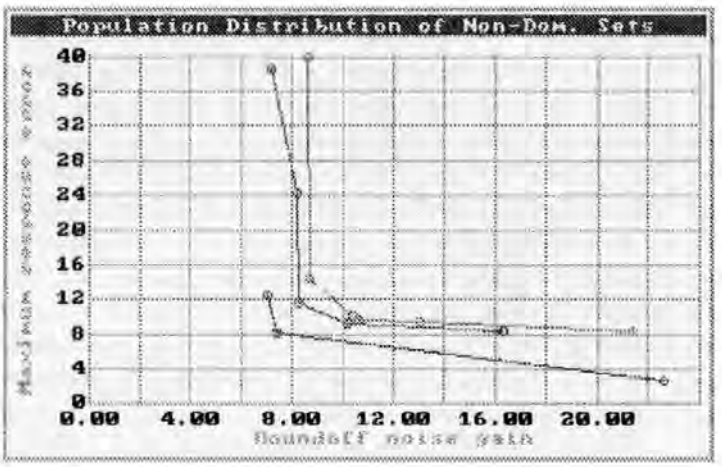


Figure 8.5: Example of an early non-dominated set.

two functions differed by an order of magnitude, so an attempt was made to equalise the two by scaling them to an approximately equal range. This had the effect of slowing, but not halting, the discovery of individuals with good phase response, but did not force the search to find solutions with good magnitude responses. It was found that giving the GA the ability to change the type of each second-order section made little or no difference to the quality of the solutions found.

After many unsuccessful trials, the investigation was moved to the optimisation of a filter's noise and magnitude response together, and to exclude the phase response. To simplify the noise calculations, the structure was fixed to use only the canonic second order section. At this stage error spectral shaping (Section 2.3.4) was also added, but had no noticeable effect on the solutions found.

A selection of results from these optimisations are shown in Figures 8.6—8.14, and summarised in Table 8.1. In the captions for these figures *dp* refers to the passband ripple and *ds* to the stopband attenuation, both in decibels (dB). Figure 8.5 is a typical example of the first three non-dominated sets in a population early in a run. The 'wavefront' can clearly be seen, along which performance trade-off is occurring.

All of the results are shown after a run of 1,000 generations for a lowpass filter with band edges at 0.1 and 0.2, and all but the last are for a 16-bit wordlength It was found that if a suitable solution with low magnitude response error has not been found

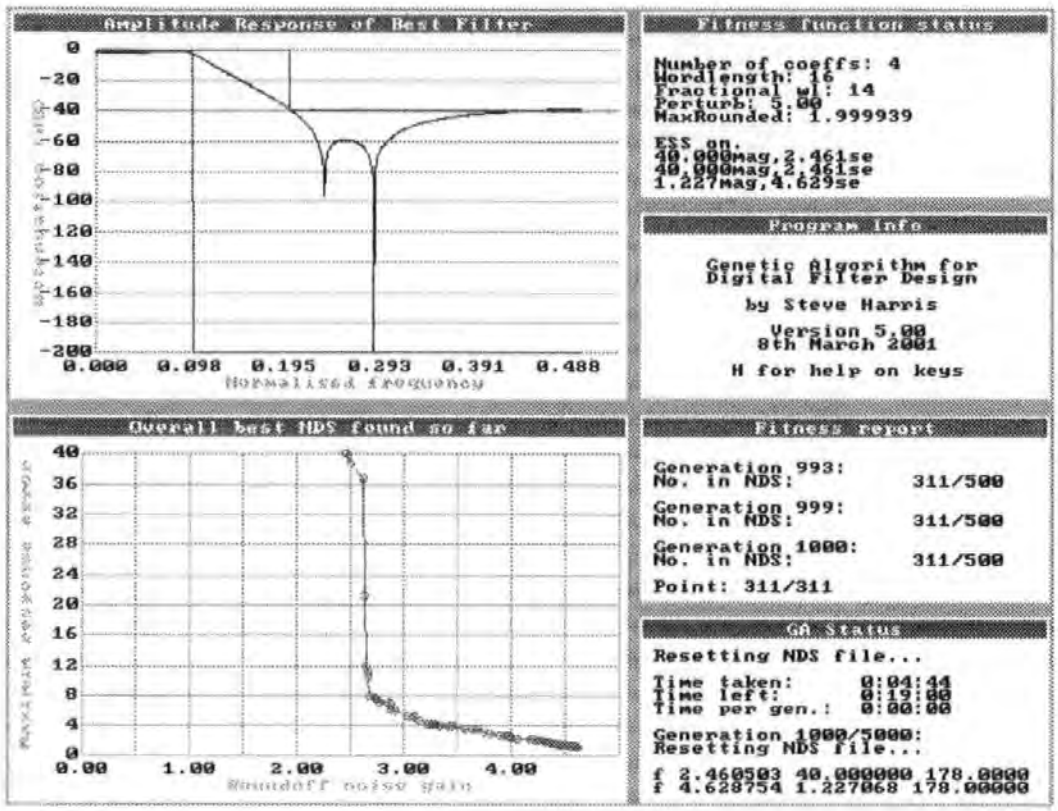


Figure 8.6: Best result for a fourth-order filter with target dp=0.5dB, ds=40dB, 16 bit wordlength.

Wordlength	Order	Target passband ripple	Target attenuation	Actual max. error	Roundoff noise gain
16	4	0.5	40	1.227	4.629
16	4	0.75	40	0	6.052
16	4	1	40	0	5.795
16	4	1	50	1.456	3.963
16	6	1	60	0	8.753
16	6	1	70	0	10.598
16	6	1	80	4.560	11.703
16	6	1	90	4.182	35.755
24	6	1	80	0.678	16.476

Table 8.1: Comparison of a variety of best-magnitude response results found by the multi-criterion GA. The last two columns show the actual error from the target template and the roundoff noise gain.

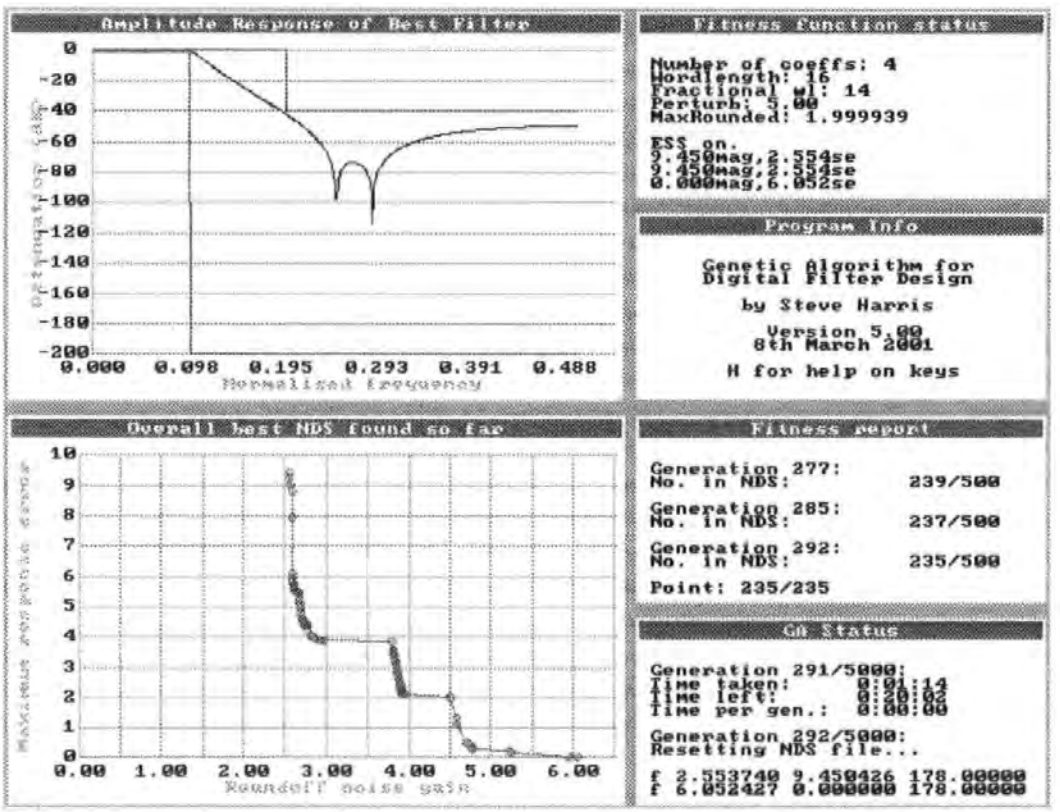


Figure 8.7: Best result for a fourth-order filter with target dp=0.75dB, ds=40dB, 16 bit wordlength.

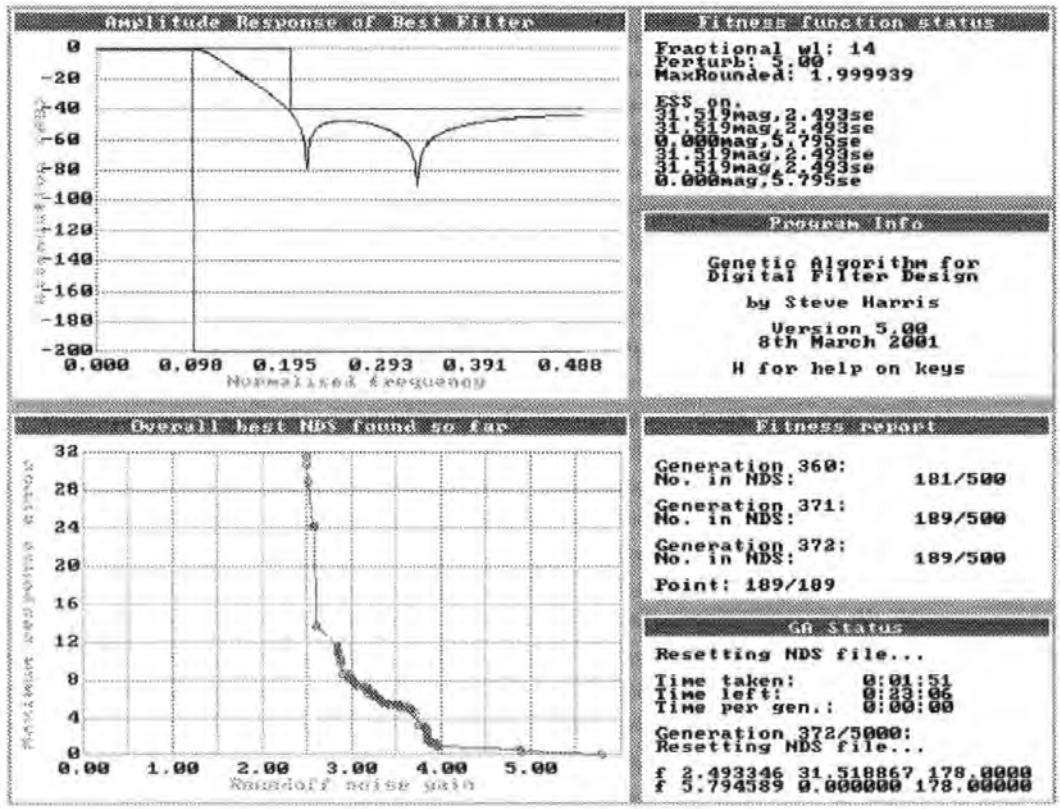


Figure 8.8: Best result for a fourth-order filter with target dp=1.0dB, ds=40dB, 16 bit wordlength.

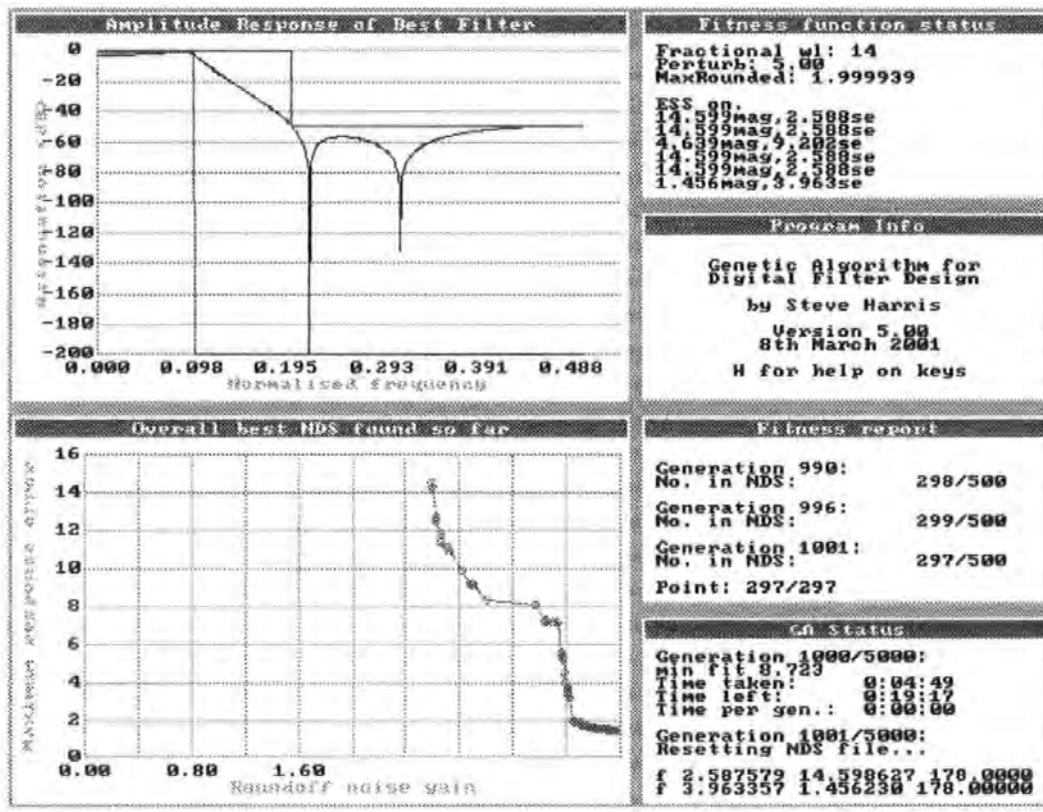


Figure 8.9: Best result for a fourth-order filter with target dp=1.0dB, ds=50dB, 16 bit wordlength.

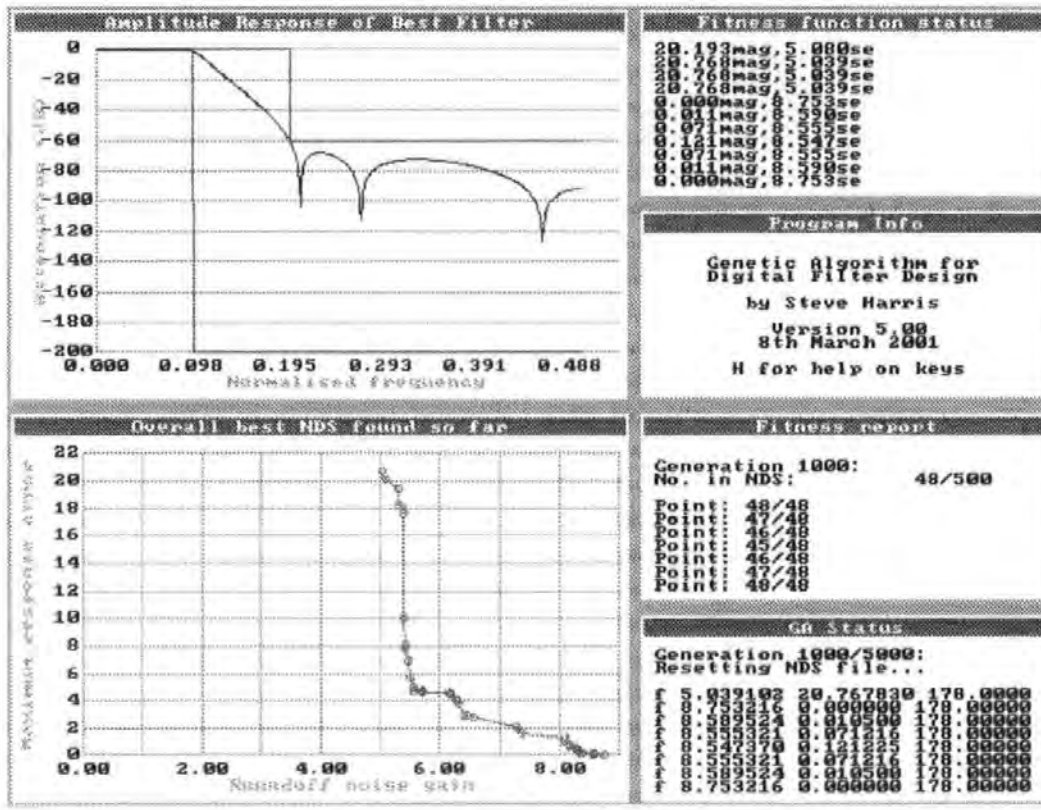


Figure 8.10: Best result for a sixth-order filter with target dp=1.0dB, ds=60dB, 16 bit wordlength.

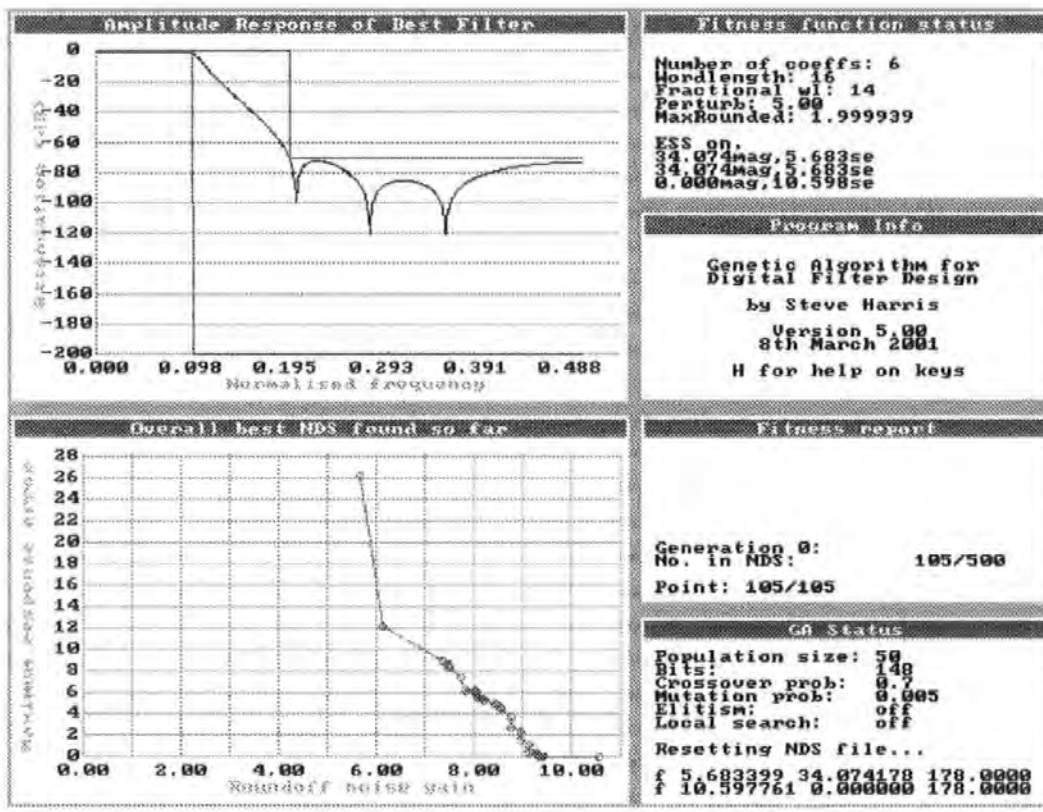


Figure 8.11: Best result for a sixth-order filter with target dp=1.0dB, ds=70dB, 16 bit wordlength.

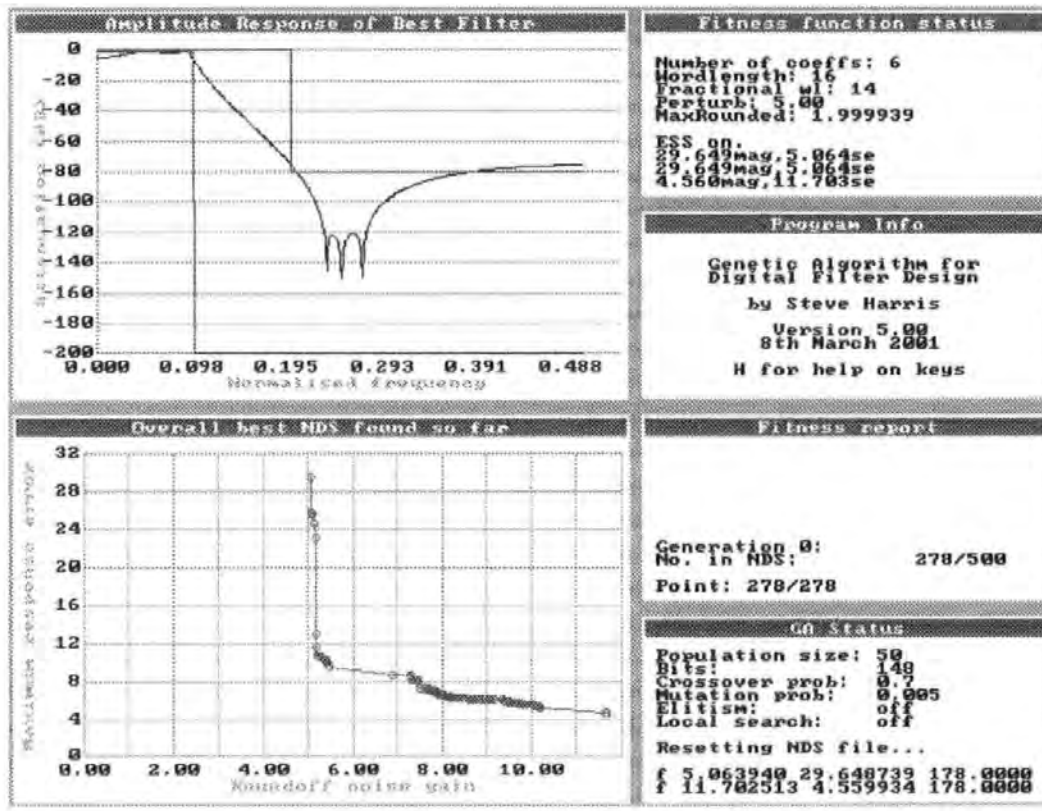


Figure 8.12: Best result for a sixth-order filter with target dp=1.0dB, ds=80dB, 16 bit wordlength.

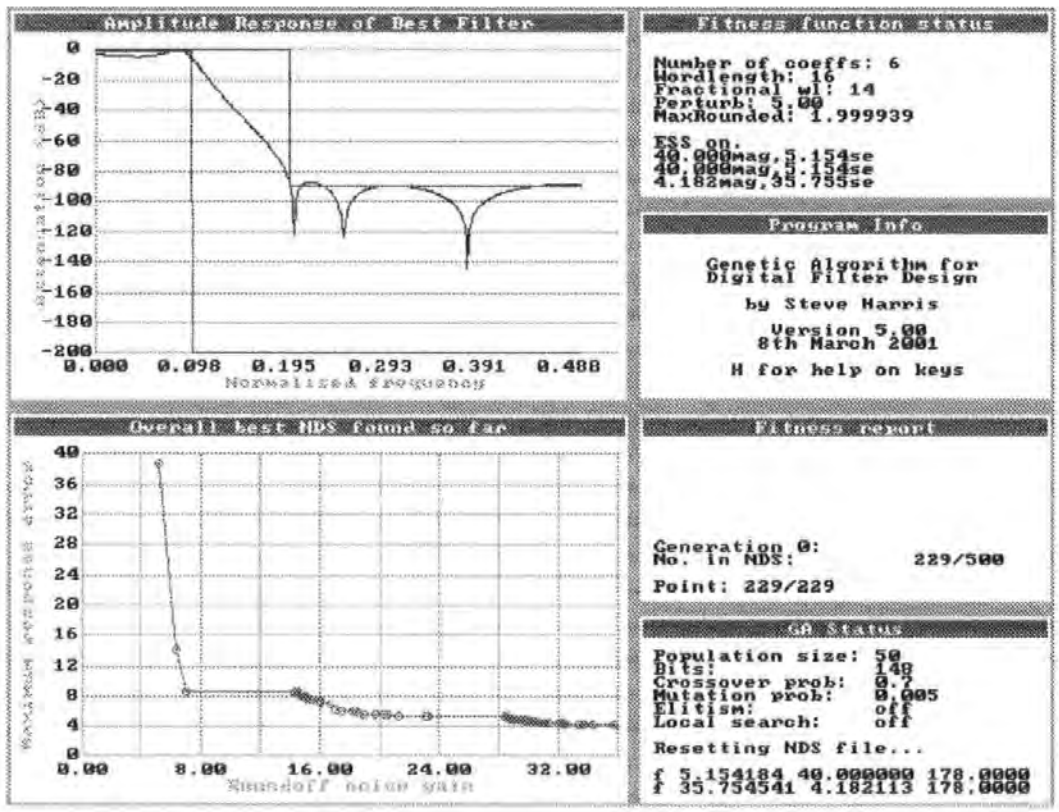


Figure 8.13: Best result for a sixth-order filter with target dp=1.0dB, ds=90dB, 16 bit wordlength.

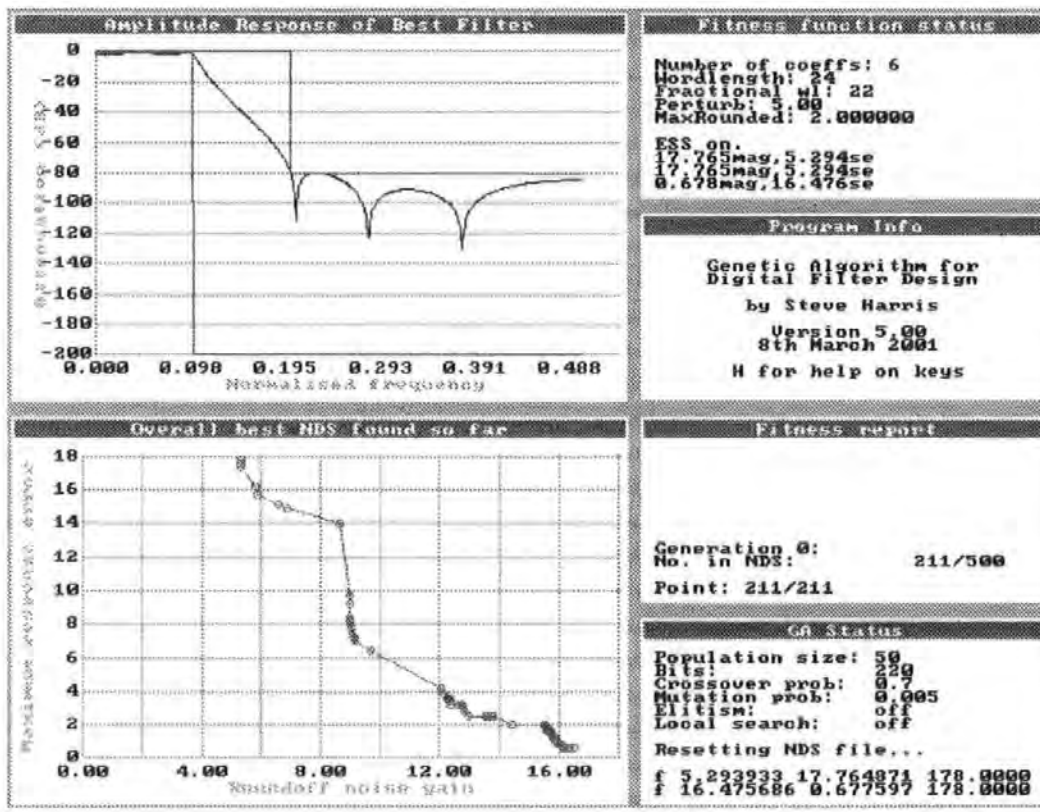


Figure 8.14: Best result for a sixth-order filter with target dp=1.0dB, ds=80dB, 24 bit wordlength.

after 1,000 generations, then one was not usually found even after 5,000 generations. The solution with the best magnitude response is shown in the upper left box, with the NDS over all points examined so far in the lower left. The GAs were run with a population size of 100, a crossover probability of 0.6, and a mutation probability of 0.01. Different mutation probabilities were tried from 0.001 to 0.05 with little variation in performance.

The first results, in Figure 8.6, are for a filter with a desired dp of 0.5dB and a ds of 40dB. These show that the GA was able to find a solution close to the template, but with a magnitude response error 1.227dB outside the template. The shape of the NDS shows that the GA was able to find many solutions, but none of them came close to the template. There is also a sharp rise in the magnitude response error when the noise starts to fall. In this case, the GA has not been able to find a range of suitable solutions, as was hoped.

The results in Figure 8.7 are for a similar filter, but with a desired dp of 0.75dB. In this case the GA was able to find one solution which fit the magnitude response template, and one other close to it, with a slightly lower noise, but the response error then rose sharply as the noise decreased. The 'stepped' nature of the NDS in this case indicates that the search space has different characteristics to the previous search.

In Figure 8.8, the passband tolerance has been widened again to a dp of 1dB. Again, a suitable solution was found, but with a slightly lower noise than for a dp of 0.75. This was to be expected, as the looser frequency response tolerance releases more degrees of freedom for noise reduction. In this case, only one good solution was found, the second-best with respect to the magnitude response being noticeably outside the template.

As the passband tolerance has been widened there have been progressively fewer solutions in the NDS. This is due to the changing nature of the search space, but the exact reasons are unclear.

The next set of results illustrate the change in performance as the stopband tolerance is tightened. Figure 8.9 shows the best result for a desired template with a dp of

1dB and a ds of 50dB. It is clear that in this case the best solution is unacceptable, lying almost 1.5dB outside the template, although the noise figure is low. It is unclear if this poor result is because the magnitude response template is unattainable at this wordlength, or whether low-noise solutions are easier to find, which drove the GA away from good magnitude response solutions.

When the desired attenuation was increased to 60dB for the next run, the Matlab initialisation script predicted that a sixth-order filter was necessary, so the order was increased for the remaining runs. This longer filter could achieve higher-attenuation filters than 60dB, so the search space contains a variety of solutions at 60dB. This more favourable situation meant that the GA was able to find an acceptable solution here, and also in the next run at a ds of 70dB, shown in Figure 8.11.

In Figure 8.12 however, with a desired attenuation ds of 80dB, the best solution was 4.5dB outside the desired template, either because the search space was too hard for the GA as the optimum region was too small, or because the filter specification was unattainable. When the attenuation was increased further to 90dB, as in Figure 8.13, the best filter is actually slightly closer to the desired template than in the 80dB case. However, this has been achieved at the expense of a much higher roundoff noise gain. The fact that the GA was able to equal the accuracy of the 80dB search with a 90dB template implies that a much better solution is possible with an 80dB template, and that there may actually be an exact one with respect to the magnitude response.

The 80dB template optimisation was repeated with a wordlength of 24 bits, as shown in Figure 8.14. As can be seen, the best solution was only 0.678dB outside the template, as compared with 4.56dB for the 16-bit solution. This improvement is probably due in part to to the existence of more solutions with at least 80dB attenuation in this search space, but the very different spacing of the zeros shows that the solutions come from widely separated regions of the parameter space. It seems probable that there is a 16-bit solution similar to the 24-bit one, but the GA had simply not found it.

A comparison of a selection of the filters in Table 8.1 with those obtained by quan-

Order	Wordlength	Desired attenuation (dB)	GA error (dB)	Quantised BZT error (dB)
4	16	50	1.456	1.07238
6	16	80	4.560	1.2439

Table 8.2: Comparison of magnitude domain results found by GA and by quantising the BZT coefficients for an elliptic filter designed in Matlab.

tising the coefficients of elliptic filters found through standard techniques using Matlab, is given in Table 8.2, where it can be seen that the GA solutions are only a couple of dB worse than the quantised-coefficient BZT filters. However, using such a BZT solution has the disadvantage that the phase response is uncontrolled and non-linear, and the degradation of the noise performance caused by quantising its coefficients is also uncontrolled.

Due to the limited success of the GA with this cascade filter design, it was decided not to attempt the design of a parallel-structured filter, as the indirect relationship between the pole-zero positions and the parallel filter coefficients implied that it would be a much harder problem for the GA to optimise.

8.8 Discussion

The results have shown that the GA is only able to find a good solution with respect to the magnitude response when the desired template has loose tolerances. When the passband ripple is reduced, or the stopband attenuation increased, the performance drops dramatically, and often the GA does not find an acceptable solutions, at least for the magnitude response.

The performance of the GA with respect to the magnitude response suggests that it is a harder problem to solve than reducing the noise or finding a partially near-linear phase response, at least from a GA perspective. Despite the fact that the filter is made up of short second-order sections, each of which is a distinct unit, the filter as a whole still needs to have the right characteristics for it to have a good response. This means

that the GA has to find entire good solutions in one go, rather than optimising a part at a time. The fact that all the coefficients work together means that altering one moves the optimum for all the others, so it is not possible to predict the fitness from just one coefficient, or even one second-order section. This makes it very hard for the GA to build up good genes. However, the fact that seeding the GA with a perturbed BZT solution allows the GA to come close to the performance of the quantised-coefficient BZT filter, while also providing some control over the noise or phase characteristics, means that the technique shows some promise. It is probable that future improvements to the GA technique could allow it to equal or exceed the quality of the quantised BZT performance in the frequency domain.

The better performance with respect to the phase and noise characteristics suggests that there is a more straightforward relationship between them and the filter coefficients. It also therefore appears that it is easier to predict the overall phase or noise performance from a single second-order section, than to predict the overall frequency response. A future analysis of how changes in chromosomes and pole-zero positions affect the noise, phase and magnitude responses could be used to guide future development of a GA-based optimisation, in order to increase the performance with respect to the magnitude response, which is perhaps the most important aspect of the filter design. It also seems likely that an alternative, more efficient method of allowing the poles and zeros to be reordered and paired would increase the ability of the GA to find more efficient structures.

The successful optimisation of a parallel structure filter would also require additional analysis, and perhaps a different approach to that for a cascade filter, because of the different relationship between the pole-zero positions, the parallel section coefficients, and the filter response.

Chapter 9

Conclusion and Future Work

This project has set out to investigate aspects of the use of optimisation within the digital filter design process. The design process has been analysed, and opportunities identified within it where the current methods could be improved by the application of alternative techniques. A number of these situations were selected, and coded in a method suitable for optimisation by Genetic Algorithms and Simulated Annealing. A variety of GA techniques were involved, culminating in the implementation of a multi-objective GA, capable of investigating the performance of digital filters with respect to a number of different measures simultaneously, and of trading off these against each other to produce a range of solutions which are potentially useful in a variety of situations.

The GA was found to have varying levels of success in optimising a range of filter types, from some where it did not work well, to others such as the hybrid Frequency Sampling designer, which produced excellent results. Its varying performance gave insights into the applicability of GA-based optimisations in the digital filter design process, and into how these could be altered in future to improve their performance.

9.1 Review

9.1.1 Digital Filter Design

The digital filter design process is an iterative, multi-step process in which the performances of different aspects of the design are examined, and the filter altered to improve them. This process can require many iterations to obtain a suitable system, and the way the process is divided into stages means that it is very difficult to trade off the performance of, say, the frequency response against the roundoff noise gain. The whole process was analysed, and stages identified which could potentially benefit from GA-based optimisations. These results were used to select filter design tasks to use to further the investigation.

9.1.2 Finite Impulse Response Filters

Initial investigations into the design of FIR filters involved the optimisation of Frequency Sampling filters by adjusting their transition sample values. A floating-point GA proved able to find good solutions for filters with up to around four transition samples. When filters with a wider transition band were used, the performance dropped noticeably, and a hybrid Simplex-method hill-climber was added to the GA to complete the optimisation. This technique proved able to produce solutions which equal or better those in the published tables of Rabiner et al [5]. The tables are limited in their scope, requiring the transition sample values to be interpolated for untabulated filters, whereas the hybrid GA technique can be also be applied to any such unlisted filter, producing a more accurate result.

The fitness function was extended to optimise the transition sample values of finite wordlength filters, and to simultaneously search for the minimum wordlength necessary to achieve a desired attenuation. Using 'traditional' methods of filter design, such an optimisation would require a great deal of additional mathematical analysis [76, 77], but the non-problem specific nature of the GA meant that it was possible to perform

this optimisation using the same GA as before, with just a small change to the fitness function. This was altered so that the floating-point coefficients were quantised to a specified wordlength before use, and the GA was then found to be able to produce high-quality solutions without the hybrid hill-climbing algorithm.

The search space for a quantised-coefficient filter contains many fewer points than a 32- or 64-bit floating point implementation, but will be discontinuous because changing the wordlength will perturb the coefficient values and cause a step change in the filter performance. The means that a hill-climber cannot optimise the wordlength simultaneously with the coefficient values, but the GA is generally unaffected by discontinuities in the search space, and is able to operate effectively on this problem.

This work resulted in two papers, the first of which was published in the Proceedings of the 1993 IEE/IEEE Workshop on Natural Algorithms in Signal Processing, and introduced the design of Finite Impulse Response (FIR) filters by GA [2]. A second paper, extending the work presented in the first, was published in the IEEE Transactions on Signal Processing in 1998 [4].

A subsequent investigation into the potential use of a Multi-criterion GA for optimising quantised-coefficient non-linear phase FIR filters proved unsuccessful. Removing the linearity constraint allows a filter to have the same magnitude response with fewer coefficients, and the GA attempted to find such a solution, with as linear a phase as possible over a specific region of the passband. An investigation into the GA-difficulty of the problem revealed that the selected representation was not suitable for the GA. This was because it was not possible to predict the filter performance from a subset of the chromosome, and the only good solution was a complete solution. Genetic Algorithms only examine point samples of the search space, so the analysis of the difficulty of the problem also only required point data to be collected, by storing a large set of random chromosomes and their corresponding fitnesses. This characteristic of the GA means that a single implementation of the difficulty calculations can be used for a wide range of problems, simply by swapping the fitness function which is used to calculate the performance of each chromosome.

Although the use of a binary, finite-precision chromosome meant that each coefficient could have fewer values, there were many more coefficients and each coefficient's effect on the fitness was harder to predict. These characteristics make the problem difficult for a standard GA to optimise. Recent work by Lu and Tzeng [97, 98] has successfully used a GA to produce non-linear FIR filters, by utilising a guided crossover technique which allows their GA to perform more effectively in this unfavourable search space.

9.1.3 Infinite Impulse Response Filters

The initial attempt to optimise the pole and zero locations of an IIR filter using the same floating-point GA as for the Frequency Sampling filter investigations was unsuccessful because random coefficient initialisation was used and there is a lack of useful structure in the search space far from the optimum. However, this illustrates an advantage of the GA, in that the same implementation can be used for a wide variety of problems by simply changing the fitness function. As it was desired to optimise the a and b coefficients directly in a quantised form, a binary GA was implemented which included a weighted-sum fitness function, intended to reduce the noise once a solution had been found which fitted the desired magnitude response. The results from initial experiments using random chromosome initialisation were not encouraging, so the GA was seeded with a known good solution found using the BZT method. This produced better results, although the unperturbed quantised BZT solution itself was of comparable performance to the best GA solution. Another technique, Simulated Annealing, was also applied to the same problem, and was found to perform a better trade-off between the two performance measures, as detailed on page 79, where it was able to find a solution which was only just within the desired magnitude response template, giving it more freedom to improve the noise response.

An analysis of the search space for this problem revealed that it only had a structure suitable for the GA close to a good solution, so it could not be expected to find the

optimum solution from a random chromosome initialisation. The SA was less affected by the nature of the space, and was able to perform better, but still required seeding with a BZT solution. It was, however, able to find a solution which traded off the performances with respect to noise and frequency response more effectively than the GA.

Like GAs, the SA technique is able to operate effectively in a discontinuous, constrained search space, and does not require a mathematical analysis of the problem, just a suitable implementation of it. SA does not work by building up a good solution from building blocks, so is able to work successfully with some problems which are GA-hard; conversely, there are problems where the GA can make use of high-fitness building blocks to outperform the more random search of SA. However, due to the non-problem specific nature of both techniques, the same fitness function can be used, as they both only take the same point samples of the search space.

The second approach was to use a true multi-criterion optimisation (MCO) GA to optimise the filter with respect to the magnitude response plus either phase response or roundoff noise gain, with the aim of providing a range of solutions with different tradeoffs, from which the most applicable could be selected by the designer. This GA was also unable to find a solution from a totally random initialisation, so it was seeded with poles and zeros, randomly positioned within the pass- and stopbands respectively. The fitness functions adopted differed from those used previously for IIR filters, in that the performances were measured separately, instead of being combined in a weighted sum. The GA proved able to find solutions with a near-linear phase in the passband, but the set of solutions did not produce a wide range with acceptable magnitude responses, and, although it regularly failed to find even a single solution which exactly fit the desired response template, the maximum deviation from this template was often just a few dB, whih may be acceptable if it occurs in the stopband.

The same GA was used with a different fitness function, with the aim of trading-off the magnitude response with the noise gain, but although solutions were found with acceptable magnitude responses, it was again discovered that the GA did not

find a range of acceptable filters, and that as the tolerances (passband ripple and/or stopband attenuation) became stricter, fewer and sometimes no filters were found with acceptable frequency responses. This appears to be because it is possible to break down the noise problem into simpler parts, and if one section has a low noise, then the whole filter will generally have a lower noise. However, all but one section could have an optimal frequency response, but if the last does not, the performance of the filter as a whole will be poor. This means that only a filter with a suitable magnitude response in every stage will have a good performance. This inability to break down the problem with respect to the magnitude response makes the problem hard for the GA.

The ease with which the GA could be changed from optimising the frequency response to optimising the noise illustrates a major advantage of the technique. As the GA is problem-independent and only takes point samples of the search space, no mathematical analysis or manipulation of the problem need be undertaken, which can be complex, especially in fixed-precision environments. This means that the same MCO technique and implementation can be used to perform several trade-off optimisations, simply by changing the fitness functions used, which in turn are based on the same implementation of the filter. In other words, once the problem has been implemented, the same GA can be used to optimise its performance with respect to a range of different criteria by selecting which measures are returned by the fitness functions. This feature of the GA makes it a suitable technique for optimisations in the domain of filter design where a wide range of disparate performance measures can be used, including magnitude and phase responses, noise performance, wordlength, filter order, and even implementation costs such as chip area and complexity. It would be extremely complicated to design a technique to trade off magnitude response and implementation complexity using 'traditional' methods, but this can be performed with an MCO GA by encoding the problem in a suitable chromosome and implementing the necessary fitness functions [45].

9.1.4 Optimisation techniques

A variety of optimisation techniques were employed in this project, but centred mainly on the GA. Two types of GA were used at different times, the first using a floating-point chromosome, which was coupled with a hybrid Simplex-method hill-climber. This was successful at optimising Frequency Sampling filters. The second type was an MCO binary-chromosome GA, which was used on a wider range of problems but proved not to be universally successful. While it could often find solutions with good noise or phase performances, it was generally less successful in optimising the magnitude response.

The GA has shown itself to be a widely-applicable technique which can be easily adapted to a range of problems, ranging from magnitude and phase optimisations to noise analyses and implementation cost reductions. It has the great advantage over traditional techniques that it can be used where direct mathematical techniques are very difficult to perform, such as the direct optimisation of quantised IIR filter coefficients with their pairing and ordering, and phase or noise responses, in a single step, as in Chapter 8. Once a filter has been implemented it is possible to perform a range of optimisations from the same code, simply by changing the performance measures returned by the fitness function.

A Simulated Annealing method based on the Simplex hill-climber was also used for one IIR-based optimisation, and was found to perform better than the GA, probably because it was less affected by the structure of the search space, as it does not rely on the same short, high fitness building blocks.

9.2 Future Work

9.2.1 New areas in FIR Filters

The existing work has covered a relatively small range of filter types, and many remain unexplored, some of which could prove to be better suited to a GA-based optimisation. The results of the analyses which have been undertaken here can offer guidance to the

selection of areas of future investigation, and show that a careful initial examination of the design problem and the nature of the search space will prove advantageous when deciding on how best to describe and code the problem in order to make it suitable for optimisation by GA. In particular, it should be possible to make a prediction as to the performance of a filter from just a portion of the chromosome, as the GA's performance is especially poor when there is no structure in the search space to guide it towards optimum solutions.

The successful finite wordlength, linear-phase Frequency Sampling (FS) FIR optimisations could be extended to include a full implementation of a finite wordlength filter. This would allow the coefficient optimisation to take product roundoff into effect, without requiring any changes to the GA.

One area of FIR filter design which appears promising is that of optimal filters, in which the optimisation task consists of searching for the extremal frequencies. Since the performance of an optimal filter only degrades slowly as extrema are moved from their optimal positions, it ought to prove a suitable subject for a future GA investigation, using a chromosome containing a list of the extremal frequencies. Optimal filters have the advantage that the extremal frequencies can be estimated very quickly to give an approximate solution which could be used to seed the GA and allow it to start in the general location of the optimum. The fast speed of the optimal design process (McClellan et al [15] reported that, even in 1973, their Remez exchange-based method only took a few seconds) makes a GA approach unlikely to be worthwhile for standard designs. However, it may be useful in situations where the standard technique is not guaranteed to converge, such as with designing multi-band filters with varying transition widths, when there may be local ripples within the transition bands.

The lack of success with the non-linear FIR filter design showed that the method of implementing the search was not ideal. While the current system was run for 8,000 generations, and improvements stopped well before the end of the run, further investigations into the existing technique with extended run lengths could improve its performance. A different approach to crossover may prove beneficial, such as that

of Cotta and Troya [100], which picks the best of all the potential offspring, or Lu and Tzeng [97], which guides poorer solutions towards better ones. A fuller analysis of the relationship between coefficients and the magnitude and phase responses could also give a better indication of how to structure the chromosome to reduce the long-distance effects which are currently observed and which limit the effectiveness of the search.

An alternative approach could make use of the fact that, unlike linear phase FIR filters, where the maximum magnitude in the impulse response is in the middle, minimum-phase filters have the maximum magnitude earlier on. A Genetic Programming-based approach could search for a function which generates not only the shape of an impulse response, but also where its peak occurs. This would allow the system to automatically generate asymmetric impulse responses, which is difficult with the GA approach used in this project, as it is unlikely that crossover between two sets of coefficients with different peak amplitude locations will produce a correctly-formed, suitable impulse response.

9.2.2 New areas in IIR Filters

It has been found that the GA finds optimising the magnitude response of an IIR filter especially difficult, because it has to find whole good solutions, and cannot break the problem down. A method of simplifying the search space could be to use only integer or powers-of-two coefficients. This should improve the performance by vastly decreasing the size of the search space. As with the FIR, although fitness improvements stopped well before the end of each run, further analyses of the current system's performance may show even longer runs to be advantageous in some cases.

A second aspect of the IIR optimisation which could be made more efficient is that of the pairing and ordering of the coefficients. Although the second GA used here could theoretically have improved the filter's noise performance, in practise its ability to do so was limited because changing the order of any of the coefficients without changing their values was exceedingly unlikely. A better approach would be to have a

chromosome which kept the coefficients in a fixed location, but which also contained information on the order in which they should be used. In this way, simply by changing the fitness function, the same GA would be able to optimise both the coefficients and the order in which they are used, in one operation. A consequence of this approach should be the increased effectiveness of structural optimisation, which could allow the selection of the structure of the filter (parallel or cascade) and also the topology of each second-order section in a cascade structure. This should allow the GA to find the best structure, the best coefficients, and the best order to use them in, to give the best performance.

9.2.3 Further Natural Algorithm Techniques

While this project has concentrated on the application of GA and SA methods to filter design, it has become clear that they are not necessarily ideal techniques for all situations. Further investigations could improve the efficiency of the existing GA and SA applications and widen the applicability of Natural techniques by including additional methods. Ways in which this could be approached are given below.

9.2.3.1 Genetic Algorithms

It was found that chromosome initialisation was very important for most of the GA applications in this project, due to the lack of structure in the search space far from the optimum. Further investigations into the best initialisation method to use for each problem could improve the GA performance by giving it a wider set of high-fitness chromosomes to work with, as the method used here of seeding the population with perturbed copies of a single good solution will direct the initial search towards a single good area instead of allowing it to pick good areas from all of the search space.

While the GA is a widely-applicable technique, its best performance is often obtained by tuning the operators to each specific problem. For example, as discussed above, the pairing and ordering optimisation for IIR filters is inefficient with the gen-

eral methods used. A better performance would be obtained by either changing the chromosome to include coefficient ordering information, or by adding an additional operator to run alongside the existing crossover and mutation. This operator would move whole poles and zeros at random within the chromosome, thereby allowing the GA to sample a wide range of pairings and orderings, and making the noise optimisations far more effective. Other improvements could be made by tuning the crossover and mutation operators to specific problems, either manually, by including problem-specific knowledge, or automatically, as in the Frequency Sampling optimisations described in Chapter 4.

9.2.3.2 Simulated Annealing

Although the changes proposed above to the GA-based optimisation should improve its performance, the existing results for the SA showed that it could outperform the GA under some situations. If the problem under investigation is not especially suited to the GA then it might be advantageous to adopt an SA optimisation instead. Future investigations would have to determine this on a per-investigation basis, as it will not always be possible to determine beforehand which will perform better.

SA does have the disadvantage that it is a single-criterion optimisation method, so it is not so suitable for producing a range of solutions to an MCO problem as the GA. In some circumstances, however, it might be possible to design a suitable weighted sum fitness function to produce a single filter with a desired trade-off between two performance measures.

9.2.3.3 Tabu Search

Tabu search is an efficient optimisation method for combinatorial problems, but is less suitable for precision coefficient optimisation. It may however be possible to combine the Tabu search with a GA or SA to produce a hybrid technique which can perform both effectively. For example, within the fitness function of a GA or SA being used to optimise filter coefficients, a Tabu search could look for the best filter structure to

use with each set of coefficients, thereby producing a more wide-ranging and flexible technique, which requires less user intervention than traditional design methods.

Alternatively, a Tabu search could use a low-precision version of the fitness function to look for high-fitness regions of the search space. This information could then be used to seed a higher-precision search using GA, SA or even a hill-climber.

9.2.3.4 Genetic Programming

Genetic Programming (GP) is used for different optimisation problems than those investigated in this project, in that it searches for expressions which perform particular functions, rather than for numeric values which fit into fixed models. A filter design method which seems ideal for a GP approach is the Window method. Standard techniques use a limited range of functions, each of which have different characteristics and different limitations. A GP-based optimisation could produce whole new families of window functions, each tailored to specific problems, or even a different function for every problem. An advantage of the GP approach is that it would not have a human designer's preconceptions about what a window function should be, and would simply look for the function with the best performance. This makes it theoretically possible that a GP optimisation of any filter could find a solution whose performance equals or betters that of any standard, general-purpose window function.

9.3 Intelligent Filter Design Tool

In the light of the results found so far, and the analysis of the filter design process, it is now possible to outline a potential intelligent automatic filter design tool. It is clear that no single optimisation technique is suitable for all aspects of digital filter design, and that to fully automate the process will require a range of methods, working together, and complementing each other's capabilities.

A possible structure for such a tool is illustrated in Figure 9.1. The system is controlled by an Expert System (ES), and would contain an extensive library of tech-

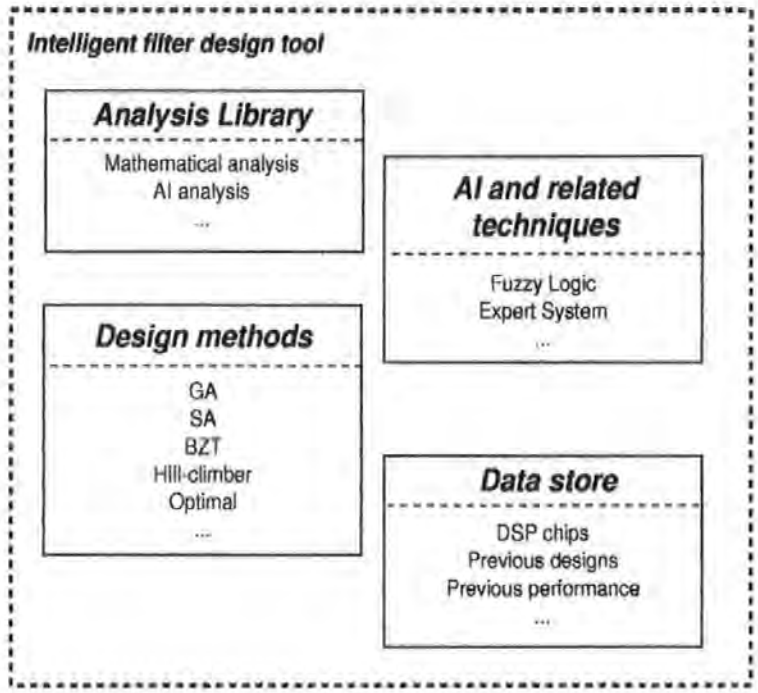


Figure 9.1: Illustration of the potential structure of an intelligent filter design tool.

niques, plus a database of its previous decisions and their associated performances. It examines the available data, and runs analyses to determine which filter structures and design methods are appropriate for that problem. The selected optimisations are then performed, and the results examined to determine if they are acceptable. If they are, it presents them to the user, but if they are not, it retraces its steps and uses another method or alters variable parameters and tries again. In this way, it will be able to rapidly try a variety of ways of achieving the desired specifications, or if these are not attainable with the given constraints, it will produce the nearest filter and possibly suggest changes to the design to improve its performance. The results found for each technique can be stored to give additional guidance to the system when it is deciding the best technique to use in future.

As an example, suppose a designer wants a linear-phase, lowpass FIR filter, with a Nyquist frequency of 10kHz, 0.1dB passband ripple, and 80dB stopband attenuation, and band edges at 3kHz and 6kHz. The system might work as in Figure 9.2. Having entered the known information, the tool determines that the wordlength, target system,

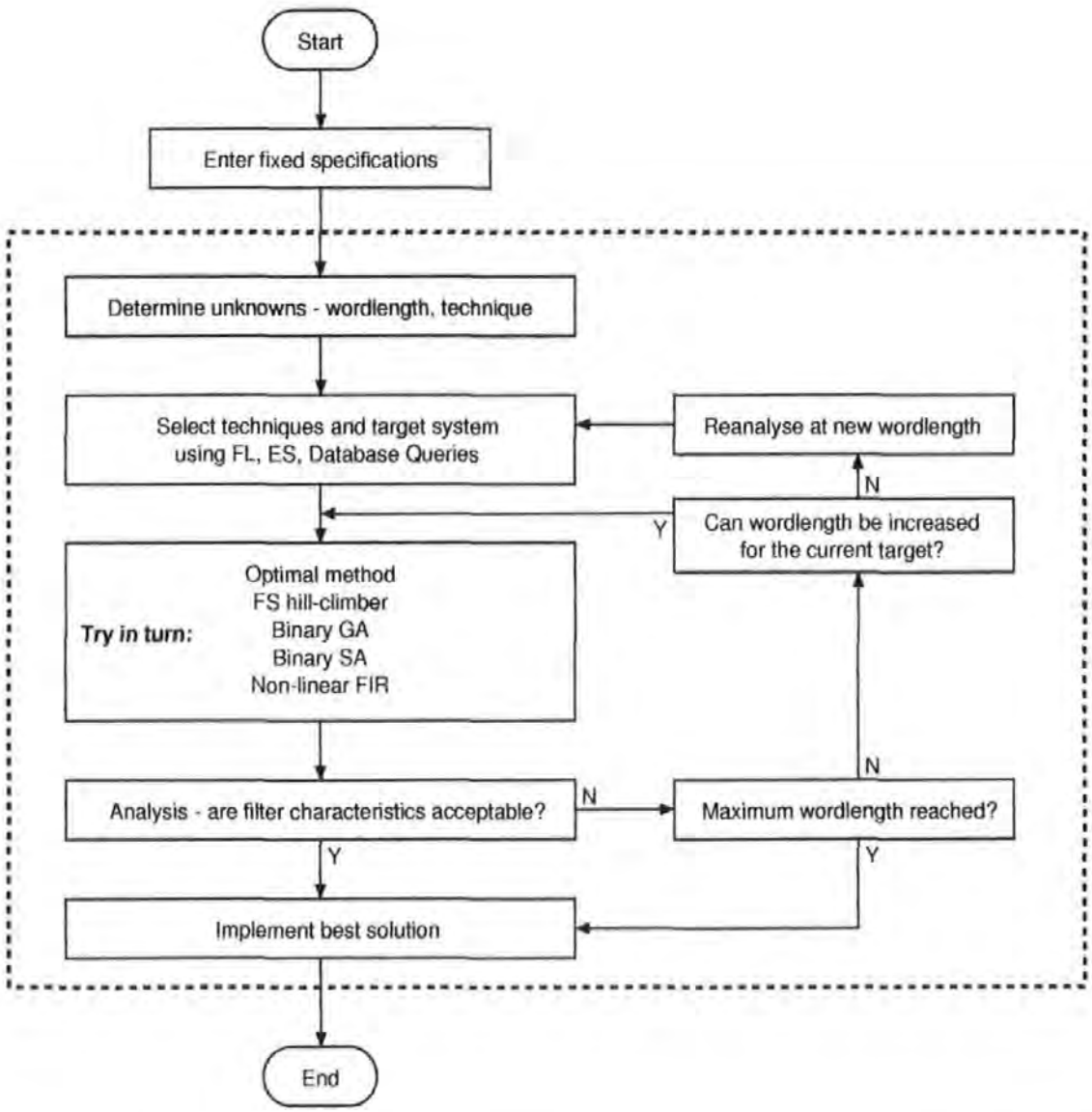


Figure 9.2: Illustration of the potential use of the intelligent design tool, to design an FIR filter as described in the text.

and design method are variables, and therefore uses an expert system to search its database to determine which methods are suitable for the problem, and grades them by their predicted usefulness, perhaps using a fuzzy logic analysis. In this case, it might perhaps select to try an Optimal method, FS hill-climber, a binary GA, SA, and as a last resort, try a non-linear FIR which may be acceptable if the response in the passband is close enough to linearity. As the wordlength is unspecified, the shortest possible wordlength is predicted and used for initial tests. This will be increased if no solution can be found.

The system tries the methods in turn, then analyses the results to determine which is the best result and if it is acceptable. If it is not, the wordlength is increased, and the methods run again. If, however, the maximum wordlength for the selected system has been reached, then the investigation into the best system to use has to be run again at the new, longer wordlength. Eventually, the system will produce the best possible solution, either within the design constraints or closest to them.

Future investigations into an intelligent design system of this sort could result in an extremely useful tool for digital filter designers, which is able to make use of a wide range of techniques, and to gain experience as it performs designs in order to increase its effectiveness in future. The concepts could also be expanded to produce tools for other areas of DSP design.

9.4 Conclusion

This project has investigated the digital filter design process, and has undertaken a range of optimisations using Genetic Algorithm and Simulated Annealing techniques. It has resulted in a number of GA-based optimisation tools, which have also provided insights into the nature of the search space, and therefore which filter optimisations will be suitable for the GA. An analysis of the filter design process has produced a framework for a wide-ranging, intelligent, adaptive design tool, which could automatically perform a range of optimisations in order to produce the best possible results.

Bibliography

[1] T. Bäck, U. Hammel, and H.-P. Schwefel. 'Evolutionary Computation: Comments on the History and Current State'. *IEEE Trans. on Evolutionary Computation*, 1(1):3–17, April 1997.

[2] E.C. Ifeachor and S.P. Harris. 'A New Approach to Frequency Sampling Filter Design'. *Procs. IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, 1993.

[3] S.P. Harris and E.C. Ifeachor. 'Automating IIR Filter Design by Genetic Algorithm'. *Procs. GALESIA International Conference*, 1995.

[4] S.P. Harris and E.C. Ifeachor. 'Automatic Design of Frequency Sampling Filters by Hybrid Genetic Algorithm Techniques'. *IEEE Transactions on Signal Processing*, 46(12):3304–3314, 1998.

[5] L.R. Rabiner, B. Gold, and C.A. McGonegal. 'An Approach to the Approximation Problem for Nonrecursive Digital Filters'. *IEEE Trans. Audio. Electroacoust.*, AU-18:83–106, June 1970.

[6] P.H. Kraght. 'A Linear Phase Digital Equalizer with Cubic-Spline Frequency Response'. *Journal of the Audio Engineering Society*, 40(5), May 1992.

[7] Y. Lian and Y.C. Lim. 'Linear Phase Digital Audio Tone Control Using Multiplication-Free FIR Filter'. *Journal of the Audio Engineering Society*, 41(10), 1993.

[8] N.J. Outram. *Intelligent Pattern Analysis of the Foetal Electrocardiogram*. PhD thesis, University of Plymouth, 1997.

[9] J.A. Van Alsté and T.S. Schilder. 'Removal of Baseline Wander and Powerline Interference from the ECG by an Efficient FIR Filter with a reduced Number of Taps'. *Journal IEEE Trans. on Biomedical Engineering*, 32(12):1052–1060, 1985.

[10] L.R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice-Hall, 1975.

[11] E.C. Ifeachor and B.W. Jervis. *Digital Signal Processing: A Practical Approach*. Addison-Wesley, 1993.

[12] L.R. Rabiner and R.W. Schafer. 'Recursive and Nonrecursive Realizations of Digital Filters Designed by Frequency Sampling Techniques'. *IEEE Transactions on Audio and Electroacoustics*, AU-19(3):200–207, September 1971.

[13] R.P. Ramachandran and S. Sunder. 'A Unified and Efficient Least-Squares Design of Linear-phase NonRecursive Filters'. *Elsevier Signal Processing*, 36:41–53, 1994.

[14] S.-C. Pei and J.-J. Shyu. 'Design of Arbitrary FIR Log Filters by Weighted Least Squares Technique'. *IEEE Trans. on Signal Processing*, 42(9):2495–2499, September 1994.

[15] J.H. McClellan, T.W. Parks, and L.R. Rabiner. 'A Computer Program for Designing Optimum FIR Linear Phase Digital Filters'. *IEEE Transactions on Audio and Electroacoustics*, AU-21 (December):506–526, 1973.

[16] D.J. Xu and M.L. Daley. 'Design of Finite Word Length FIR Digital Filter Using a Parallel Genetic Algorithm'. *Proc. IEEE Southeastern*, 2:834–837, 1992.

[17] D.J. Xu and M.L. Daley. 'Design of Optimal Digital Filter Using a Parallel Genetic Algorithm'. *Proc. IEEE Trans. on Circuits and Systems II*, 42(10):673–675, October 1995.

[18] T. Çiloğlu and Z. Ünver. 'A New Approach to Discrete Coefficient FIR Filter Design by Simulated Annealing'. *Procs. IEEE International Conference on Acoustics, Speech and Signal Processing*, 3:C101–C104, 1993.

[19] M.A. Keane, J.R. Koza, and J.P. Rice. 'Finding an Impulse Response Function Using Genetic Programming'. *Proc. 1993 American Control Conference*, III:2345–2350, 1993.

[20] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications, 2nd Ed.* Macmillan Publishing Company, 1992.

[21] S.J. Flockton and M.S. White. 'Pole-zero System Identification Using Genetic Algorithms'. *Procs. Fifth International Conference on Genetic Algorithms*, pages 531–535, July 1993.

[22] M.S. White and S.J. Flockton. 'Genetic Algorithms for Digital Signal Processing'. *Lecture Notes in Computer Science*, vol. 865:291–303, 1994.

[23] S. Sriranganathan, D.R. Bull, and D.W. Redmill. 'The Design of Low Complexity Two-Channel Lattice-Structure Perfect-Recombination Filter Banks Using Genetic Algorithms'. *Procs. 1997 IEEE Int. Symp. on Circuits and Systems*, pages 2393–2396, June 1997.

[24] K. Chellapilla, D.B. Fogel, and S.S. Rao. 'Gaining Insight into Evolutionary Programming Through Landscape Visualization: An Investigation into IIR Filtering'. In P.J. Angeline, R.G. Reynolds, J.R. McDonnell, and R. Eberhart, editors, *Evolutionary Programming VI*, pages 407–417, Berlin, 1997. Springer.

[25] D.M. Etter, M.J. Hicks, and K.H. Cho. 'Recursive Adaptive Filter Design using an Adaptive Genetic Algorithm'. *Int. Conf. on Acoustics, Speech and Signal Processing*, 2:635–638, 1982.

[26] A. Krukowski, I. Kale, and G. D. Cain. 'Decomposition of IIR Transfer Functions into Parallel Arbitrary-Order IIR Subfilters'. *Procs. NORSIG '96*, pages 175–178, 1996.

[27] S.S. Lawson. 'Direct Approach to Design of PCAS Filters with Combined Gain and Phase Specification'. *IEE Procs.-Vision, Image and Signal Processing*, 141(3):161–167, June 1994.

[28] A. Krukowski and I. Kale. Almost Linear-Phase Polyphase IIR Lowpass / Highpass Filter Approach'. *Procs. 5th International Symposium on Signal Processing and its Applications (ISSPA99)*, August 1999.

[29] S.S. Lawson and A. Wicks. 'Design of Efficient Digital Filters Satisfying Atbitrary Loss and Delay Specifications'. *IEE Procs.-G Circuits, Devices and Systems*, 139(5):611–620, October 1992.

[30] C.K. Lu, M.S. Anderson, and S. Summerfield. 'Approximately Linear-Phase Design of Allpass-Based QMF Banks'. *IEE Colloquium Digest, 16th Saraga Colloquium on Digital and Analogue Filters*, 238:11/1–11/6, 1996.

[31] T. Kobayashi and S. Imai. 'Design of IIR Digital Filters with Arbitrary Log Magnitude Function by WLS Techniques'. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 38(2):247–252, 1990.

[32] S. Chen, R. Istepanian, and B.L. Luk. 'Digital IIR Filter Design Using Adaptive Simulated Annealing'. *Digital Signal Processing*, 11(3):241–251, July 2001.

[33] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C (2nd Ed.)*. Cambridge University Press, 1992.

[34] L.R. Rabiner, N.Y. Graham, and H.D. Helms. 'Linear Programming Design of IIR Digital Filters with Arbitrary Magnitude Function'. *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-22:117–123, April 1974.

[35] Ping Wah Wong. 'Quantisation Noise, Fixed-Point Multiplicative Roundoff Noise, and Dithering'. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 38(2):286–300, February 1990.

[36] R.J. Clark, E.C. Ifeachor, and G.M. Rogers. 'The Study of Arithmetic and Wordlength Requirements for Digital Audio Filtering Hardware'. *Procs. 99th AES Convention, New York*, October 1995.

[37] B. Liu and A. Peled. 'Heuristic Optimisation of the Cascade Realisation of Fixed-Point Digital Filters'. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-23(5):464–473, 1975.

[38] J. Dattorro. 'The Implementation of Recursive Digital Filters for High-Fidelity Audio'. *Journal of the Audio Engineering Society*, 36(11):851–878, 1988.

[39] R. Wilson. 'Filter Topologies'. *Journal of the Audio Engineering Society*, 41(9):667–678, 1993.

[40] A. Roberts and G. Wade. 'A Structured GA for FIR Filter Design'. *IEE and IEEE Workshop on Natural Algorithms in Signal Processing*, November 1993.

[41] D. Suckley. 'Genetic Algorithm in the Design of FIR Filters'. *IEEE Proceedings-G*, 138:234–238, April 1991.

[42] K. Uesaka and M. Kawamata. 'Synthesis of Low-Sensitivity Second-Order Digital Filters Using Genetic Programming with Automatically Defined Functions'. *Procs. IEEE ISCAS 2000 Vol 1: Emerging Technologies for the 21st Century*, 1:359–362, 2000.

[43] J.D. Schaffer and L.J. Eshelman. 'Designing Multiplierless Digital Filters using Genetic Algorithms'. *Procs. Fifth International Conference on Genetic Algorithms*, pages 439–444, July 1993.

[44] A.G. Dempster and M.D. Macleod. 'Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters'. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, 42(9):569=–577, September 1995.

[45] D.W. Redmill and D.R. Bull. 'Automated Design of Low Complexity FIR filters'. *Procs. 1998 Int. Symp. on Circuits and Systems*, pages D429–D432, 1998.

[46] D.W. Redmill, D.R. Bull, and E. Dagless. 'Genetic Synthesis of Reduced Complexity Filters and Filter Banks Using Primitive Operator Directed Graphs'. *IEE Procs.: Circuits, Devices and Systems*, 147(5):303–310, October 2000.

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

[48] D.E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison Wesley, 1989.

[49] Z. Michalewicz and C.Z. Janikow. 'Handling Constraints in Genetic Algorithms'. *Procs. Fourth Int. Conf. on Genetic Algorithms*, pages 151–157, 1991.

[50] P. Clitherow and G. Fisher. 'Knowledge Based Assistance of Genetic Search in Large Design Spaces'. *Procs. 2nd Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 729–734, 1989.

[51] D. Goldberg and K. Deb. 'A Comparative Analysis of Selection Schemes used in Genetic Algorithms'. *Foundations of Genetic Algorithms*, pages 69–93, 1991.

[52] R. Roy and I.C. Parmee. 'Adaptive Restricted Tournament Selection for the Identification of Multiple Sub-Optima in a Multi-Modal Function'. *Procs. AISB Workshop on Evolutionary Computation*, April 1996.

[53] H. Ishibuchi and T. Murata. 'Multi-Objective Genetic Local Search Algorithm'. *Procs. 1996 IEEE Int. Conf. on Evolutionary Computation*, pages 119–124, 1996.

[54] J. Suzuki. 'A Markov Chain Analysis on Simple Genetic Algorithms'. *IEEE Transactions on Systems, Man and Cybernetics*, 25(4):655–659, 1995.

[55] J. Horn. 'Finite Markov Chain Analysis of Genetic Algorithms with Niching'. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 110–117, 1993.

[56] D.E. Goldberg. 'Real Coded Genetic Algorithms, Virtual Alphabets and Blocking'. *Complex Systems*, 5:139–167, 1991.

[57] H.M. Cartwright and S.P. Harris. 'Analysis of the Distribution of Airborne Pollution Using Genetic Algorithms'. *Atmospheric Environment*, 27A (12):1783–1791, 1993.

[58] P. Cong and T. Li. 'Numerical Genetic Algorithm Part 1: Theory, Algorithm and Simulated Experiments'. *Analytica Chimica Acta*, 293:191–203, 1994.

[59] C.Z. Janikow and Z. Michalewicz. 'A Specialised Genetic Algorithm for Numerical Optimisation Problems'. *Second Int. IEEE Conf. on Tools for A.I. Proc.*, pages 798–804, 1990.

[60] R. Harris. 'An Alternative Description of the Action of Crossover'. *Adaptive Control in Engineering Design and Control '94, Proc.*, 1994.

[61] T. Dexter, E.D. Goodman, and W.F. Punch. 'The Genetic Algorithm and Local Optimizer Hybrid Approach for the Advanced Layout Problem'. *GARAGe Technical Report, Michigan State University, Feb 97.*

[62] J. Yen, J.C. Liao, D. Randolph, and B. Lee. 'A Hybrid Approach to Modeling Metabolic Systems Using Genetic Algorithms and the Simplex Method'. *Proceedings of the 11th IEEE Conference on Artificial Intelligence for Applications (CAIA95)*, pages 277–285, 1995.

[63] A. Krukowski and I. Kale. 'Two Approaches for Fixed-Point Filter Design, Bit-Flipping Algorithm and Constrained Downhill Simplex Method'. *Procs. 5th International Symposium on Signal Processing and its Applications (ISSPA99)*, 1999.

[64] I. Pitas. 'Optimisation and Adaptation of Discrete-Valued Digital Filter Parameters by Simulated Annealing'. *IEEE Trans. on Signal Processing*, 42:860–866, April 1994.

[65] L.M. Smith and M.E. Henderson. 'Roundoff Noise Reduction in Cascade Realizations of FIR Digital Filters'. *IEEE Transactions on Signal Processing*, 48(4):1196–1200, 2000.

[66] R. Storn. 'Differential Evolution Design of an IIR Filter'. *Procs. of the 1996 IEEE Int. Conf. on Evolutionary Computation*, pages 268–273, 1996.

[67] O. Franzen, H. Blume, and H. Schröder. 'FIR-Filter Design with Spatial and Frequency Design Constraints using Evolution Strategies'. *Elsevier Signal Processing*, 68(3):295–306, August 1998.

[68] J.R. Koza. 'Genetic Programming as a Means For Programming Computers by Natural Selection'. *Statistics and Computing*, 4:87–112, 1994.

[69] K. Rodríguez-Vázquez, C.M. Fonseca, and P.J. Fleming. 'Multiobjective Genetic Programming : A Nonlinear System Identification Application'. *Late Breaking Papers at the Genetic Programming 1997 Conference*, pages 207–212, July 1997.

[70] F. Glover. 'Tabu Search - Part I'. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[71] F. Glover. 'Tabu Search and Finite Convergence'. *Accepted for publication in: Discrete Applied Mathematics: Special Edition on Foundations of Heuristics in Combinatorial Optimisation.*

[72] S. Traferro and A. Uncini. 'Power-of-two Adaptive Filters Using Tabu Search'. *IEEE Trans. on Circuits and Systems II: Analogue and Digital Signal Processing*, 47(6):566–569, June 2000.

[73] R. Battiti and G. Tecchiolli. 'The Continuous Reactive Tabu Search: Blending Combinatorial Optimization and Stochastic Search for Global Optimization'. *Annals of Operations Research – Metaheuristics in Combinatorial Optimization*, 63:153–188, 1996.

[74] M.P. Hansen. 'Tabu Search for Multiobjective Optimization: MOTS'. *Procs. MCDM 1997*, January 1997.

[75] L. Davis. 'Adapting Operator Probabilities in Genetic Algorithms'. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, 1989.

[76] V.B. Lawrence and A.C. Salazar. 'Finite Precision Design of Linear-Phase FIR Filters'. *The Bell System Technical Journal*, 59(9):1575–1598, 1980.

[77] P.A. Stubberud and C.T. Leondes. 'A Frequency Sampling Filter Design Method which Accounts for Finite Wordlength Effects'. *IEEE Trans. on Signal Processing*, 42:189–193, January 1994.

[78] T. Arslan and D.H. Horrocks. 'A Genetic Algorithm for the Design of Finite Word Length Arbitrary REsponse Cascaded IIR Digital Filters'. *Procs. GALESIA International Conference*, pages 276–281, 1995.

[79] Y. Aketa, M. Haseyama, H. Kitajima, and N. Nagai. 'A Method for Quantising Coefficients of a Filter with Genetic Algorithm'. *Electronics and Communications in Japan Pt III—Fundamental Electronic Science*, 79(4):1–10, 1996.

[80] T. Görne and M. Schneider. 'Design of Digital Filters with Evolutionary Algorithms'. *Artificial Neural Nets and Genetic Algorithms*, Albrecht, Reeves and Steele Eds.:368–374, November 1993.

[81] C.M. Fonseca and P.J. Fleming. 'Genetic Algorithms for Multiobjective Optimisation: Formulation, Discussion and Generalisation'. *Procs. 5th Int. Conf. on Genetic Algorithms*, pages 416–423, 1993.

[82] L.J. Nicolson and B.M.G. Cheetham. 'An Investigation into the Multiple Criterion Optimisation Approach to IIR Digital Filter Design'. *Colloquium on Digital and Analogue Filters and Filtering Systems*, pages 4/1–4/6, 1992.

[83] J.D. Schaffer. 'Multiple Objective Optimisation with Vector Evaluated Genetic Algorithms'. *International Conference on Genetic Algorithms 1985, Proc.*, pages 93–100, 1985.

[84] C. Fonseca and P. Fleming. 'A Review of Current Multi-objective Optimisation Methods'. *Evolutionary Computation*, 3(1):1–16, 1995.

[85] S.W. Mahfoud. 'A Comparison of Parallel and Sequential Niching Methods'. *Procs. Sixth Int. Conf. on Genetic Algorithms*, pages 136–143, 1995.

[86] D. Beasley, D.R. Bull, and R.R. Martin. 'A Sequential Niche Technique for Multimodal Function Optimisation'. *Evolutionary Computation*, 1(2):101–125, 1993.

[87] J. Horn, N. Nafpliotis, and D.E. Goldberg. 'A Niched Pareto Genetic Algorithm for Multiobjective Optimisation'. *Procs. First IEEE Conf. on Evolutionary Computation*, 1:82–87, 1994.

[88] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[89] I. Selesnick and C. Burrus. 'Maximally Flat Low-pass FIR Filters with Reduced Delay'. *IEEE Trans. on Circuits and Systems II*, 45(1):53–68, 1998.

[90] K.S. Tang, K.F. Man, S. Kwong, and Z.F. Liu. 'Design and Optimisation of IIR Filter Structure Using Hierarchical Genetic Algorithms'. *IEEE Trans. on Industrial Electronics*, 45(3):481–487, June 1998.

[91] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[92] J. Horn and D.E. Goldberg. 'Genetic Algorithm Difficulty and the Modality of Fitness Landscapes'. *Foundations of Genetic Algorithms 3*, pages 243–269, 1995.

[93] Y. Davidor. 'Epistasis Variance: A Viewpoint on GA-Hardness'. *Foundations of Genetic Algorithms*, pages 23–35, 1990.

[94] T. Jones and S. Forrest. 'Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms'. *ICGA 6*, 1995.

[95] I. Kale, G.D. Cain, and R.C.S. Morling. 'Minimum-Phase Filter Design from Linear-Phase Startpoint via Balanced Model Truncation'. *Electronics Letters*, 31(20):1728–1729, September 1995.

[96] D. Beasley, D.R. Bull, and R.R. Martin. 'Reducing Epistasis in Combinatorial Problems by Expansive Coding'. *Procs. Fifth Int. Conf. on Genetic Algorithms*, pages 400–407, 1993.

[97] H-C. Lu and S-T. Tzeng. 'Complex Genetic Algorithm Approach for Designing Equiripple Complex FIR Digital Filters with Weighting Functions'. *Signal Processing*, 80:197–204, 2000.

[98] S-T. Tzeng and H-C. Lu. 'Design of Arbitrary FIR Log Filters by Genetic Algorithm Approach'. *Signal Processing*, 80:497–505, 2000.

[99] U. Zölzer. 'Roundoff Error Analysis of Digital Filters'. *Journal of the Audio Engineering Society*, 42(4):232–244, 1994.

[100] C. Cotta and J.M. Troya. 'Tackling Epistatic Problems Using Dynastically Optimal Recombination'. *Computational Intelligence: Theory and Applications, International Conference*, pages 197–205, 1999.

Appendix A

Techniques

A.1 Increased calculation efficiency for recursive FIR filters

When generating the response of a recursive Frequency Sampling (FS) filter, it is possible to increase the computational efficiency by pre-calculating the fixed part of the response, namely the passband, as only the effects of the transition samples will vary from filter to filter, depending on the values contained within the GA's chromosome. The transfer function for a recursive filter can be given by [11]

$$H(z) = \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H(k)}{1 - e^{j2\pi k/N} z^{-1}}$$

This can be split into a comb filter which has N zeros spaced equally around the unit circle:

$$H_c(z) \frac{1 - z^{-N}}{N} \tag{A.1}$$

and a sum of single all-pole filters, whose poles are coincident with the zeros of the comb filter:

$$H_p(z) = \sum_{k=0}^{N-1} \frac{H(k)}{1 - e^{j2\pi k/N} z^{-1}} \tag{A.2}$$

The effects of the comb filter $H_c(z)$ are clearly constant for a given z, so can be pre-calculated. Similarly, the coefficients in $H(k)$ have no effect on $H_p(z)$ in the stopband, where they are always zero, so these samples need not be included. $H(k)$ is also

constant in the passband where the samples are always unity, so the effects of these coefficients on $H_p(z)$ can be precalculated and stored. This leaves only the effects of the few, variable transition samples to be calculated each time, and this can be made more efficient by precalculating the denominator $1 - e^{j2\pi k/N} z^{-1}$ as this is constant for each k. Finally, only the response in the stopband needs to be determined as the fitness only depends on the stopband attenuation. These precalculations and optimisations produced a noticeable reduction in the GA run times.

A.2 Simplex method hill-climber

The Simplex Method of hill-climbing involves manipulating an $N + 1$-vertex solid in N-dimensional space in order to search for improved solutions [33]. The simplest method of hill-climbing is to optimise each coefficient in turn, but this is inefficient when negotiating narrow valleys in the search space, as the search is required to zig-zag as it can only make a small step at a time along each axis. Although it is possible to use gradient information to rotate the axis so that they lie along the valley, and the search can be more efficient, this requires additional calculations and a regular realignment of the axes.

The Simplex method does not require any such analyses, and only takes point samples of the space, but has the usual hill-climber's limitation of only finding the optimum it is started nearest to. An initial set of $N + 1$ vertices is selected at random, and their fitnesses determined. The worst point in the shape is now moved relative to the best face in the shape in order to search for a better solution, as shown in Figure A.1.

The different moves listed in Figure A.1 are tried in turn: first a reflection, and if this produces an improvement, the reflection is extended in the same direction to see if there is any further benefit. If the first reflection did not improve the worst point, then a contraction of this point towards the best face is tried. If all of these fail, then all of the worst points contract towards the best one. The termination criterion is that the relative fitnesses of all points within the Simplex lie within a selected tolerance band,

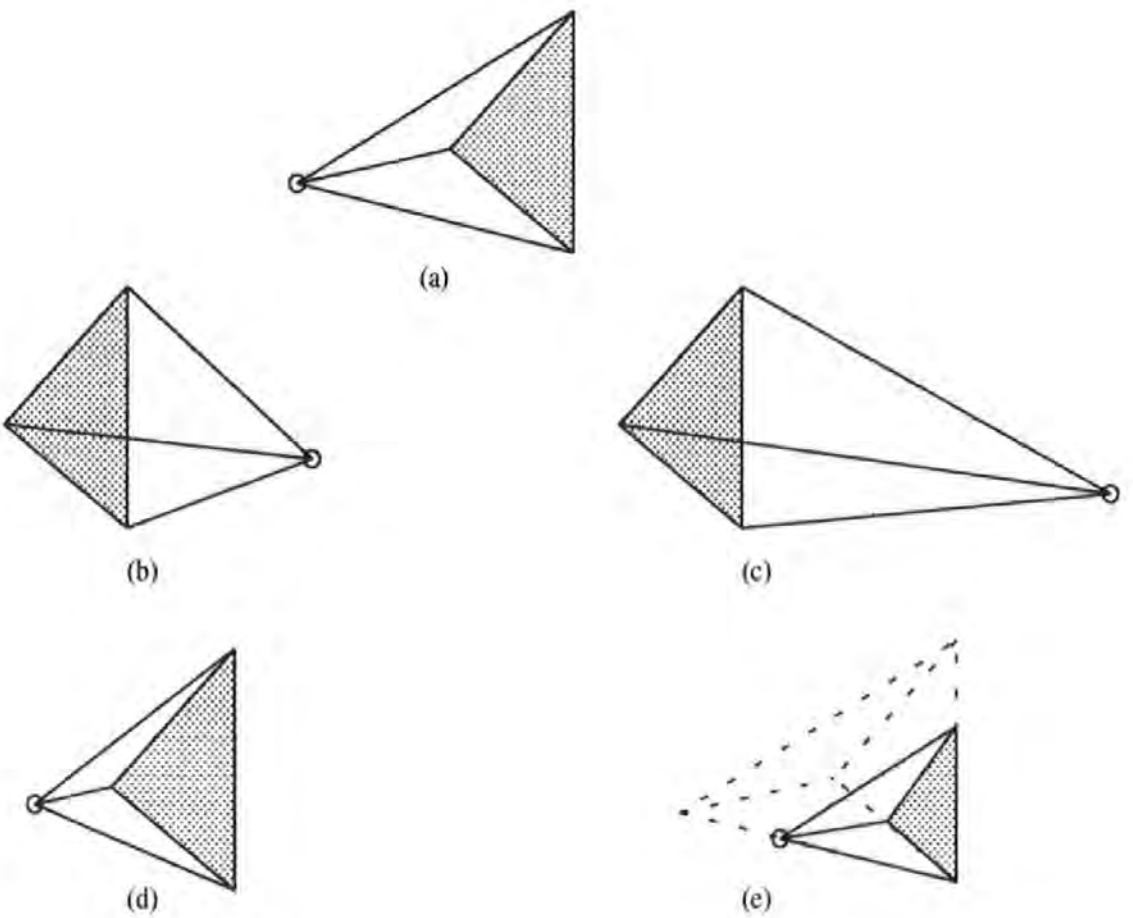


Figure A.1: Steps taken by a Simplex method hill-climber while looking for a better solution, after *Numerical Recipes in C* [33]. (a) is the Simplex at the start; the best face and worst vertex are marked. Possible outcomes are (b) a reflection through the best face, (c) a reflection and extension, (d) a contraction towards the best face, or (e) a contraction towards the best point.

which must be determined by experiment. In this work, a relative range of 0.001 was used.

A.3 Matlab initialisation script for MCO IIR design

This is the initialisation script used within Matlab to predict the necessary filter order to achieve a given target response. It was used to generate pole-zero positions to initialise the multi-criterion GA used to optimise IIR filters in Chapter 8.

The parameter `N` was either set to the desired order, or to zero to use the order predicted by Matlab. `Wp` and `Ws` are the normalised pass- and stopband edge frequencies, and `Rp` and `Rs` are the desired maximum passband ripple and minimum stopband attenuation, both in dB. `name` is the basename for the datafiles. `wl` is the desired wordlength. `Wlinphlo` and `Wlinphhi` are the normalised edges of a desired linear phase region. `lohi` is a text string, set to either `low` or `high` to specify low- or highpass. `fwl` specifies how many of the `wl` bits are used to specify fractional numbers.

```
function lhpiir(N,Wp,Ws,Rp,Rs,name,wl,Wlinphlo,Wlinphhi,lohi,fwl)

sprintf('function lhpiir(N,Wp,Ws,Rp,Rs,''name'',wl,
                                    Wlinphlo,Wlinphhi,''type'',fwl)\n');
sprintf('This uses Nyquist frequency=0.5!\n');
sprintf('Set N=0 to use the predicted order.\n');

if strcmp(lohi,'high')              Setup variables according to low- or highpass
  high=1;
  Wslow=0;
  Wshigh=Wp;
  Wplow=Ws;
  Wphigh=0.5;
elseif strcmp(lohi,'low')
  high=0;
  Wplow=0;
  Wphigh=Wp;
  Wslow=Ws;
  Wshigh=0.5;
else
```

```
  ['Incorrect parameter: ' lohi]
  return;
end

Wp=Wp*2;                                  Convert bandedges to F_s = 1 for Matlab
Ws=Ws*2;

[n,Wn]=ellipord(Wp,Ws,Rp,Rs)                          Predict the order needed

if N>0                                                Use a specified order if given
  n=N;
end

if n-floor(n/2)*2 ~= 0                                Make order even if necessary
  n=n+1;
  sprintf('n odd - increased to %i\n',n)
end

if high==1                                            Get filter pole-zero positions
  Wn = Ws;
  [z p k] = ellip(n,Rp,Rs,Wn,'high');
else
  Wn = Wp;
  [z p k] = ellip(n,Rp,Rs,Wn);
end

figure(1);
zplane(z,p);                              Display pole-zero locations in polar plot

fop=fopen([name '.pz'],'w');              Save pole-zero locations to initialise GA

for s=1:2:n
  absz((s+1)/2)=abs(z(s));
  argz((s+1)/2)=angle(z(s));
  absp((s+1)/2)=abs(p(s));
  argp((s+1)/2)=angle(p(s));
  fprintf(fop,'absz= %g\n',absz((s+1)/2));
  fprintf(fop,'argz= %g\n',argz((s+1)/2));
  fprintf(fop,'absp= %g\n',absp((s+1)/2));
  fprintf(fop,'argp= %g\n',argp((s+1)/2));
end
fclose(fop);

absz                                          Print pole-zero radii and angles
argz
absp
argp
```

```
if high==1                                          Get filter a and b coefficients
  [b a] = ellip(n,Rp,Rs,Wn,'high');
else
  [b a] = ellip(n,Rp,Rs,Wn);
end

[h w] = freqz(b,a,128);                             Get frequency response

figure(2);
subplot(2,1,1);
plot(w/(2*pi), 20*log(abs(h)) );                    Display magnitude response
subplot(2,1,2);
plot(w/(2*pi), angle(h) );                          Display phase response

% save the target .tgt file

fop=fopen([name '.tgt'],'w');

if high==1                                                    If highpass...
  temp = Ws;  Ws = Wp;  Wp = temp;

  for s=0:Ws*512                    Save the response template for the stopband...
    fprintf(fop,'-200  %g\n',-Rs);
  end

  for s=Ws*512+1:Wp*512                                    ...transition band...
    fprintf(fop,'-200  0\n');
  end

  for s=Wp*512+1:512                                          ...and passband
    fprintf(fop,'%g  0\n',-Rp);
  end

  temp = Ws;  Ws = Wp;  Wp = temp;
else                                                        ...else if lowpass...
  for s=0:Wp*512                    Save the response template for the passband...
    fprintf(fop,'%g  0\n',-Rp);
  end

  for s=Wp*512+1:Ws*512                                    ...transition band...
    fprintf(fop,'-200  0\n');
  end

  for s=Ws*512+1:512                                          ...and stopband
    fprintf(fop,'-200  %g\n',-Rs);
  end
```

```
end

fclose(fop);

% now save the .ini file                    Save the GA initialisation file
fop=fopen([name '.ini'],'w');
fprintf(fop,'# Ini file created automatically by Matlab\n');

fprintf(fop,['# lhpiir(%i,%g,%g,%g,%g,''' name ''',%i,%g,%g,
    ''' lohi ''',%g)\n\n'],N,Wp/2,Ws/2,Rp,Rs,wl,Wlinphlo,Wlinphhi,fwl);

fprintf(fop,'nstages:\t\t%i\n',n/2);
fprintf(fop,['templatefile:\t\t' name '.tgt\n']);
fprintf(fop,['datafileroot:\t\t' name '\n']);
fprintf(fop,'wordlength:\t\t%i\n',wl);
fprintf(fop,'fractional_bits:\t\t%i\n',fwl);
fprintf(fop,'perturbpc:\t\t5\n');
fprintf(fop,'passband:\t\t%g %g %g %g\n',Wplow,Wphigh,-1,-1);
fprintf(fop,'stopband:\t\t%g %g %g %g\n',Wslow,Wshigh,-1,-1);
fprintf(fop,'linearphaseregion:\t%g %g %g %g\n',Wlinphlo,Wlinphhi,
                                                        -1,-1);
fprintf(fop,'maxmagerror:\t40\n');
fclose(fop);

% now save the 'last.m' file          Save a Matlab script to recreate this data
fop=fopen('last.m','w');

fprintf(fop,['lhpiir(%i,%g,%g,%g,%g,''' name ''',%i,%g,%g,
    ''' lohi ''',%g)\n\n'],N,Wp/2,Ws/2,Rp,Rs,wl,Wlinphlo,Wlinphhi,fwl);
fclose(fop);

return;
```

This function was used to create datafiles containing pole-zero positions of full-precision filters. These were then used to initialise the GA, by mutating each bit by an amount proportional to perturbpc.

A.4 Single-criterion Genetic Algorithm

This C++-style pseudocode shows the important parts of the GA used to design Frequency Sampling (FS) FIR filters.

```
void GA::Generate()                                    The main loop
{
  HandleKeyPresses();                                  Pause, Save data, Quit

  generation++;                                        Increment generation counter

  MakeNewPopulation();                                 Generate a child population...

  GetFitnesses();                                      ... and get their fitnesses

  SelectNextPopulation();                              Select the next population

                        After a quarter of the run, if no improvement for 20 generations...
  if (( generation > maxGenerations / 4 ) &&
                                              ( noImprovementCount >= 20 ))
  {
    LocalSearch();                                     ... call the Simplex local search

                          If it found a better fitness, reset the "no improvement" counter
    if ( bestFitnessImproved )
      noImprovementCount = 0;
  }

  if ( bestFitnessImproved )
    noImprovementCount = 0;
  else
    noImprovementCount++;

  CopyNewPopulationToOld();

  if ( TerminationConditionsMet() || generation == maxGenerations )
  {
    SaveData();
    Quit();
  }
}


void GA::MakeNewPopulation()
{
                              Use stochastic remainder selection to pick individuals to
                                    reproduce, the selected ones are listed in chosen[]

                                  Selection according to integer part of scaled fitness

  count=0;
```

```
for ( int m=0; m<popsize; m++ )
{
  while ( scaledFitness[m] >= 1.0 )
  {
    chosen[++count] = m;
    scaledFitness[m] -= 1.0;
  }
}
```

Selection according to fractional part of scaled fitness

```
while ( count < popsize-1 )
{
  m = randomInt(0,popsize-1);
  if (scaledFitness[m] > 0.0)
  {
    if ( randomFloat(0,1) < scaledFitness[m] )
    {
      chosen[++count] = m;
      fitscaled[m] = 0.0;
    }
  }
}
```

Reproduce pairs of strings selected at random from those held in chosen[]

```
do
{
  use  = randomInt(0,numleft-1);          Pick two strings at random
  old1 = chosen[use];
  chosen[use] = chosen[numleft--];
  use  = rndi(numleft);
  old2 = chosen[use];
  chosen[use] = chosen[numleft--];

  crosstype = SelectCrossoverType();      Select crossover type by efficiency

  switch ( crosstype )
  {
    case 0: NoXover(); break;    Simply copy the old strings without crossover
    case 1: xover(); break;
    case 2: multixover(); break;
    case 3: arithxover(); break;
    case 4: multiarithxover(); break;
    case 5: wholearithxover(); break;
  }
```

```
    Mutate();                          Mutate the genes in the two new chromosomes

    GetNewFitnesses();                 Get the fitnesses of the two new chromosomes
    UpdataCrossoverEfficiency();       Update the crossover efficiency information

  } while ( newPopulationIncomplete() );
}


                                                          Now the five types of crossover

void GA::xover()                                          Single-point crossover
{
  crossAt = randomInt(0,genes-1);

  CopyOldToNew(0,CrossAt);
  CrossOldToNew(CrossAt+1,genes-1);
}


void GA::multixover()                                     Multi-point crossover
{
  direction = 1;

  for ( g=0; g<genes; g++ )
  {
    if ( direction == 1)
      CopyOldToNew(g);
    else
      CrossOldToNew(g);

    if ( randomInt(0,genes-1) < genes / 2. )    Give an average of 2-pt xover
      direction = 1 - direction;
  }
}


void GA::arithxover()                                     Single Arithmetic crossover
{
  crossAt = randomInt( 0, genes-1 );

  CopyOldToNew(0,genes-1);

  ArithmeticCrossover(crossAt);
}
```

```
void GA::multiarithxover()                    Multiple Arithmetic crossover
{
  for ( g=0; g<genes; g++ )
  {
    if ( randomFloat(0,1) < 0.5 )
      ArithmeticCrossover( g );
    else
      CopyOldToNew( g );
  }
}


void GA::wholearithxover()                    Whole Arithmetic crossover
{
  ArithmeticCrossover( 0, genes-1 );
}

                                              Function used to return a fitness
double GA::LowpassFitness( double *genes )
{
  BubbleSort( genes );      Sort transition samples so they decrease monotonically

            Scaling factor to convert sample numbers to interpolated sample numbers
  interpolatedSamples = 512;
  scale = samples / interpolatedSamples;

                                        Scan over the interpolated stopband samples
  for ( w = stopbandEdgeSample * scale; w<interpolatedSamples; w++)
  {
            The effects of the passband samples are precalculated as in Appendix A.1
    H = precalculatedH[w];

     Add the effects of the transition band samples - the factors are also precalculated
    for ( k=0; k<NTransitionBandSamples; k++ )
      H += genes[k] * precalculatedMultipliers[w][k];

    output[w] myabs(H);
  }

  deltas = GetMaximum( output[w] );           Find the maximum ripple

  deltas = -20.0 * log10(deltas);             Convert the ripple to decibels

  return temp;
}
```

A.5 Multi-criterion Genetic Algorithm

This C++-style pseudocode shows the important parts of the MCO GA used to design IIR filters.

```
void GA::Generate()                                          The main loop
{
  while ( generation++ < maxGeneration )
  {
    GAProcessKey( );                                         Handle keyboard input

    GAMakeNewPopulation();                                   Generate a new population

    GAGetFitnesses();                                        Calculate their fitnesses

    if ( (generation & 127) == 127 )                         Occasionally call a local search
      BitFlipLocalSearch();

    Select those in the old and new non-dominated sets, plus other poorer solutions
    GASelectNextPop();

    GANewToOld();                                            Copy the new population over the old

  }
}


void GA::GAMakeNewPopulation()                               Generate the next population
{
                    Pick individuals by tournament selection on their shared fitnesses
  SelectForReproduction();

  while ( PopulationNotFilled() )
  {
    if ( randomFloat(0,1) < crossoverProbability )
      CrossOldToNew();                                       Perform crossover
    else
      CopyOldToNew();                                        No crossover
  }

  MutateNewPopulation( mutationProbability );
}
```

```
void GA::GAGetFitnesses()                          Calculate the fitnesses
{
  for ( m=0; m<popsize; m++ )
  {
    OneFitness( m );                               Get the fitness of member m
  }

  DetermineDomination();                           Find the NDS of the new members

                  Allocate fitness by NDS level and share by crowding to disperse search
  AllocateSharedFitness();
}


void Fitness::OneFitness()                 Determine the fitness of one chromosome
{
  DecodeChromosome();                  Extract radius, angle and topology information

  SetupFilter();                     Set up the filter with the extracted coefficients...
  GetFrequencyResponse();                                    ... and get its response

                                     How well does the frequency response fit the target?
  GetMagnitudeFitness();
                    How close to linearity is the phase response in the specified region?
  GetPhaseFitness();
  GetNoiseFitness();                                What level is the roundoff noise gain?
}
```

These fitnesses can then be used in any combination to determine the non-dominated sets, from which the fitnesses are allocated and the next population selected.

Appendix B

Publications

A NEW APPROACH TO FREQUENCY SAMPLING FILTER DESIGN USING GENETIC ALGORITHMS

E C Ifeachor and S P Harris
Department of Electronic, Communication and Electrical Engineering,
University of Plymouth,
Drake Circus, PLYMOUTH, PL4 8AA, England.

Abstract

The purpose of this paper is to present a novel approach to designing frequency sampling filters using Genetic Algorithms (GAs). In this method, an approximation to the desired continuous frequency response is obtained by optimising a small number of frequency samples. Existing methods employing linear programming techniques have computation times that increase exponentially with the number of samples to be optimised, and the published design tables do not adequately cover many filter designs. Our method overcomes these disadvantages, offers considerable flexibility, and yields results that are as good as or in some cases superior to published ones. Optimisation of the transition samples was achieved using a GA designed specifically for numerical problems requiring a high precision. A local search method was used in conjunction with the GA for fine tuning. Many aspects of filter design involve optimisation, and could easily be incorporated into the GA, thereby allowing it to provide a simple yet powerful 'universal' method for digital filter design, removing the difficulty of understanding the myriads of numerical optimisation techniques that are used in digital filter design.

1 Introduction

Central to digital filter design is the problem of finding a practical response that approximates a desired or ideal frequency response as closely as possible. The desired response may be magnitude, phase or both magnitude and phase. The frequency sampling method is an efficient way of finding the response of FIR (finite impulse response) filters to approximate an arbitrary frequency response. An attraction of the frequency sampling approach is that it allows a recursive implementation of FIR filters [1,2] which are computationally efficient, especially for filters with narrow passband. Further, the method is particularly well suited to the design of non standard filters where analytical expressions are not available.

In the frequency sampling method, samples of the desired frequency response are normally taken at equally spaced frequencies and from these an approximation to the desired continuous frequency response is obtained. To minimize the error between the desired and the computed responses a small number of the frequency samples are adjusted by an optimization procedure. Rabiner et al [3] describe an optimization technique, based on linear programming, for finding optimum frequency samples for standard frequency selective filters (e.g. lowpass and bandpass filters). Unfortunately, the computation time increases exponentially as the number of frequency samples to be optimized increases.

Tables of optimum values of the transition band frequency samples are available in the literature [3] and are widely used. If designer wants a filter not tabulated approximate values of the transition band frequency samples may be obtained by linear interpolation, but this is not always possible especially if the design involves a large number of transition band samples. Further, the information in the tables is not in a form filter designers are familiar with - e.g. bandedge frequencies and passband ripples are not given. The lack of a general purpose computer program for finding optimum frequency samples has restricted the use and eroded the value of the frequency sampling method.

Institution of Electrical Engineers.
published by the IEE, Savoy Place, London WC2R 0BL, UK

The purpose of this paper, is to present a novel approach to designing frequency sampling filters using Genetic Algorithms (GAs). GAs are basically search and optimization techniques based on the principles of natural selection and genetics [4,5] requiring little knowledge of the problem area. This makes them well suited to many engineering problems such as digital filter design, where optimization is required. Our method overcomes the disadvantages referred to above, offers considerable flexibility, and yields results that are as good as and in some cases superior to published ones.

Optimum values of the transition band frequency samples are normally of high precision, about 7 to 8 decimal places. In this work we used a specialized GA [6] which has been shown to be better suited to numerical optimization problems requiring high precision than standard GAs. The parameters are represented as floating numbers. To ensure that optimum values of the transition band frequency samples are obtained, the specialised GA is first used to find a good solution close to the optimum. A local search method is then used for fine tuning. For most filter design tasks, the solution found by the GA will be good enough, being close to the optimum solution.

Our algorithm runs on a 486 IBM PC (or compatible) and shows dynamically, via a graphics display, how the frequency response is changing as the GA searches the solution space. The software will be demonstrated at the workshop.

2. Frequency sampling method

In the basic frequency sampling method, samples of the desired frequency response are taken at regular intervals as illustrated in figure 1b for a lowpass filter. In this case, N samples of the ideal frequency response are taken at intervals of

$$\frac{kF_s}{N}, \qquad k=0, 1, 2, ..., N-1 \tag{1}$$

where F_s is the sampling frequency for the filter. Given the values of the N frequency samples, H(k), of the ideal frequency response, the FIR filter coefficients, h(n), are then obtained using the inverse discrete Fourier transform (IDFT):

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) e^{j(2\pi/N)nk}, \quad n = 0, 1, ..., N-1 \tag{2}$$

For linear phase filters with positive symmetrical impulse response, equation 2 can be written in a simpler form [2].

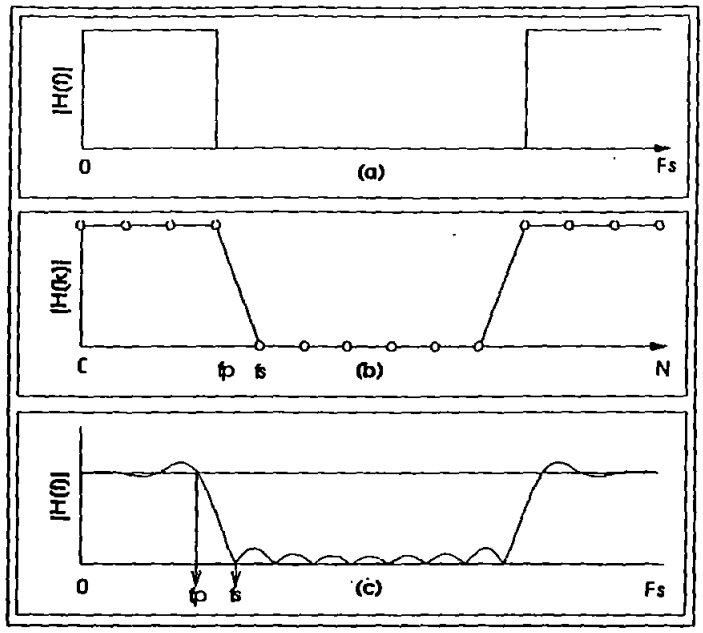


Figure 1: Concepts of frequency sampling. (a) The ideal response; (b) the sampled response; (c) the interpolated (actual) frequency response.

After computing the impulse response, h(n), the corresponding continuous frequency response may be obtained by first zero-padding h(n) and then taking its DFT. The continuous frequency response will be exactly the same as the desired response at the sampling instants, but between the sampling instants it may differ quite significantly, see figure 1c for example.

The frequency response of filters designed by the basic frequency sampling method will in general be poor, caused by the abrupt change in the values of the frequency samples from 1 (in the passband) to 0 in the stopband. To minimize the deviation of the response from the ideal response in the pass and stopband, we introduce frequency samples in the transition band as illustrated in Figure 2 for a simple lowpass filter with three transition band samples. The improvement in the pass and stopbands is achieved at the expense of increased transition width.

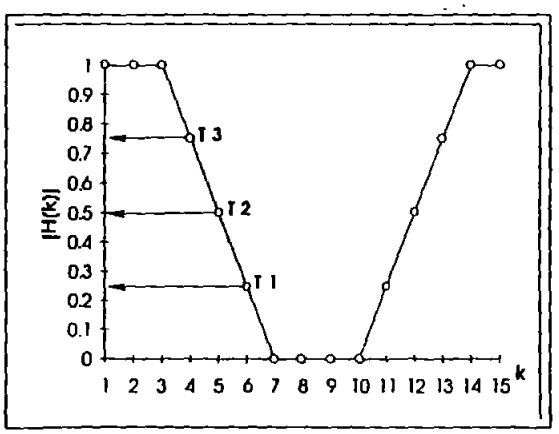


Figure 2: Frequency sampling with transition band frequency samples.

The values of the transition band frequency samples are not known in advance and must be found by optimization. A useful optimization criterion is to find the set of frequency samples, T_1, T_2, ..., T_M, that minimises the maximum ripple in the stopband.

$$\min\left[\max_{\{f \text{ in the stopband}\}} |H_D(f) - H(f)|\right] \qquad (3)$$

where

$H_D(f)$ = ideal frequency response

$H(f)$ = actual frequency response

We discuss in the next section how the GA is used to optimize the transition band samples. In the GA approach, the number of transition band samples does not have a major impact on the computational time, since in GA the search for a solution occurs in parallel. At the present we limit the number of transition band samples to 10.

3. Genetic Algorithm for optimizing transition band frequency samples

Optimization of the transition band samples was achieved using a specialized GA, which is well suited to numerical optimization problems which require high precision. As in standard GAs, we start with an initial population of possible solutions (i.e sets of frequency samples) generated at random. By applying the GA operators of reproduction, crossover and mutation at each generation of the algorithm. The initial population evolves towards the optimum set of frequency samples.

In our GA, a floating point representation is used to represent the frequency samples. For a filter with M transition band samples, each possible solution is represented as:

$$H_i = \{T_{1i}, T_{2i}, ..., T_{Mi}\} \qquad (4)$$

where the transition band samples, T_{1i}, T_{2i}...,T_{Mi}, are each represented as a double precision floating point number in the range 0 to 1.

The performance or fitness of each individual (i.e. each possible solution) is based on the maximum difference, δ_i, between the desired and actual frequency response:

$$\delta_i = \max_{\{f \text{ in the stopband}\}} |H_D(f) - H_i(f)| \qquad (5)$$

The fitness value for the *i*th individual is the inverse of the maximum difference.

As in most GA applications a substantial part of the optimization time is spent on computing the fitness values for the individuals. In our application, we estimate that about 80% is spent in evaluating the fitness function, mostly in computing the IDFTs and DFTs computations. Thus there is an incentive to seek ways to improve computational efficiency. Since for each generation only M frequency samples (in the transition band) will change, a large part of the IDFT can be precomputed and saved. Additional savings are made by using an algorithm which packs N real points into N/2 complex points, so halving the length of the FFT.

We have used forms of the GA operators designed specifically for floating-point optimisation problems. A dynamic mutation operator is used which moves the point under study by a random amount. The average distance that each point is moved is decreased as the number of generations increases, to prevent the GA from being thrown away from a fitness maximum once it has converged. The crossover operator

used was chosen randomly for each crossover from the following five crossover methods, with each being selected with equal probability. Ordinary crossover consists of a 1-point crossover, simply swapping blocks of numbers between chromosomes, and multiple crossover is similar but with a random number of crossing points. Arithmetic crossover consists of replacing the genes at the same location within two chromosomes by linear sums of the two original gene values. E.g. for vectors $\mathbf{x}$ and $\mathbf{y}$, to arithmetically cross over the kth gene:

$$\mathbf{x}' = r.\mathbf{x}_k + (1-r).\mathbf{y}_k$$
$$\mathbf{y}' = (1-r).\mathbf{x}_k + r.\mathbf{y}_k$$

where r is a random number between zero and one; if r=0.5, the resulting values are the average of the parent genes. Multiple arithmetic crossover is similar, but with a random number of genes being crossed, and whole arithmetic crossover affects the entire two chromosomes. It should be noted that the first two types of crossover do not actually create any new values, but simply move them between strings, whereas the last three variations actually alter the values of the genes, and so are closer to the actions of standard binary crossover. Inversion is not used because the genes are ordered before use, so it would not have any effect.

Stochastic remainder selection without replacement [5] is used to select members for reproduction, a technique that ensures that the best individuals are always chosen, while some poorer members are also picked to help maintain diversity and prevent premature convergence. Since there is only one maximum in the search space [3], there is no danger of the GA being trapped at a sub-optimal fitness peak, so elitism can safely be used to retain the best solutions found.

The genetic algorithm for optimizing the transition band samples are summarised below:

(1) From the user specifications determine the number of frequency samples in the pass and stop bands, and the number of transition bands samples.

(2) initialize the GA.

(3) compute the fitness value for each individual in the population and note the best individuals found so far.

(4) if the best individual is better than the best so far update the display of the response

(5) obtain the next generation: reproduction, crossover and mutation

(6) repeat steps 3 to 5 until the stopping condition is satisfied.

(7) save values of impulse response coefficients, frequency response, and values of the transition band samples for the best solution.

4 Results

We will illustrate the use of the algorithm by the following two design problems:

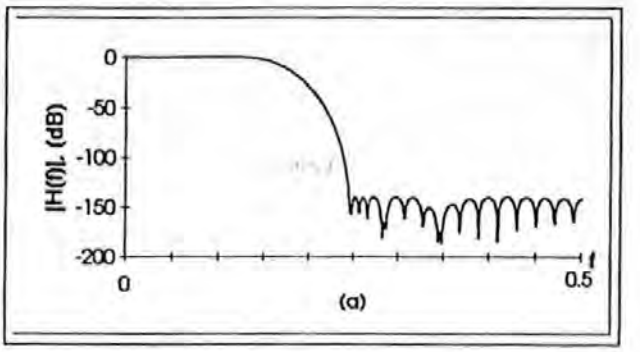
Problem 1:
Find the optimum transition band frequency samples and the corresponding filter coefficients for a lowpass filter meeting the following specifications:

passband edge frequency, f_p	0.143 (normalized)
stopband edge frequency, f_s	0.245 (normalized)
number of filter coefficients, N	49

From the specifications, the number of frequency samples, N = 49. The sample numbers corresponding to the pass band and stop band edge frequencies are 6 and 12 respectively. The number of transition band samples, M = 5. Thus the frequency samples for the ideal frequency response are given by:

$\|H(k)\|$	=	1,	k=0, 1, ... 6
		T_1	k=7
		T_2	k=8
		T_3	k=9
		T_4	k=10
		T_5	k=11
		0	k=12, 13, ... 24

The values of T_1 to T_5 are unspecified. The results of the optimization by GA are summarized in figure 3. The impulse response coefficients are put in the familiar form by circular rotation.



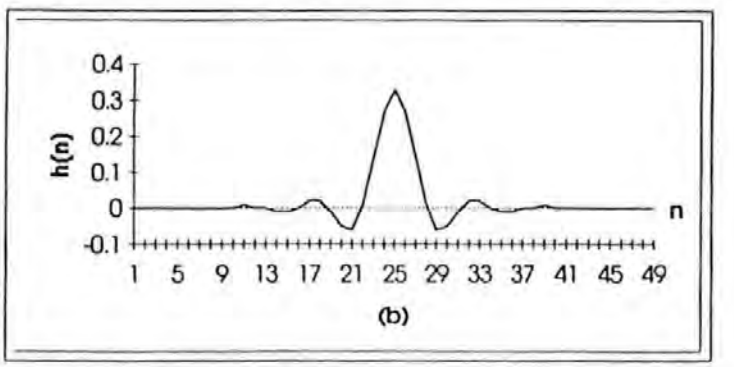


Figure 3: (a) The interpolated frequency response; (b) the filter coefficients.
passband ripple: 0.046dB; stopband attenuation: 139.64dB;
passband width: 0.15; 5 transition samples; 49 filter coefficients.
Transition sample values: 0.855456, 0.485507, 0.148961, 0.019693, 0.000644.

Problem 2:
Find the optimum transition band frequency samples and the corresponding filter coefficients for a bandpass filter meeting the following specifications:

stopband edge frequency, f_s	0.183 (normalized)
passband width	0.061 (normalized)
number of filter coefficients, N	49

From the specifications, the number of frequency samples, N = 49. The sample number corresponding to the stop band edge frequency is 9, and the pass band is 3 samples wide. The number of transition band samples, M = 5. The results of the optimization by GA are summarized in figure 4.

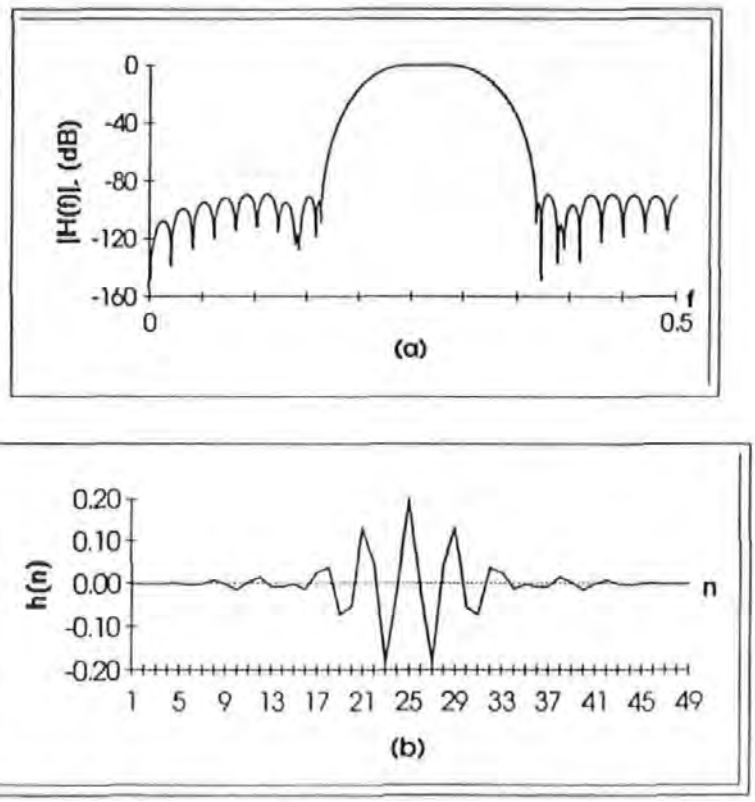


Figure 4: (a) The interpolated frequency response; (b) the filter coefficients.
passband ripple: 0.097dB; stopband attenuation: 89.08dB;
stopband width: 0.183; passband width: 0.061; 3 transition samples;
49 filter coefficients.
Transition sample values: 0.708873, 0.230358, 0.021169.

5 Discussions and conclusions

At the present, our algorithm can be used to design all four types of type-I FIR filters. Subsequently it will be developed to cope with non standard filters. We believe that this is relatively easy and should be completed well before the workshop.

We find the new technique described in this paper to be flexible and efficient. With the wide availability of PCs, we feel it should be possible to make the software, when it is fully developed, widely available to designers.

Many aspects of filter design, such as coefficient calculation and finite wordlength analysis, involve optimization. This makes digital filter design amenable to genetic algorithm solution. An attraction of GAs is that they require very little knowledge of the problem area. This may remove the difficulty of understanding the myriads of numerical optimization techniques that are used in filter design.

In the last four years, we have investigated the use of GA in filter design as part of our research in intelligent signal processing. Our work and that of others [7,8] indicates that GA could provide a simple, and yet powerful 'universal' method for digital filter design.

6 References

[1] Rabiner L R and Schafer R W, "Recursive and nonrecursive realizations of digital filters designed by frequency sampling technique." *IEEE Trans. Audio Electroacoustic*, Vol. 19, No3, Sept. 1971, pp200-207.

[2] Ifeachor E C and Jervis B W, "Digital signal processing: A practical approach." Addison-Wesley, Wokingham, UK, 1993.

[3] Rabiner L R, Gold B, and McGonegal C A, "An approach to the approximation problem for nonrecursive digital filters." *IEEE Trans Audio Electroacoustics*, Au-18, June 1970, pp83-106.

[4] Holland J, "Adaptation in natural and artificial systems." Ann Arbor. University of Michigan Press, 1975.

[5] Goldberg D E, "Genetic algorithms in search, optimization, and machine learning." Addison Wesley, 1989.

[6] Janikow C Z and Michalewicz Z, "A special genetic algorithm for numerical optimization problems." *Proc. Second Int. IEEE Conf. on Tools for AI*, Washington, Nov 1990, 798-804

[7] Suckley D, "Genetic algorithm in the design of FIR filters." *IEE proc.*, Vol 138, No 2, 1991, pp234-238.

[8] Xu D J and Daley M L, "Design of finite word length FIR Digital filter using a parallel genetic algorithm." *IEEE conference*, 1992, pp834-837.

Automating IIR Filter Design by Genetic Algorithm

Stephen P. Harris and
Emmanuel C. Ifeachor

University of Plymouth,
UK

e-mail steve@uk.ac.plym.cis

Abstract

The design of digital IIR filters is a multi-stage process, involving the optimisation of coefficient values, coefficient wordlengths, structure and section ordering. These are traditionally regarded as separate operations, and, as such, can in general only produce filters which are optimal in certain aspects, but not optimal overall. By exploiting the multiple criterion optimisation abilities of the Genetic Algorithm, we show that it is possible to perform several of these steps simultaneously. This allows the designer to specify the relative importance of each area of the design, for example, the frequency response or roundoff noise effects, thereby permitting the design of filters from a few initial specifications without requiring detailed knowledge of the individual design steps.

1 Introduction

The Genetic Algorithm (GA) [1, 2] is a relatively new search and optimisation technique, which takes its inspiration from evolution and natural selection. The GA is not only capable of searching multi-dimensional and multi-modal spaces, unlike hill-climbing, but is also able to optimise complex, discontinuous functions which are difficult to analyse mathematically. This makes it particularly suitable for optimising complex multi-objective functions.

To design a useful, practical filter it is necessary to perform several design steps in order to ensure that the filter has the desired performance in terms of frequency and/or phase response, and has acceptable noise characteristics due to quantisation effects. This multi-criterion problem is particularly suitable for optimisation by the GA [3, 4]. We have developed a GA-based FIR Frequency Sampling filter design package [5, 6], which was shown to be capable of optimising full-precision coefficient filters. This work is now being extended to cover the design of fixed-point IIR filters with quantised coefficients.

An IIR filter can be described by the following recursive expression:

$$y(n) = \sum_{k=0}^{N} a_k x(n-k) - \sum_{k=1}^{M} b_k y(n-k)$$

where a_k and b_k are the coefficients of the filter, $x(n)$ and $y(n)$ are its input and output, and N and M are the number of a_k and b_k filter coefficients, with $M \geq N$. This has an equivalent transfer function of:

$$H(z) = \frac{\sum_{k=0}^{N} a_k z^{-k}}{1 + \sum_{k=1}^{M} b_k z^{-k}}$$

An important task for the designer is to find values of a_k and b_k which produce the desired response. There are many approaches to this [7], producing filters with different characteristics.

A common way of realising IIR filters is to cascade several second-order sections together (Figure 1), the output from the first feeding the input of the next. This type of filter has a transfer function of:

$$H(z) = \prod_{k=1}^{N/2} \frac{a_{0k} + a_{1k} z^{-1} + a_{2k} z^{-2}}{1 + b_{1k} z^{-1} + b_{2k} z^{-2}}$$

or, in pole-zero form, assuming complex conjugate pole-zero pairs:

$$H(z) = \prod_{k=1}^{N/2} \frac{(z - r_o e^{j\omega_o})(z - r_o e^{-j\omega_o})}{(z - r_p e^{j\omega_p})(z - r_p e^{-j\omega_p})}$$

where r_o and r_p are the radii of the pole and zero respectively, and ω_o and ω_p are their angles. This can be represented in the shorter form:

Genetic Algorithms in Engineering Systems: Innovations and Applications
12-14 September 1995, Conference Publication No. 414, © IEE, 1995

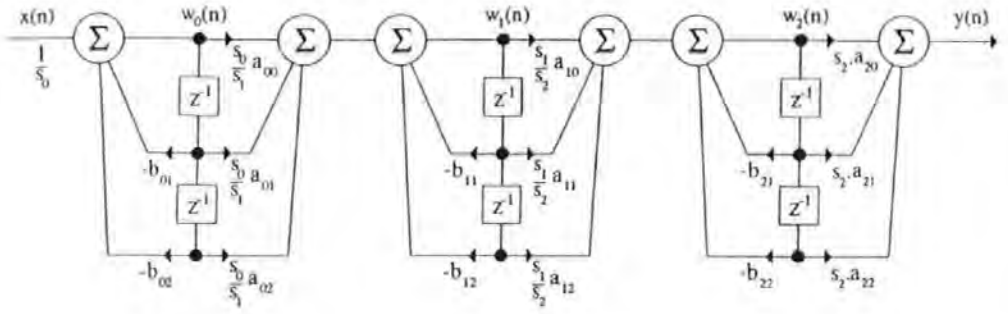


Figure 1. Cascaded second-order section IIR filter. The s_i scaling factors are to help minimise or prevent overflows.

$$H(z) = \prod_{k=1}^{N/2} \frac{N_k(z)}{D_k(z)}$$

Once suitable filter coefficients have been obtained, finite wordlength (FWL) analysis must be performed in order to determine how the filter will change when it is implemented in a real-world FWL system. When the filter coefficients a_k and b_k are quantised, this can have undesirable effects on the filter's behaviour, for example a change in the frequency response and alterations in the pole-zero positions which in the extreme case can lead to instability. Roundoff noise is also introduced by multiplications performed during the actual filtering.

Noise analysis shows that the ordering and pairing of the N_ks and D_ks also become a factor in the overall filter performance when the coefficients are quantised. Determining the best pairing and ordering of the N_k and D_k polynomials rapidly becomes non-trivial as the filter order increases, as the number of possible filters ϕ_N is given by:

$$\phi_N = \left[\left(\frac{N}{2}\right)!\right]^2$$

which for a tenth-order filter gives 14400 possibilities.

It is common practice to scale the filter coefficients to help minimise or prevent overflow. For example, in L2 norm scaling, the scaling factor for second-order section i is given by:

$$s_i = \left[\sum_{k=0}^{\infty} h_i^2(k)\right]^{\frac{1}{2}}$$

where $h_i(k)$ is the impulse response from the input to the internal node w_i for section i as shown in Figure 1.

For the quantised, scaled filter, the roundoff noise gain is given by:

$$\sigma_{or}^2 = \frac{1}{12}\left[3\sum_{k=0}^{\infty} f_1^2(k) + 5\sum_{k=0}^{\infty} f_2^2(k) + 5\sum_{k=0}^{\infty} f_3^2(k) + 3\right]$$

where $f_i(k)$ is the impulse response between the first adder in section i and the output [7].

The ordering and pairing of the N_k's and D_k's affect the overall roundoff noise for the filter, so they must also be optimised. In standard design methods, the optimisation of the filter coefficients and the minimising of the roundoff noise are two independent steps, so any filter so designed will be optimal in either one or other sense. What is needed is a design technique which allows several steps of filter design requiring optimisation to be performed in parallel, whereby we should be able to optimise both the filter coefficients and the roundoff noise simultaneously, trading one off against the other as necessary.

2 The GAs

As a first step, we developed a hybrid floating-point GA after Janikow and Michalewicz [8], which optimised the pole-zero positions for IIR filters made up from second-order sections. The GA operated with genes in the range 0–1, translated to a range of 0.5–1.0 for the radii, and 0–π for the angles. By using second-order sections which are made up from conjugate poles and zeros, the GA needed only to optimise the positive angle one of the pair, so a total of four numbers were needed to fully describe each stage.

The stopping criterion was the discovery of a filter within a given maximum passband ripple and minimum stopband attenuation. It has been found that the high-fitness area of the search space around the optimum solution for many filters of this type is extremely small, and becomes smaller as the number of sections rises. While the GA can find the general region of the best solution, it is usually unable to reach the actual optimum, so we have used a hybrid optimiser in the form of a Simplex method hill-climber. This has been found to be effective when used in conjunction with the GA when an optimal solution is required. The Simplex search on its own, however, is not suitable, as there are a number of peaks in the search space, so unless it is started adjacent to the optimum peak, it will never find it.

It was found that the GA was slow to reach the optimum solution, so since very fast methods of filter design such as the BZT and impulse-invariant method are available, it was decided to use a BZT design to initialise the GA. One member of the population was initialised directly with the BZT solution, while the remainder of the population were filled with randomly perturbed values of the pole and zero positions, chosen at random. This allows the GA to search for better solutions with differently ordered second-order sections, initially in the region of the solution found by the BZT. The GA also uses the filter order recommended by the BZT to ensure that a viable solution can be found.

The GA was then extended to optimise the coefficients in their quantised form. The fitness function was adapted to convert the chromosomes to their quantised values before calculating the filter's response and fitness. In this form, the GA was able to move closer to the optimum solution, as the search space now only contains a limited number of points. However, the Simplex search performed more poorly, as the search space is no longer a smooth surface, but is made up from small regions with uniform fitness. Once the simplex lies fully within a uniform region it cannot proceed with its search, so its convergence properties are much poorer when designing a quantised filter. With low precision filters this is not a problem, as the GA is able to perform the optimisation, but at higher a precision of, for example, 16 bits, the solution is poorer, as the GA relies on the hybrid simplex for the final optimisation.

This GA was found to perform poorly and to be extremely slow to reach the optimum solution, so a binary-coded GA was also developed in order to compare the performance of the two approaches [9].

The binary GA uses genes containing Gray-coded versions of two's complement fixed precision numbers, which are the a_k and b_k coefficients. The overall wordlength and the number of fractional bits are specified by the user. This approach forces the GA to work with quantised numbers, thereby removing one set of calculations from the fitness function. The seeding of the first random population was changed to remove the unperturbed BZT solution, as it was found that for tight tolerance filters the GA rarely looked anywhere else as the high fitness regions are relatively small.

To prevent unstable filters from being designed, any filter whose b_1 and b_2 coefficients fell outside the stability triangle of:

$$\begin{aligned} |b_2| &< 1 \\ |b_1| &\leq 1 + b_2 \end{aligned}$$

are immediately returned with a low fitness.

The operation of the GA was extended to include the roundoff noise in the fitness function. By doing so, the GA also includes the ordering of the second order sections as an intrinsic part of the optimisation, in order to produce a filter with as low a roundoff noise as possible.

A multi-objective fitness function was chosen to allow the user to specify the relative importance of the frequency response and roundoff noise of the filter to be designed. This allows the GA to search for filters ranging from those with optimal reponses to those having optimal noise characteristics. The phase response was not included, although it will be examined at a future date.

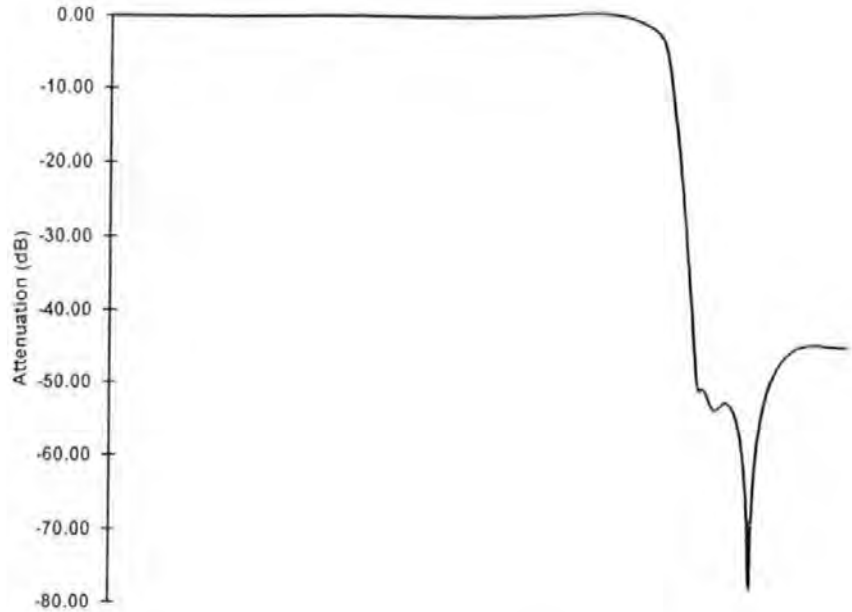


Figure 2: Lowpass filter designed by GA

The chosen fitness function was:

$$Fitness = \frac{1}{\epsilon_{max}} + \frac{1}{W.\sigma_{or}^2}$$

where ϵ_{max} is the maximum deviation from the required frequency responce in the pass- and stopbands in dB, and W is a weighting value. The weighting value was varied to alter the trade-off between forcing the GA to a good frequency response and trying to find a filter with a low roundoff noise gain.

3 Results and discussion

To test the GA, we compared its results to those found by the BZT and quantised to the same wordlength. When asked to find an eighth-order lowpass filter with normalised bandedge frequencies of 0.35 and 0.4, a wordlength of 16 with 13 fractional bits, maximum passband ripple of 0.25dB and minimum stopband attenuation of 45dB, a GA with a weighting value of $W = 100$ found the filter shown in Figure 2. The characteristics of the GA filter are compared with those for the quantised BZT solution in the table below.

Value	GA	BZT
pb ripple	0.267dB	0.247dB
sb atten.	45.417dB	94.246dB
σ_{or}^2	$122.849.q^2$	$520.566.q^2$

A tenth-order bandpass filter with band edges of 0.13, 0.17, 0.25, 0.29, with a wordlength of 24 including 20 fractional bits, and ripple tolerances of 0.2dB and 50dB in the pass- and stopbands was found, with the following results, by a GA with $W = 1000$:

Value	GA	BZT
pb ripple	0.551dB	3.793dB
sb atten.	49.690dB	49.902dB
σ_{or}^2	$1196.06.q^2$	$56.822e6.q^2$

The solutions found by directly quantising the BZT solution seem to fall into two categories: those with a good frequency response and high roundoff noise gain, and those with a low roundoff noise gain but poor frequency response. This suggests that the solutions found by the BZT lie in a small high-fitness region within an otherwise poor area of the search space, where simply quantising the filter coefficients is enough to turn the filter into an unsuitable, low-fitness solution.

These results show that the hybrid GA is a viable strategy for quantised IIR filter design, although for very low tolerance filters (e.g. with narrow transition band widths)

the BZT solution is so good that it should be included in the seeding of the GAs initial population. It also demonstrates that the GA is able to improve the roundoff noise gain of a BZT filter from that obtained when the coefficients are simply quantised.

The advantage of the combined BZT/GA approach is that the GA can be taken rapidly to good possible regions of the search space, and explore these areas for the best quantised solutions more successfully than either can alone. It has also reduced the number of stages of design by integrating coefficient calculation, section ordering, and noise analysis into a single parallel operation, thereby allowing designers to produce a complete IIR filter from a few initial specifications, without needing to understand the many different intermediate stages. The addition of extra genes to control the structure used by each second order section could also allow an improvement in the roundoff noise gain.

Wordlength optimisation is currently only possible by performing several GA runs with different wordlengths and comparing results. However, the future inclusion of genes in the chromosome to control the wordlength should allow the production of the filter with minimum wordlength that gives the desired response. Future work will examine different approaches to multi-objective fitness functions as the number of target filter specifications increases.

4 References

1. J.H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press, 1975.

2. D.E. Goldberg, *Genetic Algorithms in Search, Optimisation, and Machine Learning*, Addison Wesley, 1989.

3. T. Görne and M. Schneider, *Design of Digital Filters with Evolutionary Algorithms*, in "Artificial Neural Nets and Genetic Algorithms", Albrecht, Reeves and Steele, eds.

4. L.J. Nicolson and B.M.G. Cheetham, *An Investigation into the Multiple Criterion Optimisation Approach to IIR Filter Design*, Colloquium on Digital and Analogue Filters and Filtering Systems, 1992, pp 4/1–4/6.

5. E.C. Ifeachor and S.P. Harris, *A New Approach to Frequency Sampling Filter Design*, Procs. IEE/IEEE Workshop on Natural Algorithms in Signal Processing, 1993.

6. S.P. Harris and E.C. Ifeachor, *Automatic Design of Frequency Sampling Filters by Genetic Algorithm Techniques*, Under review by the IEEE.

7. E.C. Ifeachor and Barrie W. Jervis, *Digital Signal Processing: A Practical Approach*, Addison Wesley, 1993.

8. C.Z. Janikow and Z. Michalewicz, *A Specialised Genetic Algorithm for Numerical Optimisation Problems*, 2nd International IEEE Conference on Tools for A.I. Proceedings, 1990, pp 798–804.

9. D.E. Goldberg, *Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking*, Complex Systems, Volume 5, 1991, pp 139–167.

Automatic Design of Frequency Sampling Filters by Hybrid Genetic Algorithm Techniques

Stephen P. Harris and Emmanuel C. Ifeachor

Abstract—**A new method of designing recursive and nonrecursive frequency sampling filters is presented. We investigate the use of a hybrid real-coded genetic algorithm (GA) for optimising transition sample values to give the maximum stopband attenuation. A modification allows the coefficient wordlength to be optimized concurrently, thereby reducing the overall number of design steps and simplifying the design process. The technique is able to consistently optimize filters with up to six transition samples. Designing digital filters is a complex process involving optimization at several discrete design steps. The techniques presented here could form the basis for integrating several of the optimizations. Investigations into increasing this integration by using a binary-coded GA to optimize nonlinear phase, quantized coefficient FIR filters are introduced, with an analysis of the difficulty of the problem from a GA perspective.**

I. Introduction

THE DESIGN of digital filters, as with most engineering tasks, is a multistage, iterative process. The key stages are filter specification, coefficient calculation, structure realization, finite wordlength analysis, and implementation. Each stage involves optimization, but current practice is to perform this separately for each stage in an iterative fashion until an acceptable solution is found. However, since the effects of each stage are interrelated, optimization at only one stage leads to a design that, although optimal for that design stage, will generally be suboptimal for others. An attractive goal is to perform the optimization for several stages simultaneously in order to seek out the filter design with the best overall tradeoff across the design criteria.

The frequency sampling (FS) method has attracted a considerable amount of attention as a filter design method [1]–[14]. When used to design standard frequency-selective filters, a low number of regularly spaced samples are chosen, with fixed values in the passband and stopband and a few variable samples in the transition band, which are optimized to maximize the filter performance according to its desired use.

The FS method has the advantages that more efficient recursive versions of standard narrowband nonrecursive filters can easily be found and that filters with an arbitrary response can also be designed. The key task in the FS method is to find the values of the transition band frequency samples that produce a filter with the desired continuous response, for example, one with the maximum possible stopband attenuation. While tables of transition sample values for a limited selection of filters have been published, the interpolation required to produce an untabulated filter will result in a suboptimal solution. Designing filters with quantized coefficients adds a further level of complexity as the coefficients are often quite different to those found by simply truncating or rounding the full-precision coefficients found by standard techniques.

In many real-time applications where a linear-phase response is required, such as audio and biomedicine, FIR filters may be unsuitable, due to their unacceptably long delay of half the filter length. However, if the phase response is unconstrained, then a reduction in order of up to 50% can be made for some filters, with no loss of quality in the filter response. A compromise solution is to only optimize the phase with respect to its linearity in the passband, where the signal is most important. This releases degrees of freedom to improve the performance in the magnitude response and should allow a filter to be produced with the same magnitude response and near-linear phase in the passband but with a lower order and, therefore, a shorter delay.

In recent years, natural algorithms such as the genetic algorithm (GA) [5], [6] and simulated annealing (SA) have become popular optimization tools for performing searches of hilly, multidimensional surfaces where traditional methods such as hill climbing cannot perform well. These features can be utilized in digital filter design to perform various optimizations [7]–[10], whereas Darwinian optimization, which includes elements of both GA and SA methodology, has also proved successful [11]. The GA is capable of performing multicriterion optimization (MCO) in ways that automatically perform performance tradeoffs between design specifications.

In this paper, we introduce a fundamental investigation into the feasibility of using GA's to simplify the digital filter design process. Investigations into the use of the GA for optimizing frequency-sampling FIR filter coefficients for both recursive and nonrecursive representations are detailed, along with the floating-point hybrid GA that was developed to tackle the problem. This GA uses a family of related crossovers, and a new technique is described that dynamically biases their selection according to their success in improving the fitness. A modification allows the GA to optimize the wordlength of recursive filters simultaneously with their coefficients with no user intervention. An analysis of the nature of the search space was undertaken in order to explain and improve the GA's performance.

In order to increase the number of design steps being undertaken simultaneously, a binary GA was developed that optimized quantized coefficients directly while trading-off magnitude and phase response performances against each

Manuscript received March 25 1997; revised March 4, 1998. The associate editor coordinating the review of this paper and approving it for publication was Dr. José C. Principe.

The authors are with the School of Electronic Communication and Electrical Engineering, University of Plymouth, Plymouth, U.K.

Publisher Item Identifier S 1053-587X(98)08701-7.

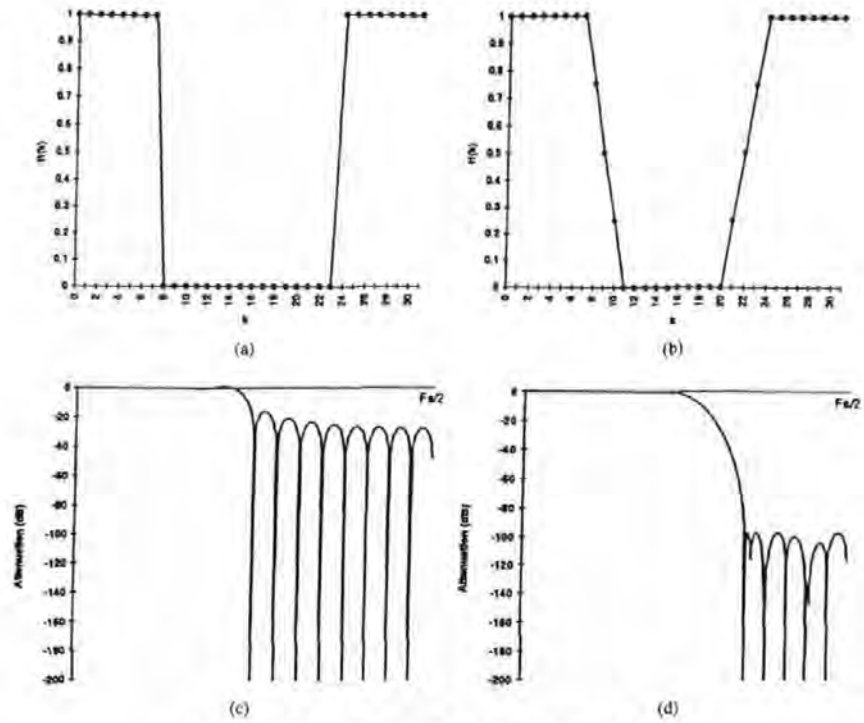


Fig. 1. Stages in the design of a frequency sampling filter.

other. This was undertaken with the aim of presenting the designer with a range of filters with different characteristics from which the most appropriate may be chosen. This approach proved to have a very poor performance, and our subsequent investigations into the search space and GA difficulty of the problem are introduced briefly.

II. Basic Theory of the Frequency Sampling Filter

An FIR filter can be uniquely represented by a set of frequency samples taken at regular intervals, as shown in Fig. 1(a), where they have been set to unity in the desired passband and to zero in the stopband. From this, the filter coefficients (and impulse response) can be found by taking an inverse DFT. The continuous frequency response can be approximated by zero padding the impulse response to a length of 1024 or more to obtain reasonable accuracy and taking a forward DFT (usually an FFT). This interpolated response can then be analyzed to determine its performance with respect to the problem under investigation. For a "brick-wall" filter such as this, the interpolated response has a large degree of ripple, resulting in a filter with poor attenuation, as shown in Fig. 1(c). It is possible to improve the performance of a "brick-wall" filter obtained in this way by including some variable samples to smooth the transition between passband and stopband, as in Fig. 1(b). The value of these *transition samples* can be optimized in order to give the minimum ripple in the passband and/or stopband; see Fig. 1(d). Once the transition samples have been optimized, an inverse DFT of all the samples can be used to give the filter coefficients.

It is possible to express the transfer function of any FS filter in either a recursive or nonrecursive form. The set of stopband attenuations over all possible transition sample values has a unique maximum for both nonrecursive and recursive filters [1], which means that a gradient descent algorithm will always be able to find the optimum solution, although this might not occur within a useful timescale for highly dimensional problems. A certain amount of off-line precalculation can be employed to improve the computational efficiency for nonrecursive filters, but this design method is inefficient for narrow-stopband filters, where only a small portion of the interpolated frequency response is needed to discover the attenuation. A recursive representation of the FIR filters allows a greater degree of precalculation and a more efficient implementation in most cases, although increases in computing power reduce the absolute speed differences between the two methods.

For variable-wordlength recursive filters, where both the transition samples *and* wordlength are being optimized, the search also has to cover a range of possible wordlengths. In this case, there is no longer a simple, smooth surface; therefore, pure hill-climbing methods can no longer be relied on to find the optimum. However, due to the coefficient quantization, the number of points in the search space is vastly reduced, and the search becomes more straightforward for the GA.

A. Recursive FS Filter Design

It is possible to calculate the interpolated frequency response of a recursive form FS filter directly by use of the z transform,

which permits much of the calculation to be performed offline. It is also possible to calculate the response for just a limited region of interest, such as the stopband. For an FIR filter with radius r, the recursive form transfer function can be written as

$$H(z) = \frac{1 - r^N z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H(k)}{1 - e^{j2\pi k/N} r z^{-1}} \quad 0 < r \leq 1 \quad (1)$$

where the factor outside the summation is a comb filter with N zeros spaced uniformly around a circle of radius r, cascaded with N single pole filters, represented by the summation term. These poles are all coincident with the comb filter's zeros at the points where $z_k = r\, e^{j2\pi k/N}$ with fully accurate arithmetic. In practice, when limited precision arithmetic is used, it is necessary for the radius to be less than one to prevent possible instability, with systems implemented with fewer bits requiring a smaller radius due to the larger perturbation caused by quantization. For a linear-phase FIR filter, $H(z)$ may be expressed by (2) [12], shown at the bottom of the page, where $\alpha = (N-1)/2$, and $M = (N/2) - 1$ for N odd and $M = (N-1)/2$ for N even. In our implementation, the radius r was either fixed by the designer or by the wordlength of the filter under investigation.

If r is exactly one, (3) can be used to avoid numerical overflow problems and allow a greater degree of precalculation as

$$H(e^{j\omega}) = e^{\frac{-j\omega(N-1)}{2}} \left[\frac{|H(0)|}{N} \frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)} + \sum_{k=1}^{\frac{N}{2}-1} \frac{|H(k)|}{N} \times \left[\frac{\sin\left(N\left(\frac{\omega}{2} - \frac{\pi k}{N}\right)\right)}{\sin\left(\frac{\omega}{2} - \frac{\pi k}{N}\right)} + \frac{\sin\left(N\left(\frac{\omega}{2} + \frac{\pi k}{N}\right)\right)}{\sin\left(\frac{\omega}{2} + \frac{\pi k}{N}\right)} \right] \right]. \quad (3)$$

The GA uses the expression between the outermost set of brackets to calculate the magnitude response for full-precision filters.

Depending on whether the first sample is taken at $\omega = 0$ or $\omega = \pi/N$, the filter is described as Type I or Type II. Since N can be odd or even, we therefore have four possible sample arrangements for a frequency sampling filter. With linear-phase constraints, FS filters are restricted to certain frequency-selective filters, depending on whether N is odd or even, which can be illustrated by the Case 2 type of impulse response, where we can express the magnitude response for N odd by [13]

$$H^*(e^{j\omega}) = \left(\sum_{n=1}^{N/2} a(n) \cos\left(\omega \left(n - \frac{1}{2} \right) \right) \right) \quad (4)$$

where $a(0) = h((N-1)/2)$, $a(n) = 2h((N-1)/2 - n)$. This equation is always zero valued at $\omega = \pi$, regardless of the actual filter coefficients since $\cos(\pi(n - 1/2))$ is always zero for integer n. This means that it is impossible to use a symmetric, N-even impulse response for any type of filter that is nonzero at $\omega = \pi$, such as highpass and bandstop. The available filter types are shown in Table I.

TABLE I
DESIGNABLE FIR FILTERS

Filter Type	IR Case	Possible Filters
Type-I, N odd	Case 1	All filters
Type-I, N even	Case 2	Only lowpass and bandpass
Type-II, N odd	Case 1	All filters
Type-II, N even	Case 2	Only lowpass and bandpass

III. THE GENETIC ALGORITHM

The GA, as developed by Holland [5], [6], has in recent years become a popular and powerful search technique. It is based on ideas borrowed from the theories of natural selection—the "survival of the fittest." In nature, evolution can be viewed as a search process where the optimum DNA must be found in order to maximize a species' chances of surviving long enough to reproduce, thereby propagating the species. This occurs through a process of DNA crossover during sexual reproduction and the loss of unfit offspring due to predation or poor adaptation to the surroundings.

The GA uses a similar process to perform the search for the optimum solution of a problem represented by an artificial computer model. The parameters that tune the performance of this model are represented within the GA as a number of *chromosomes*, most commonly in a binary representation, although a real-coded chromosome may be more appropriate for some problems. These chromosomes make up the GA's *population*, which contains all the information that the GA has found about the good regions of the search space. They are generally initialized at random in order to ensure that the search space is sampled widely and evenly.

In order to determine which of the members of the population contain solutions that are good enough to continue to the next generation, the "fitness" of each must be found. This is performed by the *fitness function*, which decodes the binary chromosomes into their model parameters and calculates the corresponding performance. The better solutions are selected according to fitness to undergo reproduction into the next generation, whereas the poorer solutions are lost. This selection increases the average fitness of the population but does not introduce any new solutions. In order to distribute information about the good solutions that have been selected, the new population undergoes *crossover*, which, in a similar way to the natural process, involves the swapping of sections of the chromosome between randomly selected pairs. If there are just a few good solutions within a population, it is possible for them to overrun the population and cause *premature*

$$H(z) = \frac{1 - r^N z^{-N}}{N} \left[\left[\sum_{k=1}^{M} \frac{|H(k)|(2\cos(2\pi k\alpha/N) - 2r\cos(2\pi k(1+\alpha)/N)z^{-1})}{1 - 2r\cos(2\pi k/N)z^{-1} + r^2 z^{-2}} \right] + \frac{H(0)}{1 - r z^{-1}} \right] \quad (2)$$

convergence, thereby allowing the GA to become "stuck" in a suboptimal solution. To combat this, *mutation* is used to allow the GA to escape to new regions of the space. Mutation simply involves the flipping of a few bits within the population at random with a low probability, as it can obviously be very disruptive.

The GA has the advantages that it can escape from suboptimal peaks even after it has converged, through the action of mutation, so that it can be used in highly multimodal situations, unlike traditional hill-climbing techniques. It can also be written in a way that makes it suitable for multicriterion optimization: a feature that is planned to be utilized in future design areas. Unlike many other design techniques, it does not require any mathematical analysis; therefore, it can be used for complex problems where such an analysis would be extremely difficult and time consuming. The GA is also able to work successfully with discontinuous search spaces, such as those produced when using quantized model parameters, since only point samples are taken. This feature is utilized in the optimization of finite wordlength filters described in Section VI-A. It has the disadvantage that it is not *guaranteed* to find the optimum solution for any given problem; therefore, a secondary, hybrid optimization method is often used in conjunction with the GA in order to improve the final solution.

While a binary-coded model is most common, it may be advantageous to use real coding instead if the problem naturally involves the optimization of real numbers or if there is a large time overhead in converting the binary values to their corresponding real numbers. The GA used in the optimization of FS filters utilized a real coding, with floating-point crossover and mutation techniques based on those of a GA designed for numerical optimization [14].

IV. The Shape of the Search Space

In order to obtain a better understanding of the function that the GA is optimizing, contour maps of the search space for a two-transition sample lowpass filter were drawn by varying the transition samples at eight-bit resolution (i.e., 256 by 256 points) and finding the fitness of each solution. A typical surface for a lowpass filter is shown in Fig. 2. The restriction that the second transition sample (T_2) must be smaller than the first (T_1) means that half of the possible area is forbidden—the thick line marks the edge of this region, which lies in the upper left half of the figure. The jagged appearance is due to the limited resolution—in reality, all edges are smooth, and there is a unique peak. For fixed radius filters, the surface always has a unique maximum regardless of the number of transition samples. When N or the width of the passband is changed, both the maximum attenuation and optimum transition sample values change, along with the position of the peak and the orientation of the ridge it lies on; however, the similarity between all of the spaces means that the same optimization technique can be used for designing any type of filter, reducing the learning time.

Fig. 2 does not, however, give a full picture of the nature of the search space. If a lowpass response is to be considered valid, it must decrease monotonically between the last passband sample of unity and the first stopband sample of zero, which means that the derivative of the magnitude response in the transition band must always be negative. If the response decreases monotonically, so must the samples it is fitted through; therefore, each transition sample must be greater than the next. In general, from all possible sets of transition samples, the proportion in which each transition sample is greater than the next P_{N_t} is given by

$$P_{N_t} = \frac{1}{N_t!} \tag{5}$$

where N_t is the number of transition samples. This shows that the valid proportion rapidly becomes very small—by $N_t = 5$, P_{N_t} is below 0.01. The fitness function automatically orders the coefficients so that the GA is constrained to solutions that are always valid with respect to the transition samples. Unfortunately, this does not ensure that the *interpolated* response also decreases monotonically: a further constraint that is often broken in filters with similar adjacent transition sample values. To ensure that this additional, stricter condition is also met, a hard boundary can be added to the fitness function by making it return a very low fitness for any filters that do not meet it. This will ensure that any such solutions being examined by the GA will be lost during selection and, therefore, should help to restrict the search to the region of the search space containing truly valid solutions. This region is shown in Fig. 3. This region is clearly smaller than before; therefore, the valid proportion will shrink even more rapidly as the number of transition samples dimensionality rises. To obtain a large attenuation, the transition sample closest to the stopband T_2 must be small so that the peak lies close to the lower edge of the contour map, but if it drops too far, the continuous response rises again before reaching the stopband, giving an unacceptable filter. These two competing factors force the optimum solution to lie close to or on the edge of the constraint boundary so that when the GA is close to the optimum solution, it will also be in a region where

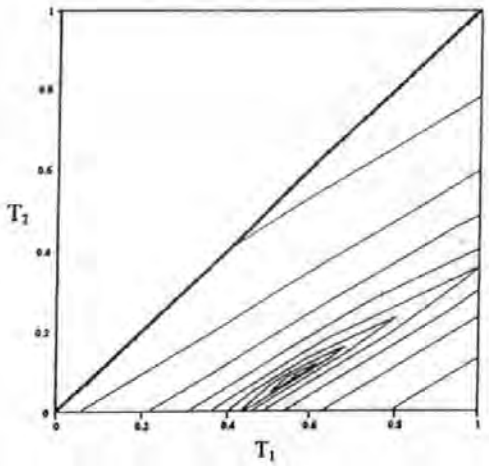


Fig. 2. Search space for a Type-I lowpass filter with $N = 46$ and a passband width = 0.166. T_1 and T_2 are the first and second transition samples, respectively.

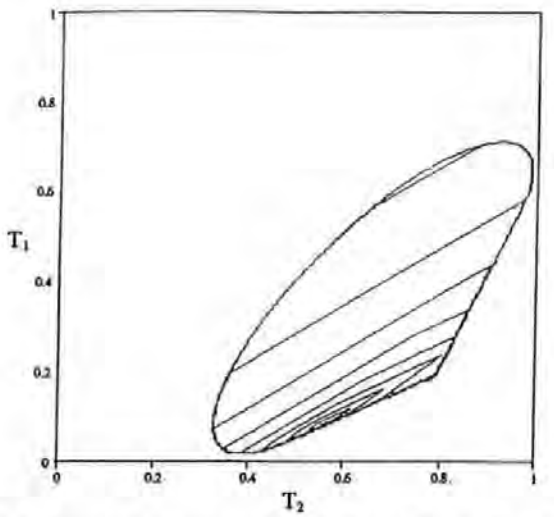


Fig. 3. Boundary of the constrained search space giving allowable solutions for the same lowpass filter; T_1 and T_2 are the first and second transition samples, respectively.

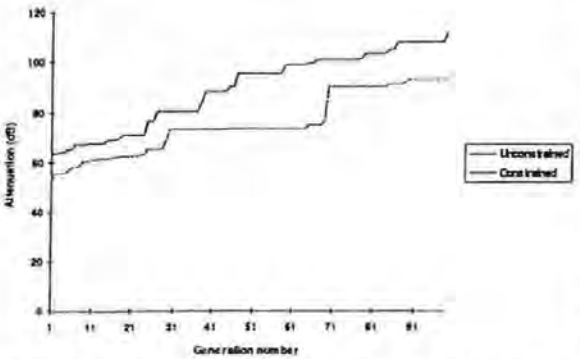


Fig. 4. Increase of fitness with generation for constrained and unconstrained optimization of an FIR filter.

crossover and mutation will often give a disallowed result. Fitness functions that constrained the search to this smaller region were tried, but for low numbers of transition samples, they did not noticeably alter the efficiency of the search. For filters with around five or more transition samples, this constrained search produced a poorer result, as illustrated in Fig. 4 for an $N_t = 6$, $N = 68$, Type-I lowpass filter, which compares the improvement of fitness with generation averaged over five runs. The poorer performance is due to an increasing proportion of the high-fitness region around the peak being outside the constraint boundary, which means that a small change in position can change a solution from being very good to being invalid. This in turn implies that crossover and mutation become increasingly ineffective, and the GA becomes less able to improve the filter as the optimum is approached. Since the use of the hard constraint boundary did not improve the performance, it was not used in the final GA.

It has been found that the search space for a recursive filter is very similar to that of a nonrecursive one, particularly when the filter has a radius close to one. If the radius is reduced, then the narrow ridge widens, and the maximum attenuation falls. Since the radius used is dependent on the wordlength, a shorter wordlength means that a smaller radius must be used to prevent instability, although this reduces the maximum attenuation that can be attained. When the GA is being used to optimize both the coefficients and the wordlength, this means that there are conflicting factors that need to be traded against each other; the coefficients need to give a high enough attenuation for the filter to be acceptable, whereas the wordlength needs to be as low as possible in order to make the filter implementation as efficient as possible. The fitness function used to attain this is discussed in Section VI-A.

V. Using the GA for FS Filter Design

It should now be clear that to perform FS filter design, the GA is used to optimize the values of the transition samples between the passband and stopband. The chromosome used by the GA is a string of N_t real-valued genes, where N_t is the number of samples in the transition band. For recursive filters, the wordlength can also be included in the chromosome. All gene values are constrained to lie in the range 0–1 so that they can be used directly as normalized transition frequency samples to calculate the fitness.

The floating-point GA used to optimize the filter coefficients and radii was based on that of Janikow and Michalewicz [14], where a number of vector convex combinations were used to perform floating-point crossover, although with a number of adaptations. The first two types of crossover simply exchange transition sample values without changing them, swapping either one or several of the them between the parent chromosomes. These do not, however, generate any new gene values, which is catered for by the remaining three types. These produce offspring by taking pairs of parent chromosomes, copying them, and replacing one, several, or all of the transition samples within them as

$$c = R \cdot a + (1 - R) \cdot b \tag{6}$$

$$d = (1 - R) \cdot a + R \cdot b \tag{7}$$

where a and b are the parent gene values, c and d are the child gene values, and R is a random number between 0–1. It should be noted that the offspring's genes can never lie outside the region bounded by the values of the parent genes; therefore, by applying crossover, the transition sample values are always moved closer by some degree. It has already been shown that the optimum solution lies close to the lower bound of values for the later transition samples so that the crossover was changed to

$$c = R \cdot a + (1.1 - R) \cdot b \tag{8}$$

$$d = (1.1 - R) \cdot a + R \cdot b \tag{9}$$

in order to enable crossover to move chromosomes apart sometimes and help it find solutions at the edges of the parameter space.

The fitness function used is simply the stopband attenuation, in decibels, which gave a better performance that the normalized stopband ripple. Since the latter is generally small, there

is little to drive the GA toward the better solutions; therefore, by converting to a logarithmic scale, we increase the emphasis on high attenuation. After performing trial runs, the mutation probability was set to 0.005 and the crossover probability to 0.7, with a population size of 30.

A simplex method [15] local hill-climbing routine is then used to complete the optimization begun by the GA. The simplex method was chosen because it does not require gradient information and does not involve any form of curve fitting such as parabolic interpolation, as this implies some knowledge of the structure of the parameter space, which is not necessarily available. Like the GA, it only takes point samples and does not require a detailed mathematical analysis of the problem. This extra optimization routine was only called after one quarter of the total number of generations had passed and if the GA had been unable to improve the best fitness for 20 consecutive generations.

A. Adaptive Selection of Crossover Method

In an attempt to improve the effectiveness of crossover, rather than selecting the crossover method purely at random, a novel dynamic selection method was implemented, where the probability of selecting a particular method is dependent on its current performance. The probability of selection is given by

$$p_x = \frac{d_x}{\sum_{l=0}^{5} d_l} \tag{10}$$

where d_x is the proportion of calls to crossover method x that caused an fitness improvement over the parent chromosomes, p_x is the probability of choosing crossover type x, and the summation is over the five types of crossover plus $l = 0$ for just mutation. This method allows the best crossover method to be selected dynamically during the run by its performance.

When the number of transition samples is small ($<\sim 5$), the crossover types that actually alter gene values perform best early in the run, whereas later on, those that simply exchange genes and pure mutation become dominant. This is because once the population has converged to the region of the optimum, changing gene values by crossover is likely to move the solutions to a much poorer solution so just swapping existing values will be more useful.

For larger N_t ($>\sim 5$), all types initially have a similar performance, but the same crossovers and mutation take over later on. Their initial performance is better than that for the short chromosomes since there are more gene values in the population, which are available to several gene positions due to the reordering that occurs in the fitness function. Exchanging them will therefore be more productive than before. The inclusion of this selection process allowed the GA to make more regular improvements in fitness so that it needed to rely less on the simplex local search, although as the optimum was already being found, the best solution found overall did not improve.

B. Improving the Computational Efficiency

Although the GA itself is very efficient and fast, it requires many fitness calculations so that the efficiency of the implementation of the problem model is of paramount importance. For all but the simplest of problems, the GA will spend most of its time calculating fitnesses so that some effort was put into optimizing the fitness calculations.

The nonrecursive method consists of an inverse DFT followed by an FFT to produce the interpolated response. This has a total operation count of

$$\Lambda_{\mathrm{cm}} = N^2 + \frac{N_i}{2}\log_2(N_i) \tag{11}$$

$$\Lambda_{\mathrm{ca}} = N(N-1) + N_i\log_2(N_i) \tag{12}$$

where

Λ_{cm} number of complex multiplies;
Λ_{ca} number of complex additions;
N filter order;
N_i number of points in the interpolated spectrum.

Since the coefficients in the passband and stopband will have the same effect on the inverse DFT part of the calculation, their effect can be precalculated. In addition, utilizing the symmetry of the impulse response reduces the contribution of the DFT to the sum to N_tN in both cases. The FFT is read-only and can be packed into a shorter FFT of length $N_i/2$, resulting in new operation counts of

$$\Lambda_{\mathrm{cm}} = N_tN + \frac{N_i}{4}\log_2\left(\frac{N_i}{2}\right) \tag{13}$$

$$\Lambda_{\mathrm{ca}} = N_tN + \frac{N_i}{2}\log_2\left(\frac{N_i}{2}\right). \tag{14}$$

Assuming that with a DSP or maths co-processor both operations take the same time, the total operation count can be given by:

$$\Lambda_t = \frac{1}{2}(4N_tN - 5N_i + \log_2(N_i)). \tag{15}$$

The recursive filter response was calculated by using (3) to produce a frequency response containing 512 points over the interval $\omega = 0$–π, which was then examined to determine the stopband attenuation. This meant that only a fixed set of ω values are used, and the sine terms could be precalculated. The contribution of the passband samples to the final response can also be precalculated, leaving just the effect of the N_t transition sample values to be determined at each fitness calculation. Further savings can be made by simply calculating the response in the stopband as this is the only region of the response used to determine the fitness. Taking all of these factors into account, we get the expression for the recursive total operation count Λ_t^r of

$$\Lambda_t^r = \frac{2N_iN_tN_s}{N} \tag{16}$$

where N_s is the number of stopband samples in the range 0–π.

Although these optimizations have improved the absolute computational efficiency of both calculations, their operation counts show that the recursive implementation is still much more efficient than the nonrecursive one for low numbers of transition samples. Since there are usually fewer than ten such samples in practice, the recursive filter GA will be faster than the nonrecursive.

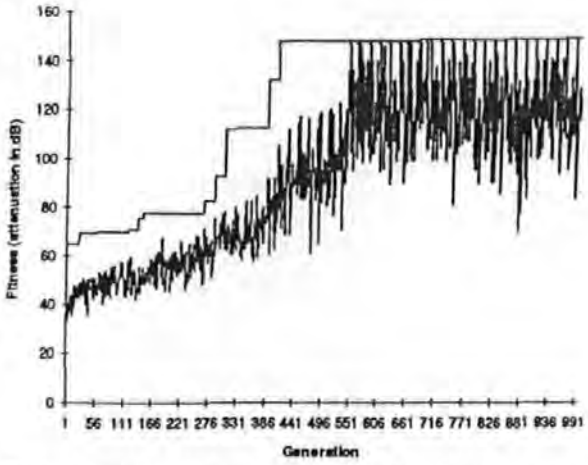


Fig. 5. Improvement of the maximum and average fitnesses with generation.

It should be noted that these results are somewhat conjectural as compiler optimizations and different memory access counts will result in the relative speeds diverging from these predictions. The operations counts are, however, sufficiently divergent that the recursive implementation is certain to be faster on average for up to ten transition samples; therefore, it was adopted as the standard method.

VI. Results for FS Filters

The improvement of maximum and average fitness with generation can be seen in Fig. 5 for a typical run to design a Type-II highpass filter with $N = 89$, a narrow stopband of three samples, and five transition samples. The run of 1000 generations was completed in around 4 min on a 486 DX2-66 PC, although a near-optimal solution was found after about generation 400. The regularly spaced vertical peaks in the latter part of the graph show where the local search routine was called after the GA failed to improve the best fitness for 20 generations. The GA is able to find the general area of the peak fairly quickly, but finds it difficult to find very high fitness solutions within the very small area of the peak, as crossover is likely to throw the offspring into comparatively very poor regions. The GA is therefore used alone for the first quarter of the run, after which the hill climber is able to make substantial improvements.

The technique was tested against tabulated results from [1] and was found to equal or improve on them in every case, whereas in [3], results for untabulated filters are given. Fig. 6 shows a Type-I filter with $N = 128$, $N_p = 20$, $N_t = 3$, and $N_s = 15$, which was designed by our hybrid GA simplex method. The attenuation is 85.5 dB, and the passband ripple was 0.10 dB. The impulse response for this filter is shown in Fig. 7. A Type-II bandstop filter is shown in Fig. 8, which has $N = 99$, and passband and stopband widths of 13 and 11 samples, respectively. This filter has an attenuation of 111.4 dB and a passband ripple of 0.075 dB.

It has been found that this technique is generally very robust for filters with up to six transition samples, taking an increasing

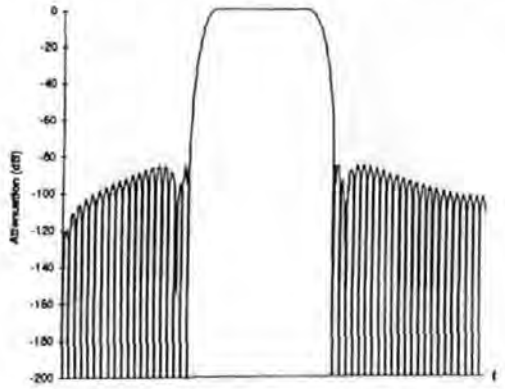


Fig. 6. Frequency response for a bandpass filter designed with GA-Simplex (see text for details).

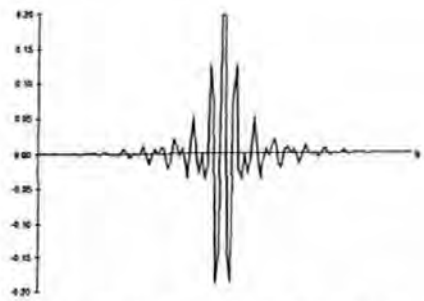


Fig. 7. Impulse response for the filter in Fig. 6.

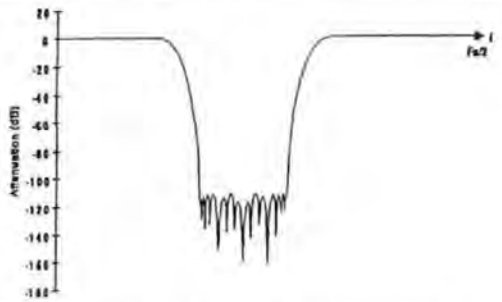


Fig. 8. Frequency response of a bandstop filter designed using GA simplex.

length of time as the number of transition samples rises. For filters with up to around four transition samples, the GA is able to find near-optimal solutions very quickly before the intervention of the local search. For five or six transition samples, the GA performed a useful amount of improvement although not to near-optimal performance, whereas the local search was able to complete the optimization. For more transition samples (up to ten were used), the hybrid GA was only able to perform a small amount of optimization, from which the local search was also unable to find the optimum within a reasonable time (e.g., 1 hr on a 486 PC).

Although the hybrid GA has been able to produce results that improve slightly on those in the literature [1], its main strength lies in the fact that it can quickly produce untabulated

filter coefficients, which are much more useful than those found by interpolation. It is also able to design filters with more transition samples, showing that the hybrid GA is a suitable technique to use for designing this type of FIR filter. The successful application of the GA to this problem leads us to consider how its use could be extended to more complex design tasks in order to perform several design steps simultaneously and thereby simplify the design process.

A. Concurrent Optimization of the Wordlength

In order to integrate a further filter design step, the fitness function for optimizing recursive filters was extended to incorporate the finite wordlength effects of coefficient quantization by including an extra gene in the chromosome that determines the wordlength at which the genes will be decoded. This allows the GA to search for the minimum wordlength necessary to achieve a given filter specification simultaneously with its coefficients. Mixed-integer programming has proved successful at optimizing quantised coefficients [16], although separate runs are required for each wordlength under investigation. Our approach allows the GA to search for the minimum wordlength *and* optimum coefficients simultaneously, with no user intervention.

Since we are using a real-coded chromosome with genes in the range 0–1, the wordlength gene must be decoded to give an integer wordlength, which is performed by scaling it up to 0–24 and taking the nearest integer. The real-coded genes are then quantized to this wordlength before being used to calculate the filter response and then fitness. This simplifies the search by limiting the number of points the GA has to examine. It has the disadvantage that the simplex hill climber is less effective because the search space is now made up of a large number of flat regions, such as those shown at 8-bit resolution in Fig. 2. This is due to the quantization of the coefficients causing finite ranges of the floating-point gene values to be interpreted as having the same value by the fitness function; therefore, they will have the same effect on the filter response. At the beginning of a search, the hill climber is able to perform well because from a large-scale perspective, the surface is smooth; however, once the search contracts around a good region, the small plateau become increasingly apparent, and eventually, the search cannot gain any information about the direction of the optimum and is unable to reach it. The GA is still able to perform successfully in such a space (which resembles the de Jong GA test function f3 [6]) as it only relies on point fitness samples and is unaffected by discontinuities or perfectly flat areas. This implies that more reliance will be placed on the GA to perform a good optimization since the simplex will be less effective here.

When calculating the filter response, in order to maintain filter stability, the radius was reduced to one less the quantization interval

$$r = 1 - \frac{1}{2^B} \tag{17}$$

where B is the wordlength. The fitness now has to take account of both the magnitude response and the wordlength with emphasis on the former since this constraint should be satisfied regardless of the wordlength. To this end, the following scheme was devised.

First, the normalized magnitude response is examined in both the passband and stopband to see if it fits within the desired limits. If it does to within 10^5, which corresponds to a deviation of only around 0.3 dB from a desired attenuation of 70 dB, then the basic fitness is set to 10^5; otherwise, it is set to the reciprocal of the normalized deviation. This gives a main fitness range of 0–10^5 for the magnitude response and is flat (at 10^5) for all filters fitting within the desired specification. By having all satisfactory solutions return the same fitness, it means that the GA is free to return a solution that only *just* fits the design specification, leaving it more freedom to reduce the wordlength. To account for the wordlength, a further term is added to this, consisting of 25 minus the wordlength. This overall fitness function therefore has extra structure, especially within the optimum peak region, which allows the GA to search for the minimum wordlength. The overall fitness function can be written as

$$f(x) = \begin{cases} (25 - B) + 1/e_{\max} & e_{\max} > 10^{-5} \\ (25 - B) + 1/10^{-5} & e_{\max} \le 10^{-5} \end{cases} \tag{18}$$

where B is the wordlength, and $e_{\max}$ is the maximum absolute error between the normalized filter response and the desired response in the passband and stopband.

This approach places the major emphasis on the optimization of the magnitude response, and once this has been achieved, the effect of the wordlength dominates (within the optimum fitness "plateau"). Other weightings have been tried, but these were found to allow the GA to perform efficiently without the intervention of the simplex local search.

Results for a typical test run are given below in Table II for a 49th order, four transition sample lowpass filter with a bandwidth of 0.25. The GA was able to fit to the desired specifications with coefficients quantized to a wordlength of only six bits. The full-precision fitness function (in which the GA only seeks to minimize the stopband ripple) was able to find a solution with much greater stopband attenuation as the last transition sample was able to have a much smaller nonzero value, as shown in Table III. The full-precision fitness function used the same radius as the 6-bit solution, and the frequency responses of both of these filters are shown overlaid in Fig. 9.

TABLE II
DESIRED AND OPTIMIZED SPECIFICATION FOR A QUANTISED-COEFFICIENT AND FULL-PRECISION FILTER. THE FULL-PRECISION DESIGN USES A MAXIMUM-ATTENUATION FITNESS FUNCTION

	Desired	Quantized	Full-Precision
Passband ripple (dB)	0.1	0.058	0.129
Stopband Attenuation (dB)	77	82.96	117.26
Wordlength	—	6	—
Radius	—	0.984375	0.9843745

VII. QUANTIZED-COEFFICIENT, NONLINEAR-PHASE FIR FILTERS

A major objective of this work has been the simplification of the filter design process; therefore, our next goal was to extend

TABLE III
TRANSITION SAMPLES FOR THE FILTERS DESCRIBED IN THE TEXT AND TABLE II

Quantized	Full-Precision
0.875	0.785269
0.515625	0.348808
0.15625	0.065764
0.015625	0.003069

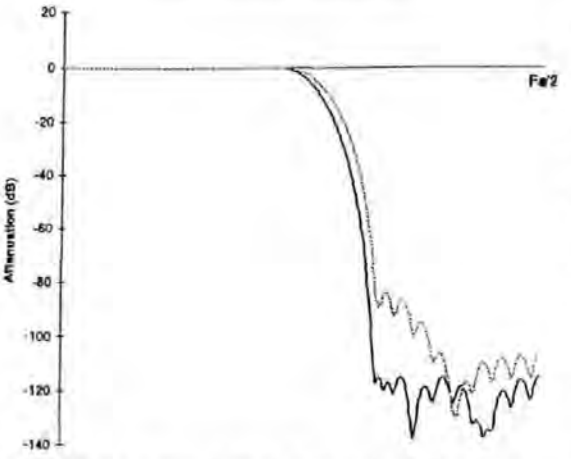


Fig. 9. Optimized 6-bit quantized-coefficient and full-precision coefficient recursive filters found by GA.

the GA to include a further simultaneous optimization. Since the linear-phase FS FIR filter had been optimized satisfactorily by the GA, a more difficult FIR filter design area was selected: that of nonlinear filters with quantized coefficients. This allowed both the phase and magnitude responses and the effect of coefficient quantization to be optimized at the same time.

Although FS filters are simple to design and can be forced to have exactly linear phase, they are not suitable for all applications, as they require a high order to produce a sharp cutoff in the frequency response and, therefore, a lot of storage for the coefficients. They also have a delay of $N/2$, which for high-speed real-time applications may be unacceptable. However, if there are no restrictions at all on the phase, then the filter length can be reduced by as much as half. For many systems, however, such as those found in audio and biomedicine, there may be strict requirements for linear phase to avoid signal distortion. A compromise solution can be reached by designing a nonlinear filter but, in addition, optimizing the deviation from linearity in the passband to be as small as possible. This means that the filter can be shorter than the linear-phase system but will be longer than the absolute minimum of the completely nonlinear one. Optimizing the phase adds another objective, which means that the simple GA used before is no longer suitable.

The standard GA only uses a single fitness measure, which is satisfactory for simple problems where only one or two criteria need to be optimized, but to do this requires some degree of trial-and-error to get satisfactory weightings for the various criteria. Although we have been able to optimize both wordlength and magnitude response this way, this approach is not a generally applicable technique and requires too much user intervention. To this end, a new GA is required that has been adapted to perform multicriterion optimization (MCO). In MCO [6], [17], an improvement in one criterion will often lead to a loss of performance with respect to another. This means that it will generally be impossible to produce a solution that performs perfectly in all respects. There is, however, a "wavefront" of best possible solutions, with a range of possible tradeoffs. This set, which contains those solutions for which it is not possible to improve the performance of all criteria simultaneously, is known as the *pareto-optimal* or *nondominated* set (NDS). None of the members of this set can be said to be "better" than any other because although one might outperform another with respect to one criterion, it will always have a poorer performance in at least one other. All other solutions are *dominated* by at least one member of this set, where $\mathbf{x}$ is said to dominate $\mathbf{y}$ if $\mathbf{x}$ is *partially less than* $\mathbf{y}$, which is defined as

$$(\mathbf{x} < p\mathbf{y}) \Leftrightarrow (\forall_i)(x_i \leq y_i) \wedge (\exists_i)(x_i < y_i) \qquad (19)$$

i.e., for all i, $x_i \leq y_i$, and for at least one i, $x_i < y_i$. None of the members of the NDS dominates another.

To perform MCO, a multiobjective GA was developed to perform the much harder optimization of a nonlinear filter's coefficients with respect to both the filter's magnitude response and its phase response in a region of the passband and to return the NDS of solutions found. To increase the number of filter design steps being undertaken simultaneously, the GA used a binary chromosome containing a concatenated list of the coefficient values, which therefore intrinsically accounted for coefficient quantization effects by searching directly for the optimum quantized coefficients. The aim of searching for the NDS is to offer designers a range of solutions with different tradeoffs between the various design criteria, enabling them to select the best filter for their specific problem.

The fitness function for the magnitude response was the maximum error from a desired response template and, for the phase response, was the χ^2 error between the response and a least mean squared (LMS) straight line fitted through the response in a selected region covering most of the passband.

A. Performance of the Nonlinear-Phase FIR GA

Initial runs were performed with randomly initialized chromosomes and proved unable to find solutions that fitted the magnitude response design templates even to within 10–20 dB, although the optimization of the phase linearity was generally more successful. It was also found that seeding the GA with perturbed copies of a known good Remez exchange solution [12] either caused premature convergence for a low perturbation, or the search failed for a higher one.

Since the GA was not performing well, an analysis of the difficulty of the search from a GA perspective was undertaken. The preliminary analysis of the problem was to take "slices" through the parameter space with a fitness function of just the magnitude response error; therefore, in this case, a lower fitness value means a better filter. To find a local near-optimum solution, coefficients calculated by the Remez

exchange method were used to seed a bit-flipping local search algorithm. Examination of these slices indicated that the parameter space becomes increasingly smooth away from the optimum and that the position of the optimum moves, eventually bearing no relation to its "best" position. This implies that far from the optimum, the GA has no useful information about where that optimum lies; therefore, the optimum cannot move toward it. It also means that as crossover and mutation alter coefficient values, the position of the optimum values of the *other* coefficients also move so that the information the GA contains about where the good regions are is useless as their position changes with each operation.

The GA is also hindered by the high *epistasis* of this problem, which is the degree to which the genes are dependent on each other's values to produce a high fitness solution. For a filter, the coefficients must provide an impulse response that has a well-formed shape. This, in turn, means that the only high-fitness chromosomes are *complete, optimal* solutions. Even if the GA can find two highly fit solutions, crossover between them will generally generate very poor solutions as the partial solutions will not produce a good impulse response when taken together. This means that the GA is unable to proceed effectively unless the entire population is very similar and has already converged around a good solution. Other difficulty measures also show that the problem becomes easier for the GA to solve as an optimum is approached, but the epistasis is the dominant factor and prevents the GA from even finding an optimum.

The phase optimization is more successful as there are a number of fixed, high-fitness solutions where the coefficients are near-symmetric, which gives a response close to linear. These solutions exist regardless of quantization or the coefficient values so that the GA is able to find them more easily.

VIII. Discussion and Conclusion

In this paper, a number of approaches to the optimization of FIR filters with both quantized and unquantized coefficients has been presented, with the aim of simplifying the filter design process by integrating a number of design optimizations into a single parallel one. This approach should make it easier to tradeoff the performance measures of the filter with respect to its various design criteria against each other in a controlled way.

The hybrid GA has been found to be a suitable technique for optimizing the unquantized coefficients of FS method filters with up to six transition samples. The simplex local search algorithm is generally required for more than four samples to perform the final optimization, due to the small area of the optimum peak and the disruptive nature of crossover and mutation. A GA has been developed with the capability of choosing the best crossover method to use dynamically during a run.

For recursive FIR filters, the GA has been extended to perform the minimization of the wordlength concurrently by optimizing the coefficients, thereby combining two filter design steps into a single process. The fitness function places most emphasis on the magnitude response to help ensure that the desired performance is reached, whereas the wordlength is most important within the set of suitable filters. This design problem has a search space that is unsuitable for standard hill-climbing optimization methods, and the GA is able to perform the optimization unaided by the hybrid simplex search.

To increase the number of design steps that were being undertaken simultaneously by the GA, a binary multiobjective GA was also developed to search for pareto-optimal sets, containing a number of solutions with varying tradeoffs between design criteria. This GA was tested on the design of nonlinear FIR filters, with a specified region of the passband where near linearity was desired, with the aim of giving a useful reduction in the filter's delay. This approach proved unsuccessful, and a detailed analysis of the problem was undertaken. The difficulty of the problem was analyzed from a GA perspective and showed that this design problem is not suitable for the GA in its current representation. This work will be detailed fully in a future paper.

Future work to examine the representation of the problem in the chromosome and the initialization may prove fruitful in enabling the GA to perform successfully, as may alternative hybrid techniques such as SA.

References

[1] L. R. Rabiner, B. Gold, and C. A. McGonegal, "An approach to the approximation problem for nonrecursive digital filters," *IEEE Trans. Audio. Electroacoust.*, vol. AU-18, pp. 83–106, June 1970.

[2] P. A. Lynn, "Frequency sampling filters with integer multipliers," *Introduction to Digital Filtering*, R. E. Bogner and A. G. Constantinides, Eds. London, U.K. Academic, 1975.

[3] E. C. Ifeachor and S. P. Harris, "A new approach to frequency sampling filter design," in *Proc. IEE/IEEE Workshop Natural Algorithms Signal Process.*, 1993.

[4] P. A. Stubberud and C. T. Leondes, "A frequency sampling filter design method which accounts for finite wordlength effects," *IEEE Trans. Signal Processing*, vol. 42, pp. 189–193, Jan. 1994.

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

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[7] S. P. Harris and E. C. Ifeachor, "Automating IIR filter design by genetic algorithm," in *Procs. GALESIA Int. Conf.*, 1995.

[8] I. Pitas, "Optimization and adaptation of discrete-valued digital filter parameters by simulated annealing," *IEEE Trans. Signal Processing*, vol. 42, pp. 860–866, Apr. 1994.

[9] T. Çiloğlu and Z. Ünver, "A new approach to discrete coefficient FIR filter design by simulated annealing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 1993, vol. 3, pp. C101–C104.

[10] O. V. Patiakin, B. S. Zhang, G. D. Cain, and J. R. Leigh, "An adaptive filter using darwinian algorithm for system identification and control," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 1993, vol. 4, pp. 627–631.

[11] G. Neri, G. D. Cain, T. Salmon, and A. Yardim, "A microprocessor-based digital flickermeter," *IEEE Trans. Instrum. Meas.*, vol. 40, pp. 1008–1014, June 1991.

[12] E. C. Ifeachor and B. W. Jervis, *Digital Signal Processing: A Practical Approach* Reading, MA: Addison-Wesley, 1993.

[13] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[14] C. Z. Janikow and Z. Michalewicz, "A specialised genetic algorithm for numerical optimization problems," in *Proc. Second Int. IEEE Conf. Tools Artif. Intell.*, 1990, pp. 798–804.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[16] V. B. Lawrence and A. C. Salazar, "Finite precision design of linear-phase FIR filters," *Bell Syst. Tech. J.*, vol. 59, no. 9, pp. 1575–1598, 1980.

[17] C. Fonseca and P. Fleming, "A review of current multi-objective optimization methods," *Evol. Comput.*, vol. 3, no. 1, pp. 1–16, 1995.

Stephen P. Harris received the B.A. degree with honors in chemistry in 1992 from Oxford University, Oxford, U.K. He is currently working toward the Ph.D. degree from the University of Plymouth, Plymouth, U.K.

His research interest is in the application of genetic algorithms to digital filter design. His final year project at Oxford concerned the use of genetic algorithms in pollution measurement and resulted in a number of publications.

Emmanuel C. Ifeachor received the B.Sc. (Hons.) degree in communication engineering from Plymouth University (formerly Plymouth Polytechnic), Plymouth, U.K., in 1980, the M.Sc. degree in communication engineering and the DIC degree from Imperial College, London, U.K., in 1981, and the Ph.D. degree in medical electronics from Plymouth University in 1985.

He did his industrial training with GEC Telecommunications Ltd., Coventry, U.K., in 1978 and 1979. He served as a Research Assistant in the Department of Communication Engineering, Plymouth University, and became a Lecturer in 1985. He is currently the Wandel and Goltermann Professor of Intelligent Electronics Systems and Head of the School of Electronic Communication and Electrical Engineering at Plymouth University. His major research interests include digital signal processing and intelligent systems with applications in biomedicine, audioengineering, and telecommunicatons. He has published over 80 technical papers, coauthored the textbook *Digital Signal Processing—A Practical Approach* (Reading, MA: Addison–Wesley, 1993), and has edited three books.

Dr. Ifeachor received the IEE Dr. V. K. Zworykin Premium for his work on fetal ECG analysis and currently serves on the IEE Professional Group on Biomedical Engineering.