EVOLVING OPTIMAL IIR AND ADAPTIVE FILTERS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND ELECTRICAL

ENGINEERING

OF GLASGOW UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Sathiaseelan Sundaralingam

June 1999

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

In this thesis, current digital filter design techniques are critically reviewed and problems associated with computational cost, complexity, frequency response and speed of convergence, identified. Based on this, a globally optimal, fine-tuned and efficient evolutionary hybrid technique has been developed to automate and optimise infinite impulse response (IIR) and adaptive filter design. The proposed hybrid design approach employs an evolutionary algorithm (EA) as a global search tool and a least mean square (LMS) algorithm, whenever appropriate, as a fine-tuner. This permits optimal and real-time tracking of time varying changes in nonstationary environments as widely encountered in telecommunications.

In the development, various improvements on existing algorithms are made, including those on components of EAs, LMS algorithm and the filter structures. The aims are to be able to evolve direct form IIR structures using simple stability monitoring techniques, to improve local fine-tuning performance and to avoid premature convergence. To evolve complex phenotype chromosomes that are needed by complex IIR filters, a novel method of crossover operation is developed. This is a variation of the standard uniform crossover in which the split points are considered to combine uniquely as indivisible floating-point complex valued genes. The split-point crossover operation produces more new members than the standard crossover operation, and hence provides a faster rate of convergence and avoids premature convergence. The EAs have been particularly designed for small population sizes and to reduce premature convergence, a new operator is designed to introduce new members into the population during

evolution.

Two techniques are investigated in the design of linear adaptive IIR digital filters, namely, the *pole design method* and the *coefficient design method.* The pole design method provides filter stability throughout the genetic search without requiring a variety of stability monitoring techniques. The coefficient design method uses simple stability guaranteeing techniques, which also improves the rate of convergence of the EAs. With the hybrid technique, complex-coefficient filters have been designed successfully and globally optimal and adaptive filters have been achieved.

The developed methodologies and designs are verified using higher order complex IIR systems and, for adaptation, inverse system modelling that is synonymous with channel equalising filters operating in multipath environments. Here adaptive complex parameters become possible to equalise amplitude and phase distortions of the received signals. Various stability-ensuring techniques are investigated extensively and their convergence performances are compared with the proposed method. The proposed hybrid, global and fine design technique is applied to solve adaptive channel equalisation and noise cancellation problems commonly existing in telecommunications.

Acknowledgements

I wish to convey my deep sense of gratitude to my supervisor Dr. Yun Li for his support, encouragement and valuable suggestions during this research.

I must be thankful to Engineering & Physical Sciences Research Council (EPSRC) and the Department of Electrical & Electronics Engineering, University of Glasgow who have given me this great opportunity to do this research with an appropriate funding.

I would like to thank the following reviewers, who made valuable suggestions, as I publish this work in several occasions: DR. David B. Fogel (Secretary of Evolutionary Programming Society), Dr. Bill Porto (Treasurer of Evolutionary Programming Society), Dr. S. Theodoridis (General Chairman of European Association for Signal Processing) and Dr. N. Saravanan (Vice President of Evolutionary Programming Society).

Thanks to my colleagues, especially, Craig Sloragh, Steve Fulton, Alexander Andreas Duncan, Richard McAleer, Dr. Anna I. Esparcia-Alcázar, Ian Paterson and other departmental staff, who have directly and indirectly contributed in my research.

Finally, my heartiest thanks go to my wife for her sacrifices and patience during past three years of my research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Importance of IIR and Adaptive Filtering

With the increasing power of micro processors, digital signal processing (DSP) has now played an important role in many engineering applications [82, 6, 76, 8, 77, 24, 19, 15]. Adaptive signal processing has evolved from techniques developed to enable adaptive control of time varying systems. It is now routinely used in a wide range of signal processing systems. Radar, equalising filters in high-speed MODEMS, echo-cancellation in speakerphones, interference removal in real-time medical imaging, and beam-forming in radio astronomy, all are examples of commercial systems that rely on adaptive filtering in one way or another [82, 74, 41, 56, 81]. Its major advantages over classical and linear time-invariant (LTI) signal processing is that it can cope with nonstationary signals whose statistical properties vary with time and that it demands a low amount of a priori information.

Adaptive signal processing has gained much popularity and has broadened the scope of digital signal processing. For practical reasons, however, the first generation of adaptive systems generally employ the basic finite impulse response (FIR) linear filter structure [82, 6, 76, 8, 77, 7]. As such, they have certain performance limitations. For example, the exact restoration of a received signal corrupted by multipath distortion is hardly possible with an FIR

structure [82, 6, 76, 37, 63, 26]. Consequently, the usage of adaptive FIR filters is now relatively low in certain areas of signal processing.

The applications of adaptive filtering approaches can be extended in all areas by developing computationally efficient adaptive infinite impulse response (IIR) filtering algorithms. The primary advantage of an adaptive IIR filter over a FIR one is that it provides significantly better performance with the same number of coefficients. For example, a desired response or, equivalently, its frequency response can be approximated more effectively by the output of a filter that has both poles and zeros compared to one that has only zeros [63]. This is because the output feedback can generate an infinite impulse response with only a finite number of parameters. Therefore, the computational costs for implementing such a system are significantly less compared with an FIR filter giving the same amount of performance (e.g. frequency response, phase response, etc.). These benefits make possible the widespread applications of adaptive filters and also highlight the importance of studying these systems with improved design and performance.

1.2 Current Design Problems

Compared with FIR filters, an IIR filter does introduce poles and hence stability problems into the system. To resolve stability and flexibility problems associated with direct form IIR filters, alternative realisations such as parallel, cascade and lattice forms can be considered. These structures offer simple stability monitoring without the complexity required by the direct form. Among other realisations studied, parallel and lattice forms have appeared to be relatively more robust [64, 31, 48, 62]. Unfortunately, the parallel or cascade structures can result in relatively more multiple optima that can arise from rearranging the poles among different sections [5, 63]. This property leads to a major difficulty in designing IIR filters and degrades the rate of convergence of an adaptive algorithm [63, 3].

Widely used algorithms for adaptive FIR and IIR filtering applications are mainly based on *least squares* (LS) and *gradient-guidance* techniques [82, 6]. Compared with a gradient-guidance technique, the least squares approach offers faster convergence but is numerically ill conditioned and computationally more expensive [24, 22, 1, 65, 53]. Furthermore, LS techniques are more constrained to a specific network topology and are considered unsuitable for complex structures having a large amount of recursion [38]. On the other hand, gradient-guidance technique, such as the least mean squares (LMS), has been mostly used in adaptive control and filtering because of its simplicity of implementation and a computational efficiency that is proportional to the number of adjustable parameters [82, 6, 22]. However, the LMS algorithm has two major drawbacks: slow rate of convergence, and sensitivity to the eigenvalue spread of the correlation matrix of the input signal vector [22]. In addition, current IIR-LMS often fails to converge to an optimum or near optimum when the associated error function is multimodal with respect to filter coefficients [63, 64, 31, 11, 61, 33, 22].

Traditional design methods of filtering systems use magnitude response as design specification and accept the phase response as is obtained from the design methodology [55, 66]. Since the magnitude and phase characteristics of LTI systems are interrelated, however, the design needs to consider both magnitude and phase responses as the design parameters in order to achieve a good set of filter coefficients [52]. Given the magnitude and group delay, an IIR filter design problem is often a multiple objective one [74]. Expressions of the magnitude and delay errors, e_m and e_d, with respect to a given specification may then be formulated as objectives in terms of filter coefficients. Simultaneous minimisation of competing objectives with e_m and e_d can hardly be obtained through conventional optimisation techniques [52, 74, 2].

In summary, conventional methods for designing IIR and adaptive filters encounter the

- Stability Constraint;

- Multimodality; and
- Multiple Objective;

problems. This gives the motivation for investigating the population based evolutionary search approach [78, 2, 39].

1.3 Evolutionary Solutions to IIR Filtering

Currently, various evolutionary algorithm (EA) based approach have been proposed for digital filtering [37, 51, 38, 35, 79, 70, 44, 47, 3, 60, 73, 45, 49, 2, 58, 25, 52, 46, 78, 10, 50]. Several evolutionary programming (EP) approaches are shown to design IIR digital filters using various realisations [3, 46, 78]. Error surfaces, with modelling IIR filters in various realisations (direct, cascade, parallel and lattice) are studied [3]. Various evolutionary and genetic search approaches to adaptive IIR filtering are published [37, 51, 44, 73, 46, 10]. Genetic evolution of the filter coefficients during the adaptation phase of the gradient lattice algorithm has also been shown to improve the convergence rate of the gradient method and provide global search capability to give lower error performance [51, 50]. Genetic algorithms (GAs) have been used to avoid premature convergence and to achieve asymptotic convergence behaviour [2]. They are successfully used to design low complexity, primitive operator, digital FIR filters [58].

The methods published to date are for the real form adaptive filters. In signal processing, the concept of complex time-waveform is increasingly used with the help of a Hilbert transformer to reduce the sampling rate [76, 6, 55]. The reduced sampling rate requires slower logic circuitry, which in turn reduces size and the costs when implementing a DSP system. The concept of complex signals is also used in quadrate amplitude modulation (QAM) systems where both in-phase and quadrate carriers can be viewed as real and imaginary components and the associated operation can be done in the complex form [56, 76]. Therefore, an

evolutionary method to evolve complex parameters would be useful for signal processing applications.

A variety of stability monitoring techniques is employed with high computational costs to evolve feedback coefficients. Hence, alternate realisations are still being used to avoid the computational burdens of stability monitoring that are required by direct form IIR structures. As mentioned, alternative realisation such as parallel and cascade provide slower rate of convergence. The performance becomes worse when the subsections are identically initialised. Although lattice forms offer a simple test of stability, it is more complex than direct and parallel form structures [63]. For example, the gradient component for a feedback coefficient requires a separate lattice filter using intermediate signals of the adaptive lattice filter as input signals.

Although various remedies are proposed in EAs, large population sizes and high mutation distributions are often used to avoid premature convergence. Increasing population sizes is computationally expensive and higher mutation distributions can degrade the convergence performance. An optimal approach to solve premature convergence is somehow more crucial. Also, evolutionary techniques, in general, provide a better global search capability, but fine-tuning cannot be guaranteed. Various approaches have been attempted to achieve fine-tuning, but non-of them meets all of the desired characteristics [25]. Therefore, simulated annealing (SA) based GAs have been considered to obtain fine-tuning on globally optimised coefficients where simulated annealing is shown as being one of the best methods to achieve local search and hence fine-tuning [73, 79, 24].

Another major drawback of the current EA based techniques is that evolutionary algorithms cannot be used for on-line adaptation. Many signal processing applications require on-line adaptation in which the parameters can be adapted as the new data are being captured. For example, adaptive channel equalising filters used in multipath environments need to be updated at frequent intervals to equalise the channel characteristics, which can change with time. These existing problems need to be solved.

1.4 Research Goals and Methodology

The work presented in this thesis is to solve these problems and achieve a number of research goals, which are:

1. To develop an EA based technique to solve the complex digital filter design problem;

2. To analyse various stability ensuring techniques that can be applied to evolve IIR and adaptive filters;

3. To develop a simple method to ensure IIR filter stability while improving the convergence rate of the EA;

4. To develop an EA-LMS hybrid approach to adaptive filtering;

5. To investigate various cost functions that can be employed with the proposed EA and filtering techniques;

6. To investigate various approaches that can be used along with the standard genetic operators with a view to avoiding premature convergence when floating-point EAs are implemented with a small population size;

7. To test that the proposed operator can avoid premature convergence when evolving complex IIR systems which have severe multiple local optima in the objective functions;

8. To improve the floating-point EA techniques to achieve high speed, high precision and high accuracy, with less computational cost and reduced complexity for signal processing applications.

9. To assess and apply the improved EA in system modelling with desired performance criteria for adaptation;

10. To use the proposed techniques to solve various DSP problems such as Multipath Channel Equalisation and Noise Cancellation with desired performance characteristics.

The concept of complex filters is used with an aim to reduce the computational requirements of adaptive channel equalisation filters employed in communication systems [76, 6]. To design complex adaptive IIR digital filters with improved performance, the EA-LMS hybrid approach is aimed to provide a better global search as well as a good local search capability as the algorithm approaches the global region. Small population sizes and simple stability ensuring methods are used. The chromosomes represent a population of filter objects in the same manner as they are implemented. Two design approaches namely *pole design* and *coefficient design* methods are considered in evolving direct form IIR filters. The pole design method evolves direct form filters without employing any stability ensuring methods. Coefficient design method, on the other hand, employs simple stability monitoring techniques. Least mean square algorithm is chosen to achieve fine-tuning as the algorithm enters the global region. Moreover, the LMS algorithm is used to track the time varying parameters when the adaptive filters are designed in nonstationary.

Although the pole design method avoids stability monitoring techniques, a major drawback to be expected in this approach is that it may provides multiple local optima with the same fitness values that can be obtained when reordering the poles within the filter. i.e. the poles can be rearranged in any manner to obtain the same fitness value. This may lead to slower convergence when crossover operation is employed between feedback sections. To solve this problem, coefficients can be designed directly from a population. Coefficient design method, therefore, will require a robust method to ensure filter stability during adaptation.

Evolutionary algorithms are dependent on their operators, as improper design of these operators can result in poor convergence performance. All of the special features of crossover operations cannot be achieved for the floating-point representation. Furthermore, the mutation of floating-point EAs is controlled by three elements rather than by a single factor as in binary GAs. These factors are known as probability of mutation, variance, and type of distribution.

The probability of mutation dictates the rate at which mutation is performed, the variance corresponds to the amount of variations and the type of distribution decides how the probability of a function defining mutation is distributed amongst the function (e.g. Gaussian, Uniform, etc.). In binary GAs, the mutation is only controlled by the probability of mutation. Introducing improper variance can cause the algorithm's convergence into inappropriate directions. This may cause either premature convergence or high population diversity. A major goal of this work is therefore to develop and improve EA techniques that will perform genetic evolution of complex, direct form IIR digital filters with small population-sizes, simple methods of ensuring filter stability and better fine-tuning capability.

1.5 Contributions

Major contributions of the work presented in this thesis are summarised below:

1. The goals 1 - 10 listed on pages 7 - 8 are achieved, providing novel methodologies, designs, applications and results.

2. Two design methodologies, pole and coefficient design methods, are developed.

3. Advantages and drawbacks of each design method are investigated.

4. A novel approach is shown to correct unstable coefficients while improving the rate of convergence.

5. The problems associated with the alternative IIR filter realisations are illustrated using the convergence curves which are obtained while modelling various IIR systems.

6. This work also provides techniques for tracking IIR systems when the coefficients to be optimised are time-variant.

7. The performances of IIR and FIR filters are compared using channel equalising filters.

8. Least mean square algorithm is demonstrated to show how the adaptive constant can affect the tracking capability of adaptive filters when equalising various communication channels.

9. Split-point crossover which combines uniquely as indivisible floating-point genes is proposed and demonstrated as being a better method than standard crossover.

10. A new operator is proposed and used along with the standard genetic operators to avoid premature convergence while increasing the speed of the EAs by introducing more new members at each generation.

11. The evolving filters and the associated genetic operations and functions are developed as modular C++ classes and functions.

Along with the above mentioned contributions, this work has been published on several occasions and a few more papers are being submitted:

1. Sundaralingam and K. C. Sharman, "Evolving Adaptive Complex IIR Stuctures," *In Proc. of the IX^{th} Annual Europeon Signal Processing Conference, EUSIPCO-98*, Volume II, pages 753-756, Greece, September 1998.

2. Sundaralingam and K. C. Sharman, "Evolving Filters in Multipath Environments," *In Proc. of the Seventh Annual Conference on Evolutionary Programing, EP-98*, pages 397-407, San Diego, March 1998.

3. S. Sundaralingam and K. C. Sharman, "Genetic Evolution of Adaptive Filters," *In Proc. of the International Conference on DSP, DSP-97*, pages 47-53, London, December 1997.

4. S. Sundaralingam and Y. Li, "Hybrid Approach to Adaptive IIR Filter Design," *Submitted to IEEE Trans. On Acoustics, Speech & Signal Processing*, May 1999.

5. S. Sundaralingam and Y. Li, "Direct Form Digital IIR Filter Design via Global Evolution," *Submitted to IEEE Trans. On Acoustics, Speech & Signal Processing*, May 1999.

1.6 Outline of Dissertation

The rest of this thesis is organised as follows:

Chapter 2 presents the extensive and critical review of the various types of classical adaptive IIR filtering algorithms used and/ or proposed to date and will investigate the issues favouring or limiting the applications of each.

Chapter 3 provides an extensive study on evolutionary techniques, which are now currently being used, and will inspect the issues of limiting each application.

Chapter 4 presents the recent development of floating-point EA techniques for designing complex IIR digital filters. In particular, this chapter will show the design techniques of poles without employing any stability monitoring. A method of using complex parameters directly in the chromosome's structures will be described. Along with this representation, the detailed functional description of genetic operations will be given. A new crossover scheme for floating-point EA will be shown. The procedures of performing mutation on these floating-point structures will be defined. Various cost functions will be discussed and their performances will be investigated. Among various selection schemes proposed, the reason for choosing tournament selection will be described. Finally, this chapter will show the modelling of simple direct and inverse systems to demonstrate the proposed GA techniques.

Chapter 5 presents the techniques of designing the coefficients of direct-form IIR digital filters with simple stability monitoring techniques. This chapter will also outline the problems that can arise from premature convergence. A variety of remedies will be discussed to resolve premature convergence in floating-point EAs. This chapter will introduce a new genetic operator called *immigrant* that can be used along with the standard genetic operators such as crossover and

mutation to avoid premature convergence within small population sizes.

Chapter 6 shows how fine-tuning can be introduced and improved with the use of gradient based algorithm where feasible. Techniques of designing direct form IIR digital filters using a hybrid algorithm are given with various applications. A variety of system modelling examples are shown with fine-tuning performance of the proposed approach. Furthermore, this chapter will provide the techniques of adapting nonstationary systems with the use of online algorithm.

Chapter 7 presents some adaptive filtering applications where the recently developed method has been successfully applied, with convincing results. In particular, comparative results will be given for channel equalisation to compare the frequency and BER performances of IIR filters against FIR equalisers. These results demonstrate the ability of the developed hybrid techniques to equalise or model complex systems.

Finally, in Chapter 8, conclusions will be drawn with a summary of salient features of this research, while indicating the areas of further research.

Chapter 2

Adaptive Filters and Design

Adaptive IIR digital filters and their conventional design techniques are highlighted in this chapter. Algorithms that have been used in such designs mainly belong to gradient-guidance and least square techniques. These algorithms are in general, well established for adaptive FIR filters and are very suitable when the error surface is quadratic or unimodal, because of their simplicity [82, 6, 7].

This chapter provides an overview of various classical algorithms for adaptive IIR filtering. In particular, the algorithms, which are based on LMS and RLS, techniques have been discussed. Recursive least square techniques have been shown to improve the performance of the LMS algorithms [63, 22, 79]. The aim of this chapter is to outline the basic principles of designing classical algorithms, which will be employed in Chapters 6 and 7 to achieve various performance improvements with evolutionary approach. This chapter also outlines the issues, which limit performance when designing the adaptive filters using these classical methods.

The rest of this chapter is organised as follows: The basic materials needed to understand the current research is clarified in the following section. Followed by this foundation, LMS and RLS adaptive IIR filtering techniques to date are thoroughly investigated. The issues favouring or limiting these techniques are revealed more clearly. Finally, a summary of these techniques is given, indicating the merits and drawbacks of each system.

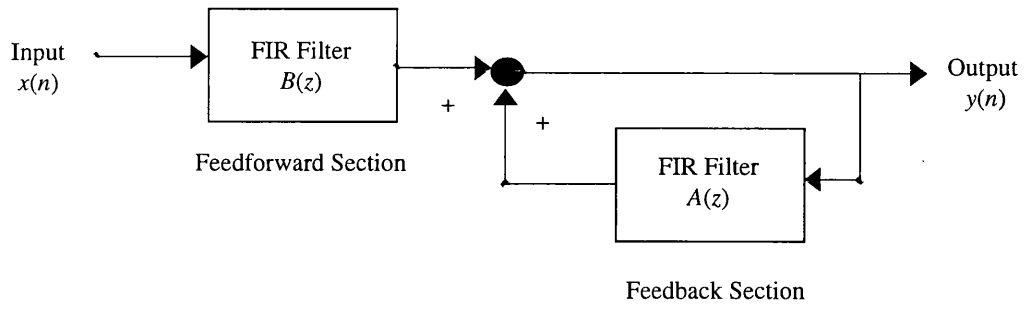

Figure 2.1: A recursive filter.

2.1 IIR Filters and Stability

An infinite impulse response digital filter, which is also known as autoregressive moving-average (ARMA) model, has both a recursive and a non-recursive section as illustrated in Figure 2.1. In this figure an IIR filter is viewed as two FIR filters, one of which is connected in a feedback loop. The output, $y(n)$, and the transfer function, $H(z)$, of this model can be written in the following mathematical notations:

$$\begin{aligned} y(n) &= b_0x(n) + b_1x(n-1) + \cdots \\ &\quad + b_{M-1}x(n-M+1) + a_1y(n-1) + a_2y(n-2) + \cdots \\ &\quad + a_Ly(n-N+1) \end{aligned} \tag{2.1}$$

$$H(z) = \frac{B(z)}{1-A(z)} = \frac{\sum_{i=0}^{M-1} b_i z^{-i}}{1 - \sum_{j=1}^{N-1} a_j z^{-j}} \tag{2.2}$$

where $\{a_j\}_{j=1}^{N-1}$ and $\{b_i\}_{i=0}^{i=M-1}$ are $N-1$ and M adjustable denominator and numerator coefficients respectively while $\{A(z)\}$ and $\{B(z)\}$ are the polynomials of z.

The major design concern for an IIR filter is to ensure that the recursive (feedback) section is stable. The stability of a recursive filter requires that all poles of the filter have a magnitude less than 1. The stable region corresponding to this condition can be represented as a unit circle in a complex z-plane as illustrated in Figure 2.2.

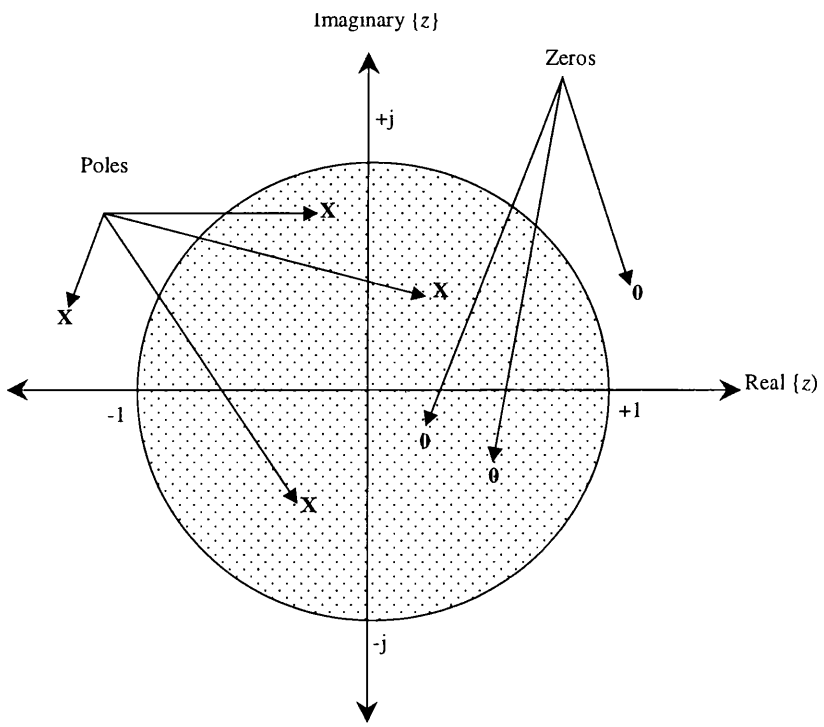

Figure 2.2: Illustration of unit circle of the complex z-plane. In physical systems the complex poles appear as complex conjugate pairs, which are symmetric about the real axis.

The pole polynomials of a recursive section can be obtained through factorisation. Factoring polynomials, however, of order greater than 2 are non-trivial. Hence stability monitoring is difficult. The poles may be complex numbers - however these complex numbers will in general occur in conjugate pairs given that the filter coefficients are real numbers. Therefore, physically realisable systems (real systems) can have complex poles but appear as complex conjugate pairs in a z-pane. However, complex filters are widely used in many areas of signal processing in order to reduce the signal's bandwidth and hence the sampling rate [55, 76]. Furthermore, most speech-band modems confirm to CCITT-recommended modulation formats, which, for the high speed modems needing equalisers, involve either pure phase modulation or combined phase and amplitude modulation. Both types of modulation can be viewed as forms of quadrature amplitude modulation (QAM) and require the use of complex IIR equalisers [56, 6]. The theory behind complex filtering techniques can be found in [55, 76, 6]. If the complex filters are considered the poles need not be appeared as complex conjugate pairs.

Although IIR filters have some favourable properties, they also have some undesirable properties. In general IIR filters are not linear phase and therefore may introduce phase distortion [55, 66]. Consequently, the use of IIR filters in phase sensitive applications such as data communications, and multi-channel high-fidelity audio should be carefully considered. In some situations however, by careful design IIR filters can often be made to approximate linear phase in the chosen pass-band [55].

2.2 Adaptive IIR Filters and Design

An adaptive IIR digital filter is comprised of two basic components: a discrete time varying IIR filter with input $x(n)$ and output $y(n)$, and a control algorithm that adjusts the filter coefficients to optimise some performance criterion that

is based on prediction error, $e(n)$, given by

$$e(n) = d(n) - y(n) \tag{2.3}$$

where $d(n)$ is the desired signal. Considering this error, the goal of the adaptive filters is to find the optimum setting of parameters defining the system so as to minimise suitably defined cost functions. One commonly used cost function is the mean square error (MSE) given by

$$e = \mathrm{E}\,|e(n)|^2 \tag{2.4}$$

where E is the statistical expectation.

The prediction error can be formulated either from equation or output error methods [63, 26, 44, 82]. It has been shown in the literature that the algorithms relating to equation error formulation have severe biasing problems and cause inaccurate estimates of parameters [63]. The term bias refers to the difference between actual parameter values to be found and its estimates. The cost functions based on output error formulation are non-quadratic concerning the filter coefficients and may have multiple local optima which in turn cause the gradient algorithm to get stuck in a local solution [63]. Nevertheless, the output error formulations are now widely used, as it provides more accurate estimates than an equivalent equation error adaptive IIR filter [38].

An MSE cost function based on output error formulation is depicted in Figure 2.3. The mean squared value is plotted against adaptive filter coefficients when an adaptive filter,

$$A(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{2.5}$$

is used to model an unknown system,

$$G(z) = \frac{0.5 - 0.4 z^{-1} + 0.89 z^{-2}}{1 - 1.4 z^{-1} + 0.98 z^{-2}} \tag{2.6}$$

by minimising the MSE error

$$e = \frac{1}{w} \sum_{n=0}^{w-1} \{y(n) - \hat{y}(n)\}^2 \tag{2.7}$$

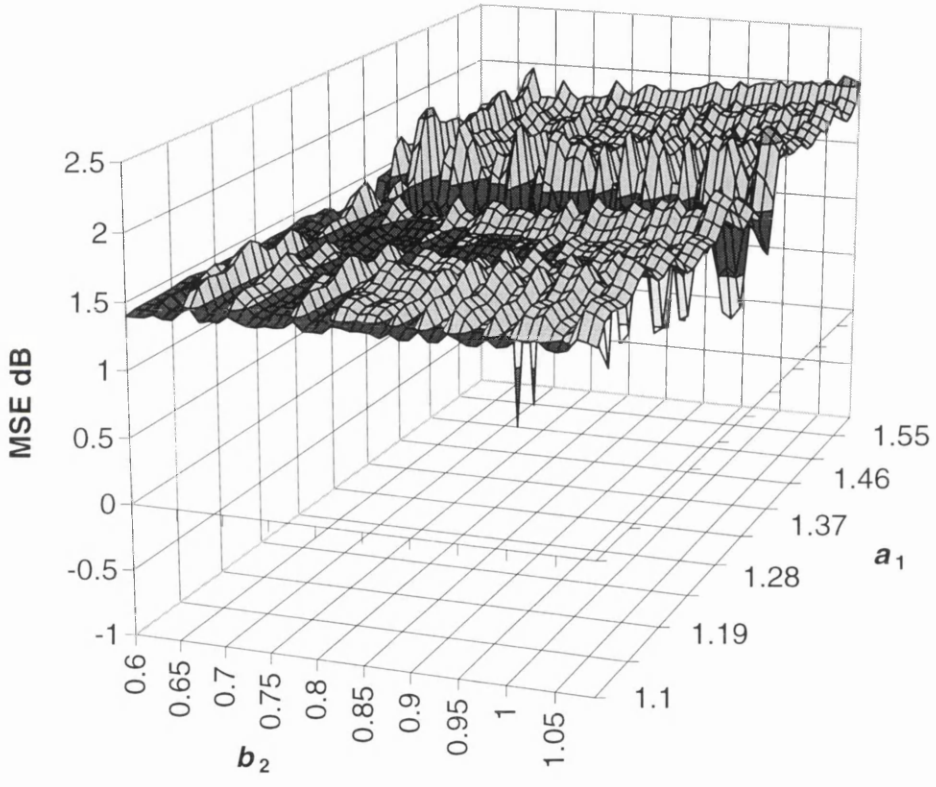

Figure 2.3: An example of multimodal error performance surface of a second-order adaptive IIR system. Mean square error surface is shown against b_2 and a_1 for various choices of b_0, while modelling the second-order IIR system given in equation (2.6)

Given a training set of $w = 200$ samples, the MSE values of the coefficients of the adaptive filter were calculated using the training set $\{\hat{y}(n), y(n)\}$, where $y(n)$ represents the output of the unknown system and $\hat{y}(n)$ represents the output of the adaptive filter. For the sake of simplicity, three coefficients are varied whilst other parameters are kept constant. The error surface shown in this figure is multimodal and has multiple local optima. Another example is shown in Appendix A. The error performance surface equation based on output error methods of an adaptive IIR filter is non-quadratic and a straightforward-gradient based solution does not exist. For an adaptive FIR filter, the minimum MSE is found simply by locating the bottom of the hyperparaboloid. Mathematically this is achieved by finding the zero gradient vectors using vector calculus techniques. It is important to note that the adaptation of parameters by minimising MSE is valid only if the statistics of $\{x(n)\}$ and $\{d(n)\}$ are wide sense stationary and ergodic process [76].

The length of the training set plays an important role in the convergence, as too small training set may result in convergence to incorrect parameter values. Visualising the error surface around the best solution can easily show the reason for such an inaccurate convergence. For example, consider modelling of a second-order ARMA model represented by the following z-transfer function,

$$G(z) = \frac{0.5 - 0.4z^{-1} + 0.9z^{-2}}{(1 - 0.7z^{-1})(1 + 0.65z^{-1})} \tag{2.8}$$

by an adaptive filter

$$A(z) = \frac{0.5 - 0.4z^{-1} + b_2(n)z^{-2}}{(1 - 0.7z^{-1})(1 - p_2(n)z^{-1})} \tag{2.9}$$

For the sake of simplicity, we assumed that all parameters of the adaptive filter except $b_2(n)$ and $p_2(n)$ were assigned to the optimum values. The model and the adaptive system were excited by the same input, which is a Gaussian variable with zero mean and unit variance. Mean square error performance was obtained for various choices of $b_2(n)$ and $p_2(n)$.

Figure 2.4(a) shows the 3-D plot showing the local MSE surface around the best solution when $w = 100$. Figure 2.4(b) on the other hand, shows the plot

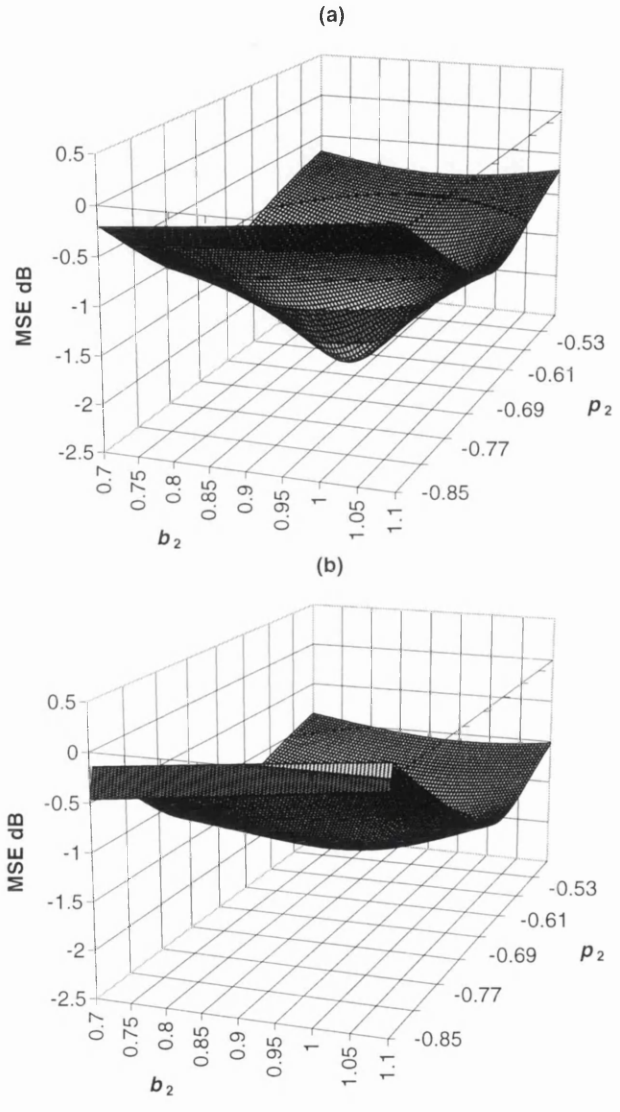

Figure 2.4: Local mean square error surface while modelling an ARMA model by an adaptive filter using different number of samples. (a) shows the local MSE when number of training data was 100. (b) shows the MSE obtained when number of training signals was 30.

obtained when $w = 30$. Mean square error shown in Figure 2.4(a) has minimum MSE (MMSE = -2.33073 dB) at $\{\hat{b}_2 = 0.89, \hat{p}_2 = -0.65\}$, whilst Figure 2.4(b) has its MMSE = -1.74582 dB at $\{\hat{b}_2 = 0.93, \hat{p}_2 = -0.62\}$. The MMSE obtained for $w = 100$ is very closer to the optimum values which are located at $\{b_2 = 0.9, p_2 = -0.65\}$, but the solution obtained for $w = 30$ converges to inaccurate values which are clearly far away from the optimum point. Appendix A shows how these mesh plots were obtained.

Common algorithms that have found widespread applications are the LMS and the recursive least square [63, 47, 36]. In terms of computation and storage requirements, the LMS algorithm is the most efficient [60]. Furthermore, it does not suffer from the numerical instability problem inherent in the RLS algorithms [45]. For these reasons, the LMS algorithm has become the algorithm of first choice in many applications [82, 6, 76]. However, the RLS algorithms have superior convergence properties. Unfortunately its uses in signal processing applications have been relatively limited due to its higher computational requirements and numerical errors [53]. In recent years there has been renewed interest in the RLS algorithm, especially in its fast versions [36]. Stabilisation techniques that prevent numerical divergence without performance degradation have been recently proposed with added computations [1, 65].

2.3 Least Mean Square Algorithms for Adaptive IIR Filters

A well-known algorithm, which falls into gradient search category, is LMS. The LMS algorithm was the first developed by Widrow and co-workers in 1967 [82]. It is a practical method of obtaining estimates of the filter weights $\boldsymbol{\theta}(n)$. For example in FIR-LMS, the filter weights are updated according to,

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu \boldsymbol{x}(n) e(n) \tag{2.10}$$

where:

$$\boldsymbol{\theta}(n) = [w_0(n), w_1(n), \cdots, w_{M-1}(n)]^T \tag{2.11}$$

$$\boldsymbol{x}(n) = [x(n), x(n-1), \cdots, x(M-1)] \tag{2.12}$$

$$e(n) = y(n) - \boldsymbol{\theta}^T(n)\boldsymbol{x}(n) \tag{2.13}$$

μ is called step size or convergence factor, $\{w_i\}_{i=0}^{i=M-1}$ are the filter weights (feedforward components) and $\{\boldsymbol{x}(n)\}$ is the input signal vector. The LMS algorithm shown above does not require prior knowledge of the signal statistics, but instead uses their instantaneous estimates. The weights obtained by the LMS algorithm are only estimates, but these estimates improve gradually with time as the weights are adjusted and the filter learns the characteristics of the signal. The first generation of LMS algorithms was developed for adaptive FIR systems. However, the algorithm has now been widely employed in designing adaptive IIR structures.

Consequently, using the above background, various IIR-LMS algorithms have been investigated and as a result, the following achievements were realised [63]:

- Full Gradient IIR-LMS
- Simplified IIR-LMS
- Fentuch's IIR-LMS
- Filtered Error (FE) Algorithms
- Recursive Prediction Error (RPE)

2.3.1 Full Gradient IIR-LMS

The weight vector update for full gradient IIR-LMS is:

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) - 2\mu e(n)\frac{\partial y(n)}{\partial \boldsymbol{\theta}(n)} \tag{2.14}$$

The weight vector, $\boldsymbol{\theta}(n)$, can be given by

$$\boldsymbol{\theta}(n) = [a_1(n), \cdots, a_{N-1}(n), b_0(n), \cdots, b_{M-1}(n)]^T \tag{2.15}$$

where $\{a_i\}$ and $\{b_j\}$ represent the denominator and numerator coefficients respectively while T denotes the vector transformation. The gradients of the filter output $(y(n))$ with respect to the filter coefficients can be written more compactly as

$$\frac{\partial y(n)}{\partial \boldsymbol{\theta}(n)} = \boldsymbol{\Delta}(n) = [\alpha_1(n), \cdots, \alpha_{N-1}(n), \beta_0(n), \cdots, \beta_{M-1}(n)] \tag{2.16}$$

where $\{\alpha_l(n)\}_{l=1}^{N-1}$ and $\{\beta_m(n)\}_{m=0}^{M-1}$ are

$$\alpha_l(n) = \frac{\partial y(n)}{\partial a_l(n)} \quad \text{for } 1 \leq l \leq N-1 \tag{2.17}$$

and

$$\beta_m(n) = \frac{\partial y(n)}{\partial b_m(n)} \quad \text{for } 0 \leq m \leq M-1 \tag{2.18}$$

If the coefficients adapt slowly, then the following approximation can be made:

$$\boldsymbol{\theta}(n) \approx \boldsymbol{\theta}(n-1) \approx \cdots \approx \boldsymbol{\theta}(n-N-M+1) \tag{2.19}$$

The slowly varying weight assumption can be some what forced by the algorithm designer by choosing a very small step size for the algorithm. The gradient components of the algorithm can be described as follows:

$$\begin{aligned} \beta_m(n) &= x(n-m) + \sum_{i=1}^{N-1} a_i(n) \frac{\partial y(n-i)}{\partial b_m(n-i)} \\ &= x(n-m) + \sum_{i=1}^{N-1} a_i(n) \beta_m(n-i) \end{aligned} \tag{2.20}$$

and

$$\begin{aligned} \alpha_l(n) &= y(n-l) + \sum_{i=1}^{N-1} a_i(n) \frac{\partial y(n-i)}{\partial a_l(n-i)} \\ &= y(n-l) + \sum_{i=1}^{N-1} a_i(n) \alpha_l(n-i) \end{aligned} \tag{2.21}$$

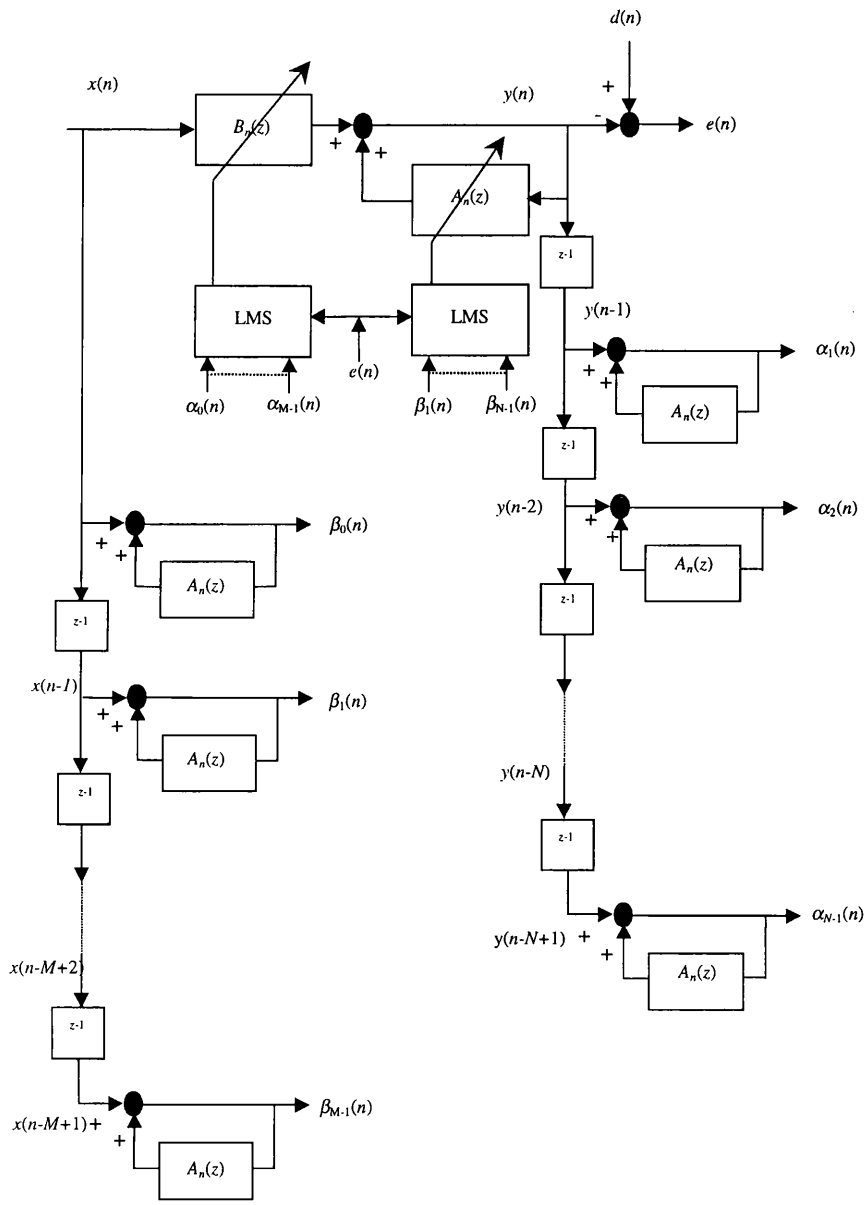

Figure 2.5: Signal flow graph of full gradient IIR-LMS.

The resulting algorithm is termed the full gradient IIR-LMS, because it uses all of the gradient components for filter implementation. The signal flow graph of this implementation is shown in Figure 2.5. It can be noted that, on each update, the current feedback weights $\{a_l(n)\}$ are used with current and past values of the input $\{\boldsymbol{x}(n)\}$ and with past outputs $\{\boldsymbol{y}(n)\}$ to produce the gradient components in equation 2.20 and 2.21 in order to update the LMS equation. Therefore the computation required at each new sample is of the order $N(N+M)+2M+N$ multiply accumulates (MACs) per iteration.

2.3.2 Simplified IIR-LMS Algorithm

In order to overcome the computational complexity of the full-gradient IIR-LMS algorithms, the following simplifications can be made by choosing a small step size α. Hence $\alpha_l(n)$ can be reasonably estimated using the data sample $\{y(n-1)\}$ and $A(z)$, similarly $\beta_m(n)$ can be calculated using the data sample $\{x(n)\}$ and $A(z)$. Initial gradient terms can be obtained from 2.20 and 2.21 by substituting $l=1$ and $m=0$ respectively.

$$\alpha_1(n) = y(n-1) + \sum_{i=1}^{N-1} a_i(n)\alpha_1(n-i) \tag{2.22}$$

$$\beta_0(n) = x(n) + \sum_{i=0}^{M-1} a_i(n)\beta_0(n-1) \tag{2.23}$$

From the initial terms, the other gradient components can be obtained in such a way that

$$\frac{\partial y(n)}{\partial a_l(n)} \approx \alpha_0(n-l) \quad \text{for } 1 \leq l \leq N-1 \tag{2.24}$$

$$\frac{\partial y(n)}{\partial b_m(n)} \approx \beta_0(n-m) \quad \text{for } 1 \leq m \leq M-1 \tag{2.25}$$

The signal flow graph representing this algorithm is shown in Figure 2.6. The complexity of this algorithm is now reduced to around $2(N+M)+2N$ MACS per iteration.

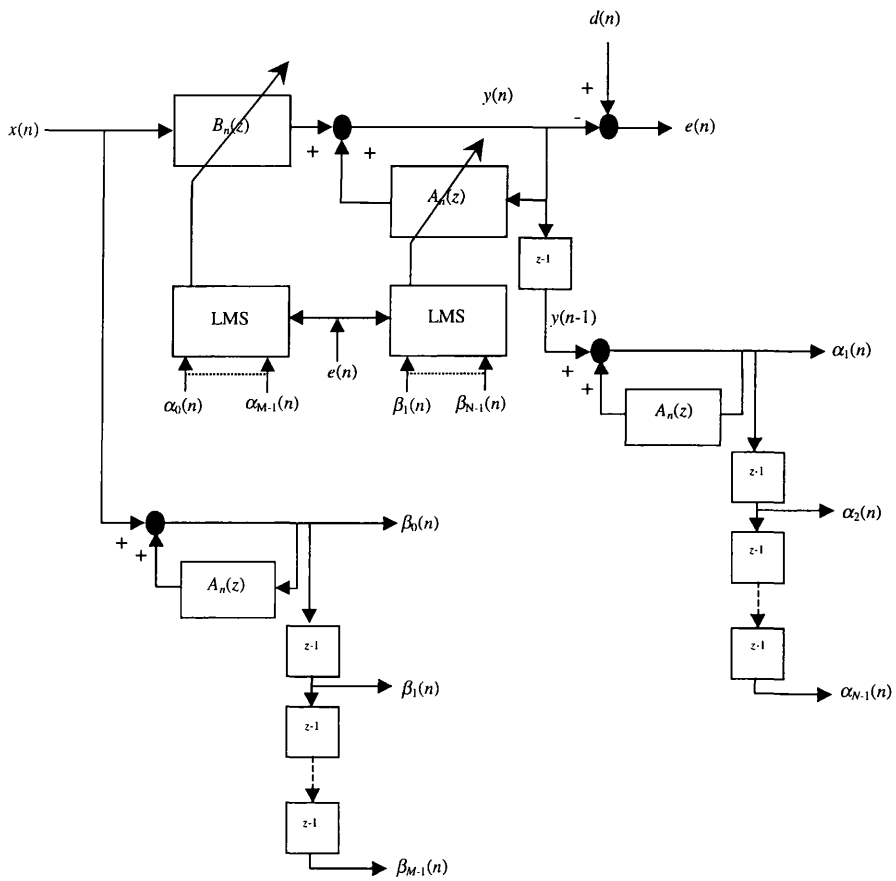

Figure 2.6: Signal flow graph: simplified IIR-LMS.

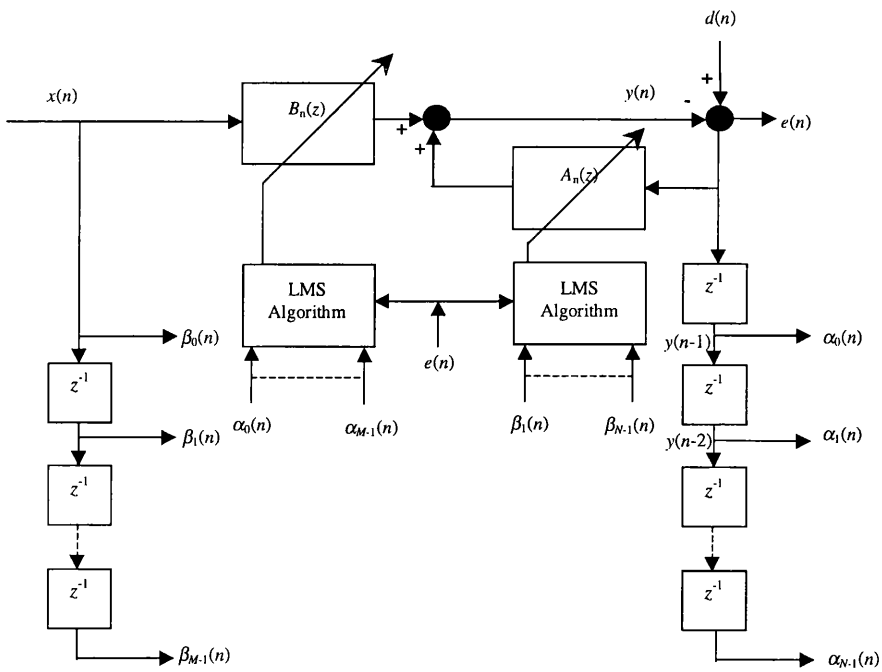

Figure 2.7: Signal flow graph: Feintuch's algorithm.

2.3.3 Feintuch's IIR-LMS Approach

Feintuch made a further simplification, where it is assumed all derivatives of past outputs with respect to current weights are zero. Therefore, the gradient components now become

$$\alpha_l(n) \approx y(n-l) \tag{2.26}$$

$$\beta_m(n) \approx x(n-m) \tag{2.27}$$

The computational complexity of this algorithm is now only $2(N+M)$. In typical applications, Feintuch will outperform the FIR-LMS if the system being identified has high level of recursion within it. However, simply using Feintuch to identify a non-recursive system, results in very little advantages. For a typical system, up to 10 or 20 even poles may be used with this algorithm.

Furthermore, Feintuch's algorithm has been found to be inherently more stable than the full gradient due to a tendency to adapt poles away from the unit circle towards the origin of the z-plane. The signal flow graph of Feintuch's algorithm is illustrated in Figure 2.7. This algorithm is still widely used and

analysed within the signal processing literature.

Even though Feintuch's algorithm is superior to other two algorithms, it may not converge to a minimum (local or global) of the MSE surface unless the denominator polynomials satisfy a Strictly Positive Real (SPR) condition. If this condition is not satisfied, the algorithm may converge to an arbitrary point on MSE surface and the overall performance may be unacceptable.

Strictly Positive Real Condition: The SPR condition is related to the concept of hyper-stability, which describes the output stability of feedback systems that may have both nonlinear and time varying components [73]. It is important to note that in addition to the SPR condition, hyper-stability requires certain restrictions on the data and on the adaptive filter configuration [63, 6]. It can easily be shown that the filter is bounded by the SPR condition if

$$\mathrm{Re}\left\{\frac{1+C(z)}{1-A_*(z)}\right\}-\gamma \geq 0 \qquad \text{for all } |z|=1 \tag{2.28}$$

where $\mathrm{Re}(u)$ denotes the real part of u while $\{1+C(z)\}$ is a filter employed to smooth $e(n)$ and $A_*(z)$ denotes the pole polynomials which are to be identified [63]. When $C=0$ and $\gamma=0.5$ it can be shown that

$$|A_*(z)| \leq 1 \qquad \text{for all } |z|=1 \tag{2.29}$$

for the SPR. The SPR region for various values of c_1 for a second-order system is illustrated in Figure 2.8. From this figure it can be seen that the SPR region can be purposely deformed by varying the value of c_1. This gives the motivation behind the FE algorithm given below.

2.3.4 Filtered-Error Algorithm

To overcome the convergence problem associated with the SPR condition, the error signal can be smoothed prior to the input into adaptive algorithms. Smoothing the error signal can assist the IIR algorithm in converging under the SPR

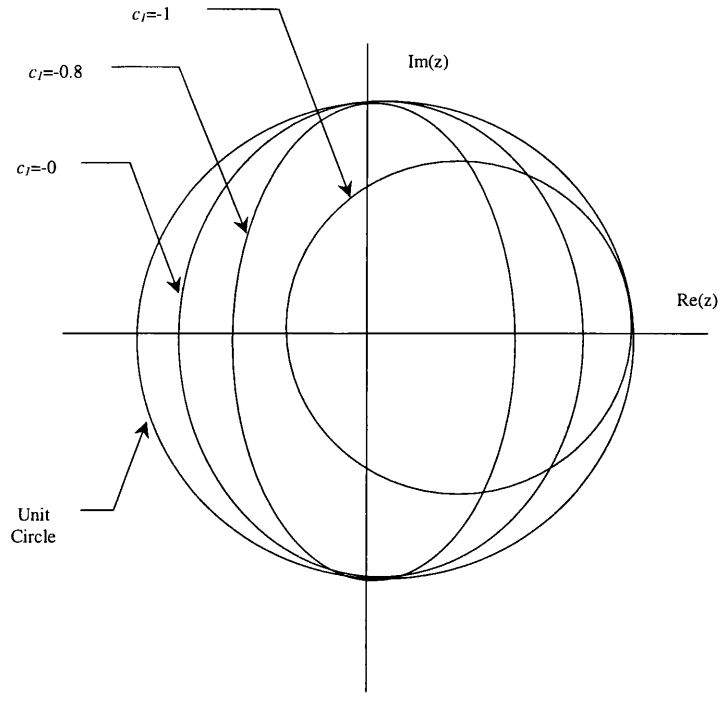

Figure 2.8: Second-order system, region of SPR with error smoothing.

conditions. A simple filtered error is given by

$$e_f(n) = \sum_{j=0}^{p} c_j e(n-j) \tag{2.30}$$

where c_j coefficients should represent a simple low-pass filtering function. However, in order to have satisfactory convergence it is desirable that

$$1 + C(z) = 1 - A_*(z) \tag{2.31}$$

However, such a condition is not always possible to achieve in practice since A_* is usually unknown. A signal flow graph illustrating FE algorithm is shown in Figure 2.9.

2.3.5 Recursive Prediction Error Algorithm

A major drawback found in all of the above algorithms is the poor convergence behaviour, which is caused by the statistical properties of the adaptive filter's input signals. These signals can be de-correlated in the time domain by a time varying step size matrix, which gives the motivation for RPE algorithm.

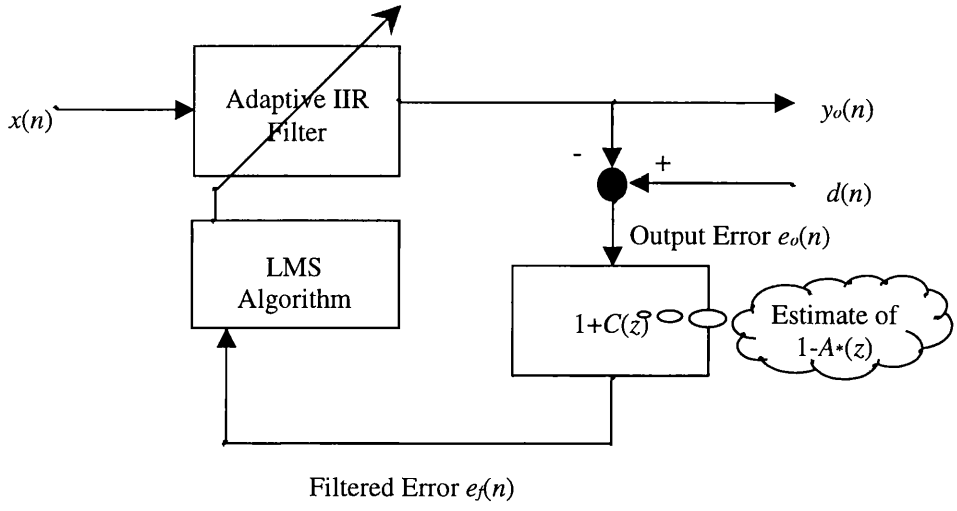

Figure 2.9: Filtered error algorithm. Filtering the output error with a filter of transfer function $1+C(z)$, which serves as an estimate of the underlying system poles $1 - A_*(z)$.

The RPE algorithm can be written as:

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu \boldsymbol{P}^{-1}(n+1) \frac{1}{1 - A(z)} \boldsymbol{\phi}(n) e(n) \tag{2.32}$$

where μ is a constant step size that controls the algorithm convergence rate, $\boldsymbol{\phi}(n)$ is the regression vector containing the data,

$$\boldsymbol{\phi}(n) = [y(n-1), \cdots, y(n-N+1), x(n), \cdots, x(n-M+1)]^T \tag{2.33}$$

and $\boldsymbol{P}^{-1}(n)$ is an estimate of the Hessian matrix (see [63] for details) updated according to

$$\boldsymbol{P}(n+1) = \lambda \boldsymbol{P}(n) + \mu \boldsymbol{\phi}_f(n) \boldsymbol{\phi}_f^T(n) \tag{2.34}$$

where λ is called forgetting factor and $\boldsymbol{\phi}_f(n)$ is filtered version of $\boldsymbol{\phi}(n)$ given by

$$\boldsymbol{\phi}_f(n) = \frac{1}{1 - A(z)} \boldsymbol{\phi}(n) \tag{2.35}$$

The motivation of using λ is to give greater importance to recent data than older data. Since computing the inverse of $\boldsymbol{P}$ is computationally expensive, $\boldsymbol{P}^{-1}$ is generally updated directly using the matrix inversion lemma [60]. In this case we have

$$\boldsymbol{P}^{-1}(n+1) = \frac{1}{\lambda} \left(\boldsymbol{P}^{-1}(n) - \frac{\boldsymbol{P}^{-1}(n) \boldsymbol{\phi}_f(n) \boldsymbol{\phi}_f^T(n) \boldsymbol{P}^{-1}(n)}{\frac{\lambda}{\mu} + \boldsymbol{\phi}_f^T \boldsymbol{P}^{-1}(n) \boldsymbol{\phi}_f(n)} \right) \tag{2.36}$$

Practical Limitations: There are, two main problems that may be encountered when the RLS technique is implemented directly. The first, referred to as *blow-up*, results of the $\boldsymbol{\phi}(n)$ is zero for a long time, when the matrix $\boldsymbol{P}(n)$ will grow exponentially as a result of division by λ (which is less than unity) at each sample point:

$$\lim_{n\to\infty} \boldsymbol{P}^{-1}(n+1) = \lim_{n\to\infty} \left(\frac{\boldsymbol{P}^{-1}(n)}{\lambda_n} \right) \tag{2.37}$$

The second problem with the RLS is its sensitivity to computer round off errors, which results in a negative definite $\boldsymbol{P}$ matrix and eventually to instability. For successful estimation of coefficients $\boldsymbol{\theta}$, it is necessary that the matrix $\boldsymbol{P}$ be positive definite. For these reasons, in the latter part of this thesis we consider RPE algorithm with $\boldsymbol{P}$ where $\boldsymbol{I}$ represents an identity matrix of size $(N+M-1)X(N+M-1)$. The RPE algorithm which uses $\boldsymbol{P}$=$\boldsymbol{I}$ is denoted as normalised RPE in the rest of this thesis.

Summarising, amongst IIR-LMS algorithms studied, Feintuch is the simplest to implement with $2(N+M)$ MACs per iteration. Also, Feintuch is found to be more stable than other IIR-LMS algorithms such as full gradient IIR-LMS and simplified gradient IIR-LMS. The full gradient IIR-LMS is rarely implemented due to its high level of computation required. The performance advantages for this high level of computation are also noted to be minimal. The simplified gradient IIR-LMS is also rarely implemented, with again minimal performance improvement seen over the computationally simpler Feintuch's IIR-LMS. The situations where the full gradient and simplified gradient IIR-LMS's have improved performance over Feintuch's IIR-LMS are usually in environments where the system being modelled has very high levels of feedback. Recursive prediction error algorithm can be applied if the system demands high tracking performance.

The following table summarises the gradient-guidance algorithms described above.

Algorithm	Processing Order	Main Features
Full Gradient	$N(N+M)+2M+N$	Best approximation, High level of computation
Simplified Gradient	$2(N+M)+2N$	Less computation than Full Gradient
Feintuch's	$2(N+M)$	Reduced computation, Poor approximation, Suffer from SPR condition
Filtered Error	$2(N+M)+p$	Higher computation than Feintuch's, Hyperstable
RPE (Simplified gradient)	$N(N+M)+3M+2N+3(N+M)^2$	High level of computation, Best approximation, Fast convergence, Suffer from blow-up error and sensitive to computer round-off errors

Table 2.1: Comparison between gradient-guidance algorithms.

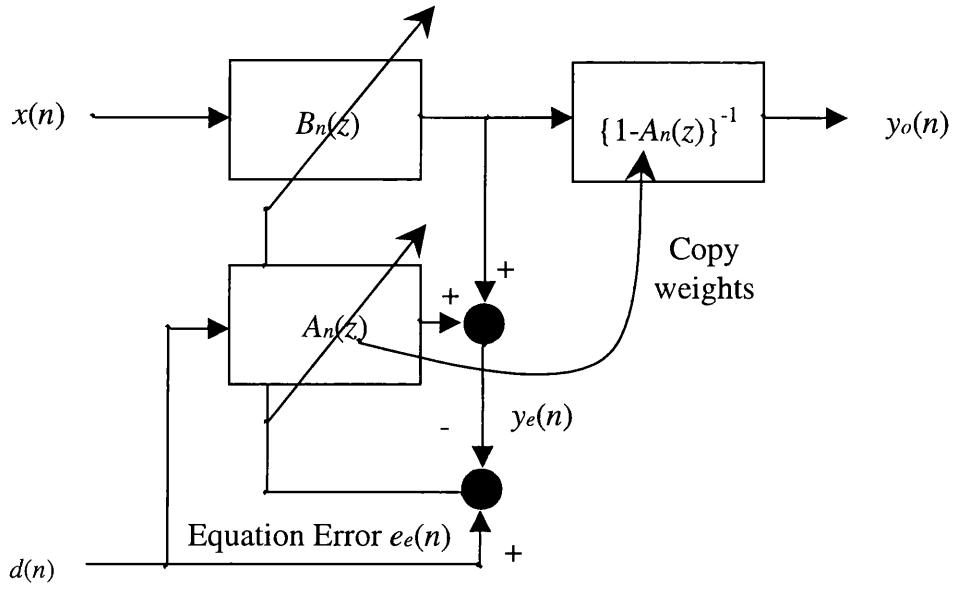

Figure 2.10: An equation error adaptive IIR filter. The filter weights are updated in all zero form and then copied into an all pole filter.

2.4 Recursive Least Square Algorithm for Adaptive IIR Filtering

Recursive least square algorithm for adaptive IIR filters can be formulated using an equation error method. An equation error adaptive IIR filter is shown in Figure 2.10. Unlike output error formulation [45], it essentially operates on a two-input, single-output adaptive FIR filter such that the polynomials, $A(z)$, associated with the poles are adapted in an all-zero form. After each update of the weights, the inverse of $\{1 - A(z)\}$ is copied to an all-pole filter which is in cascade with $\{B(z)\}$. The impulse response from $\{x(n)\}$ to $\{y(n)\}$ is infinite, and the cascaded filters have the same form as that of output error formulation. However, as mentioned earlier, the estimated coefficients obtained with this approach are generally different from those generated by the output error formulation due to biased estimates [63]. In a system identification application, this corresponds to incorrect estimates of $A_*(z)$ such that $\mathrm{E}[\theta(n)] = \theta_* + bias$, where $\theta_*(n)$ denotes the coefficients to be identified [63]. However, it can be shown that this bias will be zero if additive noise signal is zero [63]. Therefore,

RLS algorithm based on equation error formulation can give unbiased estimates if the additive noise is negligible.

By using the equation error formulation, RLS equation for an adaptive IIR filter can be given as (see [24] for details)

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \boldsymbol{G}(n)e(n) \tag{2.38}$$

where

$$\boldsymbol{P}^{-1}(n+1) = \frac{1}{\lambda}\left\{\boldsymbol{P}^{-1}(n) - \boldsymbol{G}(n)\boldsymbol{\phi}_e^T(n)\boldsymbol{P}^{-1}(n-1)\right\} \tag{2.39}$$

$$\boldsymbol{G}(n+1) = \frac{\boldsymbol{P}^{-1}(n-1)\boldsymbol{\phi}_e(n)}{\alpha(n)} \tag{2.40}$$

$$\boldsymbol{\alpha}(n) = \lambda + \boldsymbol{x}^T(n)\boldsymbol{P}(n-1)\boldsymbol{x}(n) \tag{2.41}$$

where $\boldsymbol{\phi}_e(n)$ is called the regression vector:

$$\boldsymbol{\phi}_e(n) = [x(n), \cdots, x(n-M+1), d(n-1), \cdots, d(n-N+1)]^T \tag{2.42}$$

where $x(n)$ and $d(n)$ are the input signals. The subscript e is used to distinguish equation error from the output error. The Hessian matrix, $\boldsymbol{P}^{-1}(n)$ is updated in the same manner as described in equation (2.36).

Over the last decade, fast versions of the RLS (e.g. Fast Kalman) algorithms have been developed [21, 36]. However, practical use of the fast RLS algorithm in real-time applications has been prevented in the past because of divergence due to numerical error accumulation in the prediction parameters [53]. Although this numerical error problem has been resolved, the resulting algorithm needs much more computation than standard RLS [21].

In summary, RLS algorithms are encouraged in adaptive IIR filtering due to the multimodal error performance surface of an output error gradient based algorithms and to achieve fast convergence. However, these algorithms are not often employed in practical applications due to its high computational cost, biased estimates, blow-up and numerical errors. In the latter part of this thesis, normalised RPE (i.e. $\boldsymbol{P}$=$\boldsymbol{I}$ in the equation (2.3.5)) is employed to achieve fine-tuning and track time varying changes.

Chapter 3

Evolutionary Algorithm Based Global Optimisation and Design Methods

In the previous chapter we have shown various classical algorithms in the design of adaptive IIR filters. These algorithms are mainly based on recursive least mean square technique, which has several drawbacks when the algorithms are implemented for practical applications. A major drawback of these algorithms is that they may fail when the error function to be optimised is multimodal. Furthermore, classical algorithms don't constrain the search space within a known range. For example, by constraining the search space of feedback coefficients within a stable region, stable filters can be easily produced. The drawbacks of classical algorithms focus the research activities into optimisation techniques and hence various evolutionary algorithms have been grown [37, 51, 44, 3, 73, 52, 46, 78]. Evolutionary algorithms are computer programs that evolve over time in a manner similar to the evolution of living matter. They involve a population of software modules whose objective is to solve some problem, whose solution is initially unknown. Using techniques based on survival of the fittest, reproduction, and mutation the modules compete and breed to yield new software modules.

A variety of evolutionary algorithms have been further developed and received increasing attention recently. The common ones are genetic algorithms, evolutionary programming and evolutionary strategies (ES). They all share a common conceptual base of simulating the evolution of individual structures. Each individual in the population receives a measure of fitness in the environment. Reproduction focuses attention on high fitness individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing a general heuristic for exploration.

In this chapter, various evolutionary operators, which have been proposed or modified most recently to improve the algorithm performance, will be further studied. Flowcharts for each algorithm are presented, and the differences between these evolutionary techniques are detailed. The aim of this chapter is to provide basic understandings of various evolutionary techniques, which are needed to investigate this research successfully.

3.1 Genetic Algorithms

Genetic algorithms are a class of computational modules that attempt to mimic the mechanisms of natural evolution to solve problems in a wide variety of domains. The theory behind GAs was proposed by John Holland in his landmark book [23]. Genetic algorithms operate on a population of individuals represented by chromosomes, which are in essence a set of character strings that are analogous to the DNA of living organisms. These individuals in a population yield different solutions to an objective function created for a particular problem. According to evolutionary theories, only the most suited elements in a population are likely to survive and generate offspring, thus transmitting their biological heredity to new generations [23, 16].

After an initial population has been chosen, a GA operates through a simple cycle of three stages:

1. *Reproduction*

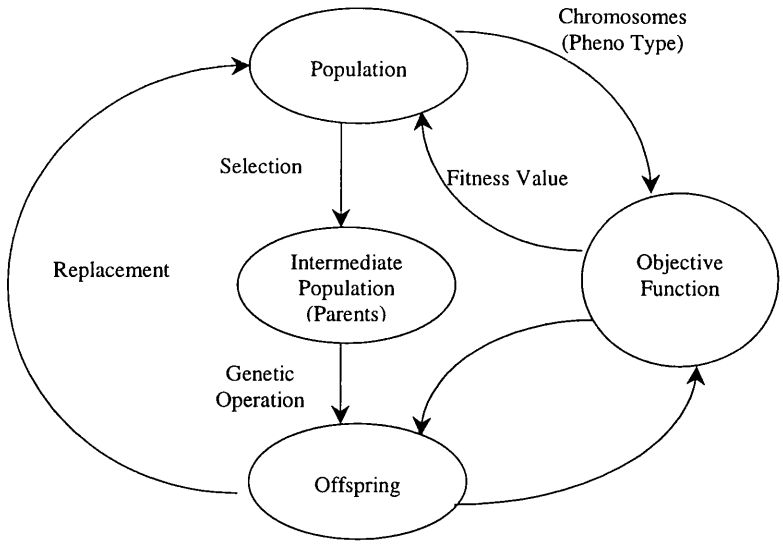

Figure 3.1: An example illustrating genetic cycle.

2. *Genetic Manipulation*

3. *Replacement*

A GA operation is shown in Figure 3.1.

3.1.1 Creation of initial population

An initial random population of strings is created. For example, in binary coded GAs the elements of binary strings, 0's and 1's, can be generated by flipping a coin or by using some other random function. Each element of a population is called a chromosome, while each element of a chromosome is called a gene.

3.1.2 Reproduction

In the first stage, the performance of each individual in the population is evaluated with respect to the constraints imposed by the problem. An objective function is used to evaluate the status (performance) of each chromosome. This is an important link between the GAs and the systems, which are to be optimised. The objective function takes a chromosome as input and produces a number or objective value as a measure of the chromosome's performance. The

range of values of the objective function can be varied for each problem. To maintain uniformity over various problem domains, a fitness function is needed to map the objective value to a fitness value [23]. Before doing this, each element of the population is decoded into the original values with a range [p_{min}, p_{max}] specified for a particular problem. Based on each individual's fitness, a selection mechanism creates an *intermediate population* for genetic manipulation. The selection process assures the survival of the best-fitted individuals. Each member of the intermediate population is called the *parent* and hence the intermediate population often takes a name called *parent population.* The combined evaluation/ selection process is termed reproduction.

There are at least four selection schemes currently in use: proportional, remainder stochastic sampling with replacement, rank-order and tournament.

Proportional selection: selects the mates according to the probability of their relative fitness values. It is a purely random approach in which the probability of an individual with a higher fitness value is greater than a lower one. It may be implemented with a roulette wheel selection referred in [23, 16].

Figure 3.2 illustrates an example of the roulette wheel selection process, where each individual is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the roulette wheel, individuals are chosen using stochastic sampling with replacement to fill the intermediate population. For an example, 16 individuals are mapped onto the roulette wheel shown in Figure 3.2.

Remainder stochastic sampling: this is a selection process which more closely matches the expected fitness value. The probability of contribution for each string is calculated as in the proportional selection scheme. Then, the expected number of individuals for each string is calculated as the product of the probability value for that string and the size of the population, being rounded off to the nearest integer. For example, a string with $\frac{f_i}{f} = 1.36$ receives 1 copy, and receives a 0.36 chance of placing a second copy. If the total number of individuals thus created is less than the population size, the fractional parts

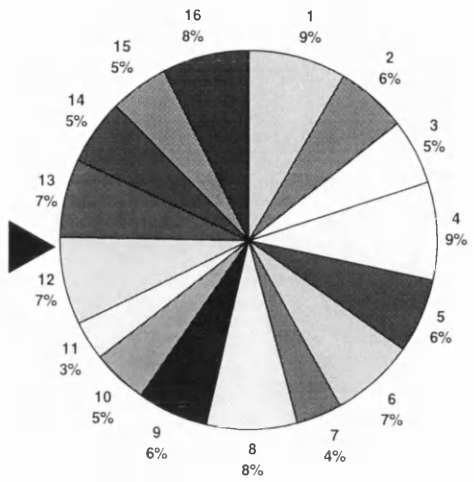

Figure 3.2: Proportionate selection using roulette wheel method.

of the expected number values may then be used in a roulette wheel selection procedure to fill the remaining slots in the population.

Remainder stochastic sampling is most efficiently implemented using a method known as stochastic universal sampling as illustrated in Figure 3.3. For example, a population of 16 individuals is arranged in random order, where each individual is assigned space on the pie-graph in proportion to fitness. An inner roulette wheel is also placed inside the pie with 16 equally spaced pointers. A single spin of the roulette wheel will now simultaneously pick 16 members of the intermediate population.

Ranking scheme: In this scheme, the M best out of a population size of N members are selected to form the members of the next generation. The remaining population can be filled in any manner. In the ranking schemes, the fitness value is used to rank all the strings. This is an important property which leads to a high selective pressure and hence fast convergence of an EA [80, 46, 39].

Tournament: chooses better individuals by holding a tournament among s competitors, with s being the tournament size. The winner of the tournament is the individual with the highest fitness of the s tournament competitors, and the winner is then inserted into the mating pool. The mating pool, being comprised

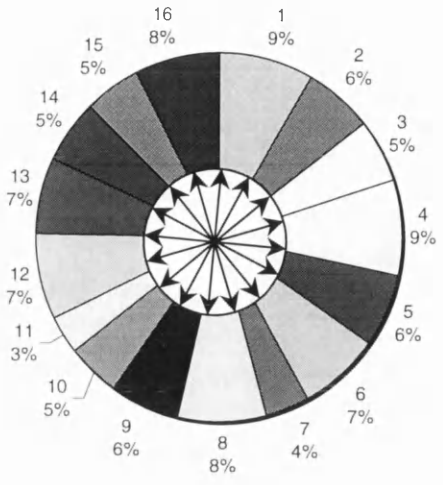

Figure 3.3: Remainder stochastic sampling with replacement. The individuals $\{1, 4, 8, 13\}$ get 2 copies, the individuals $\{2, 5, 6, 9, 10, 12, 15, 16\}$ get a single copy whilst $\{3, 7, 11, 14\}$ are die off.

of tournament winners, has a higher average fitness than the average population fitness [40]. This fitness difference provides the selection pressure, which drives the GA to improve the fitness of each succeeding generation [40].

3.1.3 Genetic Manipulation

The manipulation process employs genetic operators to produce a new population of individuals, which are termed offspring, by manipulating the genetic information (referred to as genes); possessed by the parents. Genetic manipulation comprises two operations, namely crossover and mutation.

Crossover: a recombination operator that combines subparts of two chromosomes (parents), which are to produce offspring that contain some parts of both original's genetic material. A probability term, *probability of crossover* α_c, is set to determine the operation rate. A number of variants on crossover operations are proposed. They include single-point, two-point, uniform, and adaptive uniform crossover etc.,. [23, 16, 4, 78, 40]. The simplest form is single-point crossover, which is illustrated in Figure 3.4. The parents and crossover point are randomly chosen and the portions of the two chromosomes beyond the crossover point are swapped to form the offspring. Multipoint crossover is

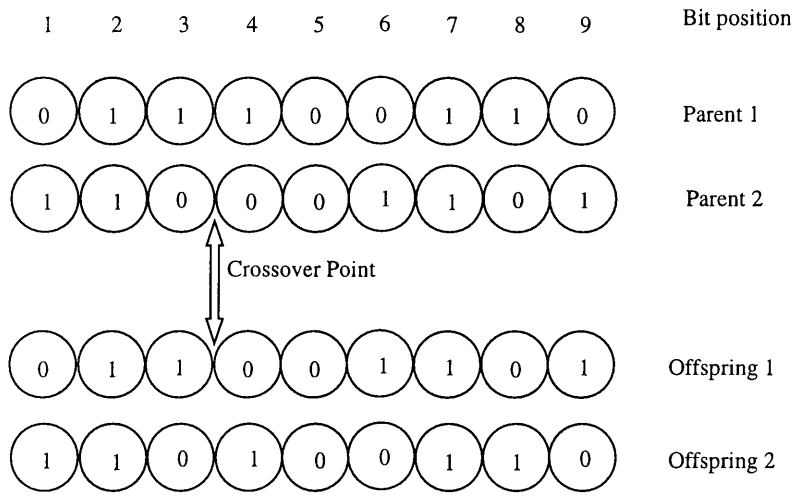

Figure 3.4: Simple single-point crossover.

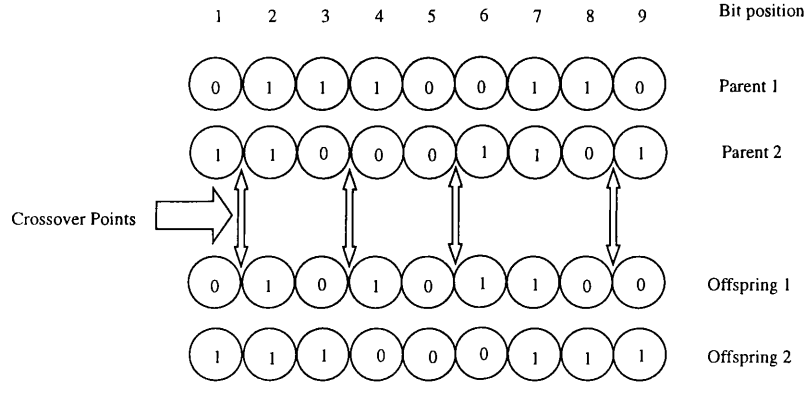

Figure 3.5: Multi-point crossover.

similar to single-point crossover, except that m crossover portions are chosen at random with no duplication. An example of this operation is illustrated in Figure 3.5, where 4 such crossover points are shown with binary strings of chromosomes.

In practice one and two-point crossovers have been widely used in standard GAs [57, 23]. Consequently, uniform crossover has been proposed as an alternative to one and two-point crossover so that more schema can be combined [4, 2, 9]. This approach combines two parents to produce two children like a normal crossover operator. It differs in that a binary template is used

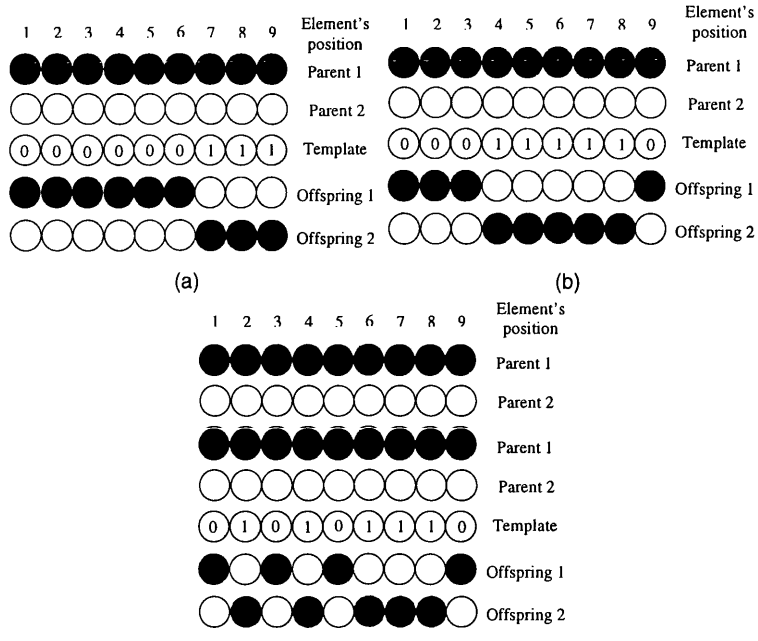

Figure 3.6: Uniform crossovers as (a) one-point (b) two-point and (c) multiple-point crossovers.

to choose which bit position from which parent contributes to which child. This process can be seen in Figure 3.6, the two parent chromosomes are shown above the crossover template, and the two children below. The template decides which parent contributes to which child at that element position. The uniform crossover can be thought of as an n-point crossover operator, where the value of n is dependent upon the exact makeup of the crossover template. For example, consider the binary templates shown in Figures 3.6(a) and 3.6(b), which contribute simple one and two-point crossovers respectively. Therefore, the uniform crossover operator, or a variant of it, can be viewed as a generic n-point crossover operator illustrated in Figure 3.6(c), specific instances of it being equivalent to a one or two-point crossover.

Mutation: is an operator that introduces variations into the chromosomes. The motivation of using mutation is to prevent the algorithm from reaching premature convergence [23]. After several generations it is possible that selection will drive all the bits in some position to a single value, either 0 or 1 in the binary case. If this happens without the GA converging to a satisfactory solution, then

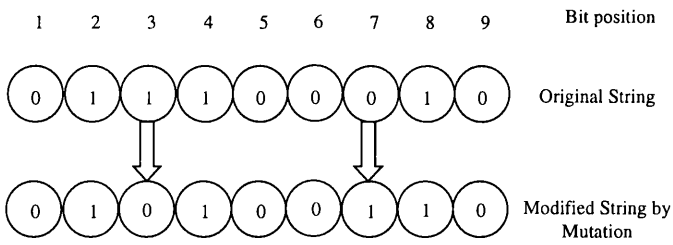

Figure 3.7: An example illustrating mutation operation. The mutation is applied to third and seventh elements of the string.

the algorithm has prematurely converged [34]. This may be a particular problem if a small population size is employed. Without a mutation operator, there is no possibility for reintroducing the missing bit value. Mutation therefore acts as a background operator, occasionally changing bit value, and allowing alternative alleles to be tested.

Mutation introduces random variations into chromosomes so that the population has enough genetic diversity. The operator occurs occasionally, with a probability called *probability of mutation* α_m. It randomly alters the value of a string position. A randomly generated bit replaces each bit of a bit-string if the probability test has passed. An example of mutation of the third and the seventh bits is shown in Figure 3.7, where the bit string [011100010] is changed to [010100110].

3.1.4 Replacement

After generating the offspring, two representative strategies are used to replace the old population [74]:

Generational-Replacement

Each population of size n generates an equal number of new chromosomes to replace the entire old population. This strategy may make the best member of the population fail to reproduce offspring in the next generation. So the method is usually combined with an elitist strategy where one or a few of the

best chromosomes are copied into the succeeding generation. The elitist strategy may increase the speed of domination of a population by a super chromosome.

SubsubsectionSteady-State Reproduction This strategy means that only a few chromosomes are replaced once in the population to produce the succeeding generation. Usually the worst chromosomes are replaced when new chromosomes are inserted into the population. The number of new chromosomes is to be determined by this strategy. In practice, only one to two new chromosomes are being used by steady-state reproduction.

3.1.5 Implicit Parallelism

A distinguishing feature of genetic algorithms over other evolutionary techniques is that they are able to process many building blocks or schemata in parallel. This property is called *implicit parallelism.* In natural populations, thousands or even millions of individuals exist in parallel. A parallel GA is generally formed by parallel components each is responsible for manipulating sub-population. There are two different ways of exploiting parallelism in GAs:

Centralised selection Models: use centralised selection mechanisms, a single selection operator works on the global population. Thus the parallel GA has a synchronous selection stage.

Island Models: employ distributed mechanisms; each parallel component has its own copy of the selection operator. In addition, each component communicates its best strings to a sub-set of other components. A migration operator and migration frequency defining the communication interval achieves this. These parallel GAs have an asynchronous selection stage.

3.1.6 Genetic Algorithm Cycle

A GA cycle is repeated until a desired termination criterion is reached [51]. For example, the fitness function or a predefined number of generations can be used as a termination criterion, but it is usually unknown. The best chromosome

in the final population can become a highly evolved solution to a problem. A Flowchart illustrating the complete genetic cycle is shown in Figure 3.8.

3.2 Evolutionary Programming

Evolutionary programming (EP), is another stochastic optimisation strategy similar to GAs, but it places emphasis on the behavioural linkage between parents and their offspring, rather than seeking to emulate specific genetic operators as observed in nature. Like GAs, EP is a useful method of optimisation when other techniques such as gradient decent or direct analytical discovery are not possible [3, 13].

Evolutionary programming was first introduced by Lawrance Fogel in his landmark book "Artificial Intelligence Through Simulated Evolution", in 1966. In the book, finite state automata were evolved to predict symbol strings generated from Markov processes and nonstationary time series. Recently there has been renewed interest in the method prompted by the work of David Fogel and others [3, 46, 78, 13].

3.2.1 The Evolutionary Programming Process

Unlike GAs, EP uses phenotypic representation of the parameters and relies on mutation as the primary search operator. The basic method of EP can be summarised as follows:

1. Initial population of trial solutions are chosen randomly. The number of solutions in a population is highly relevant to the speed of optimisation, but no definite answers are available as to how many solutions are wasteful.

2. Computing fitness assesses each offspring solution. Each string s_i is assigned a fitness value $\phi(s_i)$ which may be a complex function of the true fitness of s_i or the raw fitness value of s_i itself.

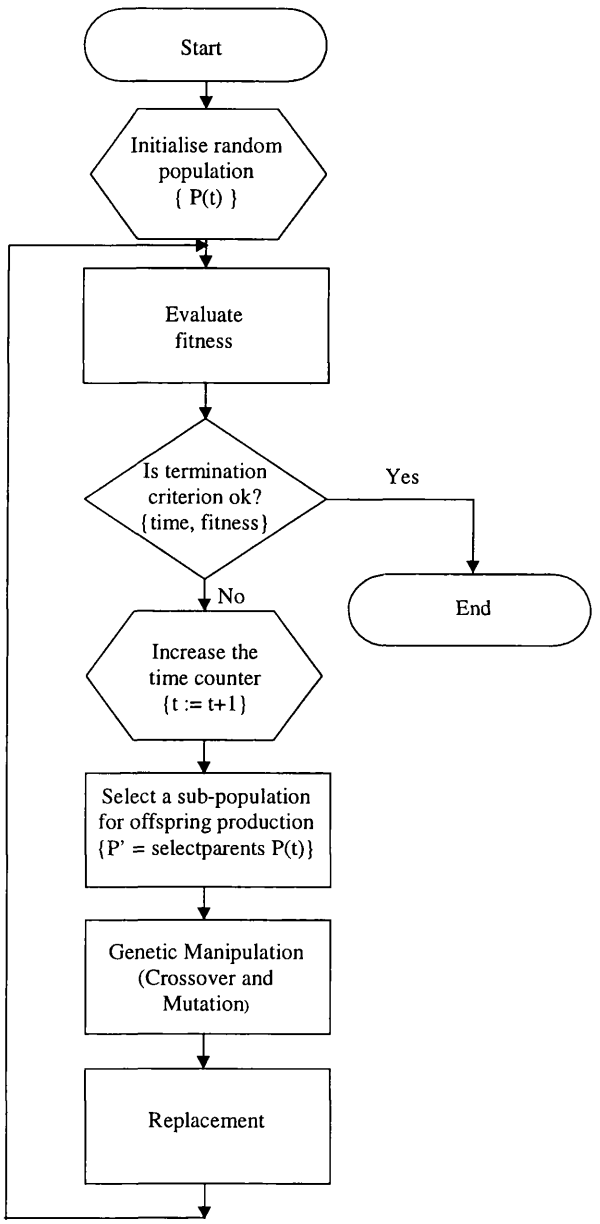

Figure 3.8: A flow-chart illustrating the GA.

3. Selection of better individuals based on fitness measurement. Typically, tournament is held to determine N solutions to be retained for the population, although this is occasionally performed deterministically. There is no requirement that the population size is held constant however, or that only a single offspring is generated from each parent.

4. Each solution is replicated into a new population. Each of these offspring solutions are mutated according to a distribution of mutation types. The severity of mutation is judged on the basis of the functional change imposed by the parents. Using each $s_i, i = 1, \cdots, N$ a new string $s_i(t+1)$ is generated as follows: $s_i(t+1) = s_i + N_{0,\phi(s_i(t))}$; where $N_{0,\phi(s_i)}$ represents a Gaussian random variable with mean 0 and variance $\phi(s_i(t))$. At each generation, the variance $\phi(s_i(t))$ corresponding to a chromosome i is obtained from its fitness value so as to reduce the severity of mutation as the algorithm approaches global solution.

A flowchart representing the EP is shown in Figure 3.9.

3.2.2 Important Features of Evolutionary Programming

There are three main ways in which evolutionary programming differs from genetic algorithm:

1. An important feature of EP is the lack of any kind of crossover or recombination operator. It has been reported in [13] that macromutations like the crossover and inversion operator used in GAs are not required for successful adaptation.

2. In EP, there is no constraint on the representation, but typical GA approach involves encoding the problem solutions as a string of representative tokens, the genome. In EP, the representation follows from the problem. For example, a neural network can be represented in the same

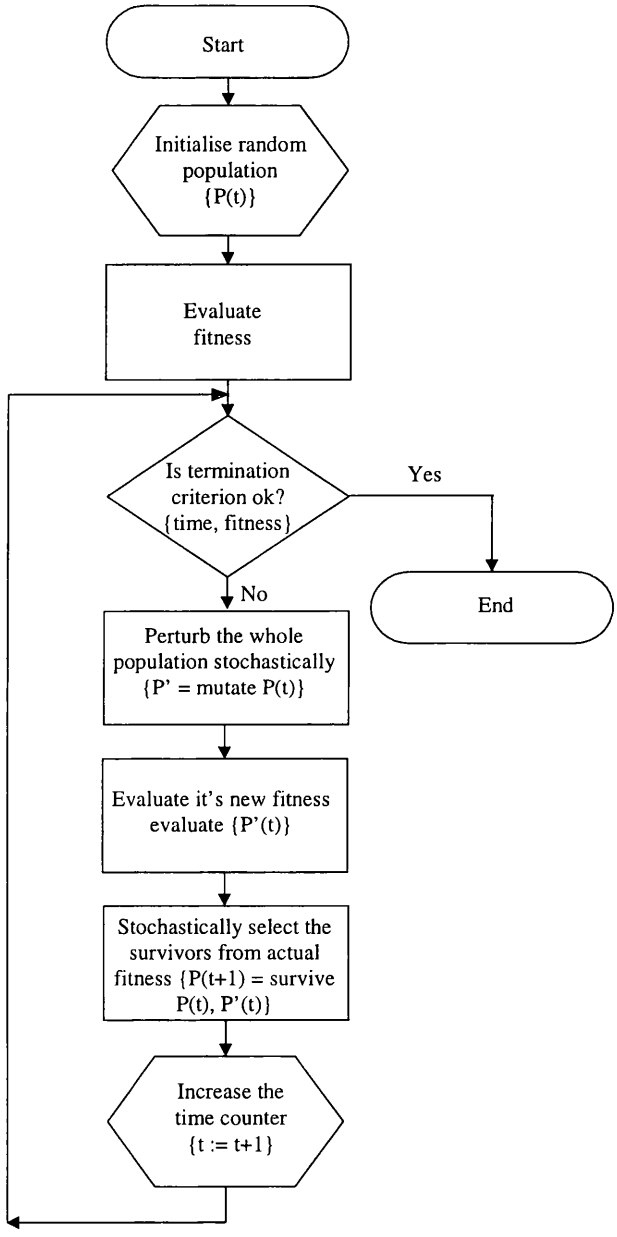

Figure 3.9: A flow-chart illustrating the EP.

manner as it is implemented since the mutation operation does not demand a linear encoding.

3. The mutation operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in the behaviour of the offspring as highly probable and substantial variations as increasingly unlikely. Furthermore, the severity of mutation is often reduced as the global optimum is reached, this is the motivation of using the fitness to calculate the standard deviation of the mutation operator. It can be seen that the standard deviation of the mutation process is calculated from the fitness value of the parent's chromosomes. A complete detail of calculating this variance can be found in [12].

3.3 Evolutionary Strategies

Evolutionary Strategies (ES) are another class of evolutionary techniques, which are based on the same principle as EP and GAs. The algorithm operates on a population of string structures, each of which represents a potential solution to the optimisation problem. An important difference between evolutionary strategies and genetic algorithms is that ES operate at the phenotypic level using the parameter values themselves as genetic material while the latter work on a genotype level (coding of the real parameters).

3.3.1 Important Features

Unlike other evolutionary algorithms, evolutionary strategies use a unique method of mutation. For example, in early ES', each parameter value has a standard deviation associated with it, which decides the variance of the mutation. The parameter values and standard deviation are concatenated to form the string structure of a population. In the early ES', these standard deviations are fixed for each parameter and for the entire run of the algorithm.

Subsequently, the original version of mutation has been modified by Schwefel [59] in 1981, where the standard deviation of the mutation process are a part of the genetic material and undergo genetic modifications during the adaptation. The mutation process is carried out as follows:

$$\sigma_x(t+1) = \sigma_x(t) \times N_{0,d_\sigma} \tag{3.1}$$

$$x(t+1) = x(t) + N_{0,\sigma_{x(t+1)}} \tag{3.2}$$

where $x(t)$ and $\sigma_x(t)$ are parameter and the associated standard deviation at iteration t, while $x(t+1)$ and $\sigma_x(t+1)$ are the new values after mutation. Further, N_{0,d_σ} is a Gaussian process with mean 0 and standard deviation d_σ. Thus in Schwel's ES mutation works both on the parameter value, x, and on the standard deviation, σ_x.

3.3.2 Differences between ES and EP

There are two key differences between evolutionary strategies and evolutionary programming:

1. Evolutionary programming typically uses stochastic selection via a tournament. Each trial solution in the population faces competition against a pre-selected number of opponents (tournament size s) and receives a win if it is at least as good as its opponent in each encounter. In contrast, ES typically uses deterministic selection (e.g. Remainder stochastic sampling with replacement) in which the worst solutions are purged from the population based directly on their function evaluation.

2. Evolutionary strategies use recombination process, while EP on the other hand often does not employ such a process during evolution.

3.4 Summary and Consideration in Filter Design

This chapter has provided an overview of various evolutionary algorithms. There are currently 3 main paradigms in evolutionary algorithm research: GAs, EP and ES, together with classifier systems. Fundamental differences between these algorithms and classical methods are:

1. EAs search from a population of solution vectors, not a single solution vector and hence global, hard constraints and multiobjectives problems can be easily solved.

2. EAs exclusively use values of the function under study, and do not consider gradient information, and hence less prone to noise and local optima.

3. EAs use probabilistic, not deterministic transition rules and hence more chance of getting a global solution.

In this chapter, the salient features of each evolutionary technique have been revealed more clearly. This study has given a clear foundation on various evolutionary techniques and has provided a theoretical concept to expand this research to improve the current techniques in various DSP applications. Evolutionary algorithm optimise a population of chromosomes unlike conventional optimisation techniques such as simulated annealing (SA) that optimise only a single solution vector. Considering several solution vectors of high performance reduces the probability of selecting an improper solution. Evolutionary algorithms remain highly general because their optimisation is directly based on the function value.

A comparative study of various evolutionary techniques shows that evolutionary strategy (ES) gives the best convergence performance [46, 78]. However, genetic algorithms are demonstrated as being the most powerful method for stability constraint, multimodal and multiobjective problems which often arise when designing IIR filters [74, 37, 2]. This contradictory result arises due to the

randomness of the EAs operation and hence causes difficulties of predicting the performance in various applications or the differences of their operation. An important difference between GAs and ESs is the fact that GAs operate on a genotype level, while ESs operate at the phenotypic level using the parameter values themselves as genetic material [78]. In ESs crossover occurs only between the boundaries of the parameters and mutation acts on individual parameters as a whole and not at the level of binary alphabets used to represent the coefficient [46, 78]. Furthermore, ESs use unique variance of mutation for each parameter and are mostly evolved along with the solutions [78]. Therefore it is evident that ESs use comparatively short representation of chromosomes and can provide better precision when optimising large problem domains. However, the ESs represent long chromosomes and hence require high computation when compared to one that has only the parameters itself without using evolving variances.

In summary, among various evolutionary techniques revised, genetic evolution of IIR digital filters is appeared as being quite robust, but uses binary representation of coefficients with the added expense of coding techniques. Moreover, the binary coding would require prohibitively long representation and hence provide less precision when designing higher order filters [25]. A comparative study on binary and floating-point GAs shows that the latter is faster, more consistent from run to run, and provides higher precision, when designing a nonlinear dynamic control model [25]. Therefore, it is evident that the EAs, which use floating-point representations along with crossover, and mutation operations that behave exactly the same way as binary GAs would be useful when designing IIR filters.

Chapter 4

Evolving Stable Poles and Globally Optimal IIR Filters

Stability of adaptive IIR systems is determined by observing the pole locations. An unstable system has one or more of its poles, which lie outside the unit circle. To determine instability, pole polynomials must be factored. Factoring the polynomials of higher order systems is computationally expensive [63]. To resolve this problem, poles can be designed directly from a population so that the adaptive filters can be easily ensured within the stable region throughout the genetic search [61, 69]. Unfortunately, the poles are generally complex values. Therefore, applications of evolutionary algorithms that evolve complex valued chromosomes are of paramount importance. In addition, the equalising filters employed in QAM modems have complex parameters and will necessitate the use of complex valued chromosomes [56].

The work presented in this chapter shows the recent development of designing complex IIR filters using floating-point EAs. In particular, this work employs a method for representing complex filters into chromosomes that avoids coding/ or decoding techniques. Unlike standard methods, this method evolves the poles instead of feedback coefficients, which in turn simplifies the initial selection of poles to lie inside the stable region. A new method of crossover

operation is introduced which performs crossover between floating-point chromosomes as the same manner as in the binary GAs. These new techniques are applied to system modelling problems to determine the ability of the floating-point EA's convergence on simulated data.

The rest of this chapter is organised as follows: Section 4.1 provides the motivation of using complex filters in digital signal processing. In Section 4.2 we present the design techniques of modelling complex IIR filters using pole design method. The advantages and drawbacks of the approach when designing the poles are clearly outlined. Two major objective functions, which are based on MSE and mean modulus error (MME) are discussed. It has been shown that, the length of the training data affects the convergence of the EA. Section 4.3 shows the design techniques of recently developed genetic operators to evolve floating-point chromosomes. The reason for choosing the tournament selection is discussed. We have also shown how EA can use various cost functions with the use of tournament selection. Design methods of developing mutation operator are given. Simulations illustrate the advantages of using Gaussian function as a mutation operator. The special features of uniform crossover are discussed. A major drawback of floating-point chromosome is outlined, and a new method of crossover is given to overcome this problem. Finally, in Section 4.4, we present a selection of simulation results. These were obtained by modelling various IIR systems with the EA techniques developed during course of this research. The chapter also gives a brief summary of the work with suitable conclusions.

4.1 Complex Filtering

There are two major reasons that encourage the use of complex filtering in digital signal processing. The first reason of using the concept of complex filtering is to represent QAM signals as complex data for mathematical convenients [6]. For example, quadrature and in-phase carriers can be stored in DSP processors as complex data and hence associated processing can be easily done in complex

forms [76]. Second reason of using complex filtering is to reduce the signal bandwidth prior to sampling. To realise this reduction of signal bandwidth requires an understanding of the concept of complex waveforms discussed below:

Passband Filtering: There are many applications, which involve signals concentrated in a narrow band, i.e. with bandwidth much less than its centre frequency [55]. It can be easily shown that a continuos-time signal with highest frequency B can be uniquely represented by samples taken at minimum rate (Nyquist rate) of $2B$ samples per second [55]. However, if the signal is a bandpass signal with frequency components in the band $B_1 \leq F \leq B_2$, a blind application of the sampling theorem would have us sampling the signal at a rate of $2B_2$ sampling per second. Signals with higher sampling rate would require high speed digital circuitry which in turn limits the use of general purpose DSP processors in real applications [76, 6]. To overcome this problem the signal can be downconverted into some lowest frequency so that less sampling rates can be applied. A theoretically sound technique is bandpass sampling, i.e. undersampling at a rate on the order of the information bandwidth, but chosen such that aliased images do not overlap and that the image nearest DC is properly positioned in the baseband [76, 6]. However, in doing so our signal becomes complex, since its symmetry has been destroyed [19]. More details of complex signals can be found in textbook [19, 55, 15].

4.2 Evolutionary Approach to Globally Optimal Filtering

Consider an IIR digital filter, comprised of M feedforward coefficients $\{b_k(n)\}_{k=0}^{M-1}$ and L poles $\{p_k(n)\}_{k=1}^{L}$. We assume the general case of filter having complex coefficients, and we also assume the filter input $x(n)$ is complex. The aim of the evolving process is to choose the filter parameters $\{b_i\}_{i=0}^{M-1}$ and $\{p_j\}_{j=1}^{L}$ to

satisfy some desired criteria. A standard criterion is to minimise the MSE,

$$J(\Omega) = \frac{1}{w}\sum_{n=0}^{w-1} |d(n) - \hat{y}(n;\Omega)|^2 \tag{4.1}$$

where $\hat{y}$ denotes estimate of filter output Ω is vector of filter parameters given by

$$\Omega = \{b_i(n),\ i = 0, 1, 2, \cdots, M-1; p_j(n),\ j = 1, 2, \cdots, L\} \tag{4.2}$$

where $d(n)$ is the desired output sequence, and w represents the number of training samples. The training samples or training signals are a set of examples that contains elements which consist of paired values of input sequence, $\{d(n), \hat{y}(n)\}_{n=0}^{w-1}$, in order to learn the algorithm during adaptation. This kind of adaptation, which uses training signals, is called trained adaptation.

We mention however that the evolutionary search approach is able to deal with a wider class of cost functions with ease, e.g. least square error (LSE) and mean modulus error (MME). In this work, LSE and MME criteria have been successfully employed. For a given training set $\{d(i), \hat{y}(i)\}_{i=1}^{w}$, the least square recipes is to minimise the summed squared error as illustrated in (4.1). The MME criteria on the other hand minimise the summed modulus error:

$$S = \frac{1}{w}\sum_{i=1}^{w} |d(i) - \hat{y}(i;\Omega)| \tag{4.3}$$

In the least square approach small errors have less emphasis than large errors. In contrast to modulus error, which gives equal weight to all errors.

A major advantage of evolutionary techniques over classical algorithms is that the search space of parameters can be constrained to a suitable range in order to achieve certain performance. In this Chapter, we design the poles rather than feedback coefficients so that the search space of the feedback sections can be restricted within the known stable region. The pole polynomials can be easily converted back into feedback coefficients. The relationship between poles and feedback coefficients can be obtained by comparing the z-transfer function:

$$H(z) = \frac{\sum_{i=0}^{M-1} b_i z^{-i}}{\prod_{j=1}^{L}(1 - p_j z^{-j})} = \frac{\sum_{i=0}^{M-1} b_i z^{-i}}{1 - \sum_{j=1}^{L} a_j z^{-j}} \tag{4.4}$$

where $\{p_j\}_{j=1}^{L}$ are the poles whilst $\{a_j\}_{j=1}^{L}$ represent the feedback coefficients of the filter. For example, the pole polynomials $\{p_k(n)\}_{k=1}^{2}$ of a second-order system can be related to the feedback coefficients $\{a_k\}_{k=1}^{2}$ in such a way that:

$$a_1(n) = p_1(n) + p_2(n) \tag{4.5}$$

$$a_2(n) = p_1(n)p_2(n) \tag{4.6}$$

where $\{a_k(n)\}_{k=1}^{2}$ are complex feedback coefficients at time n. Consequently, alternate realisations such as parallel or cascade forms can be considered to facilitate this conversion trivial. The sub-filters of these realisations can be represented as simple low order filters, but there is no restriction to reduce the filter length as first or second-orders. However, in this work we consider direct form realisation and use simple numerical methods for translating the poles into coefficients when designing large order IIR systems. The primary advantage of using the direct form structures is to develop evolving models in the same manner as it is implemented. The adaptive IIR filtering approach using alternative realisations provide high complexity and poor performance than the direct form filters [64, 31, 3, 48, 62]. Appendix B shows how the coefficients and the poles can be related to each other for a general order filter.

Evolutionary algorithms generally maximise a performance criterion rather than minimising a cost function. The maximising functions are called fitness or objective functions and can be formed using the cost functions described above. They measure the status of each chromosome. A simple fitness function formulated from MSE criterion is

$$f(mse) = \frac{1}{1 + J(\Omega)} = \frac{1}{1 + \frac{1}{w}\sum_{n=0}^{w-1} |d(n) - y(n;\Omega)|^2} \tag{4.7}$$

where w is number of training set or window size. It can be seen in the equation (4.7) that MSE is added to 1 when calculating the fitness function. The reason for introducing this value is to prevent the fitness function from reaching an infinite value when the MSE approaches to zero.

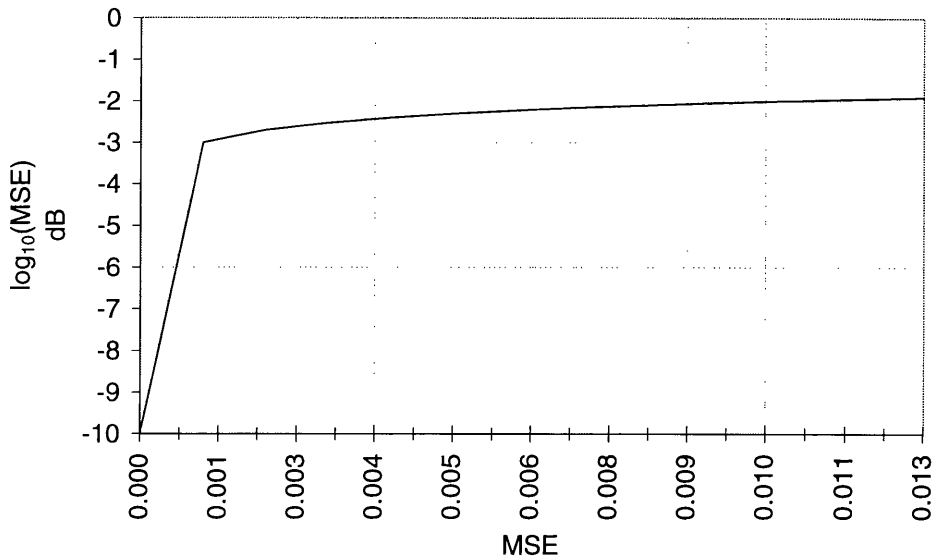

Figure 4.1: Logarithmic scaling of MSE.

Logarithmic scaling of the objective functions can be used to improve the adaptive algorithm's ability to locate good sets of filter coefficients when GAs are implemented with proportional selection schemes [78]. Logarithmic scaling provides nonlinear mapping between the objective function values and actual error performance. This scaling accommodates larger variations of objective functions values within a smaller range. For example, Figure 4.1 illustrates a logarithmic scaling of MSE plotted against actual values of MSE. Figures 4.2 and 4.3 show these error performances while modelling a direct system (4.29) of parameters (4.30) given in Section (4.4.1) with 100 training samples. The logarithmic scaling has been successfully employed to improve the selection pressure of the proportional selection scheme when binary GAs were implemented to model various IIR models [78].

Even though the pole design method offers several salient features, it has a major drawback that provides multiple optima with the same fitness values that arise from reordering the poles within the filter. For example, consider the MSE performance shown in Figure 4.4 which was obtained when modelling a second-order ARMA model by an adaptive filter with equal number of parameters. The model and the adaptive filter used in this example have the following transfer

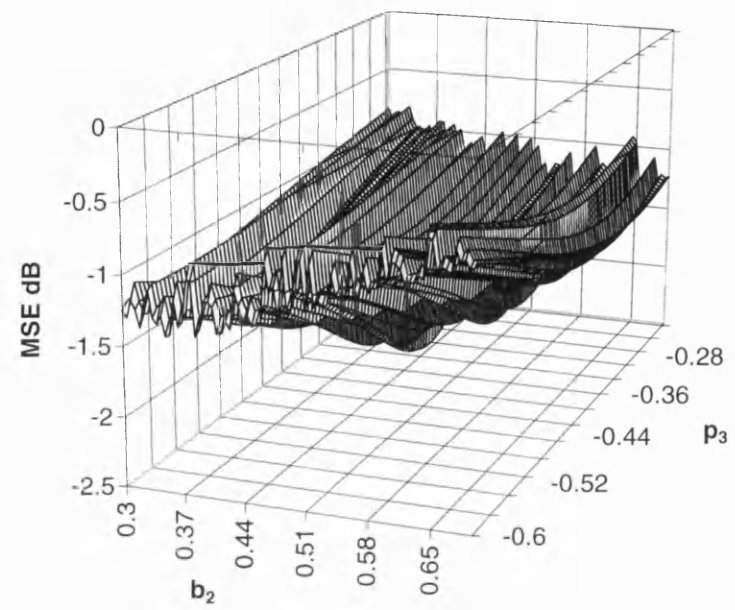

Figure 4.2: Logarithmic scaling of MSE while modelling the system (4.29) of parameters (4.30).

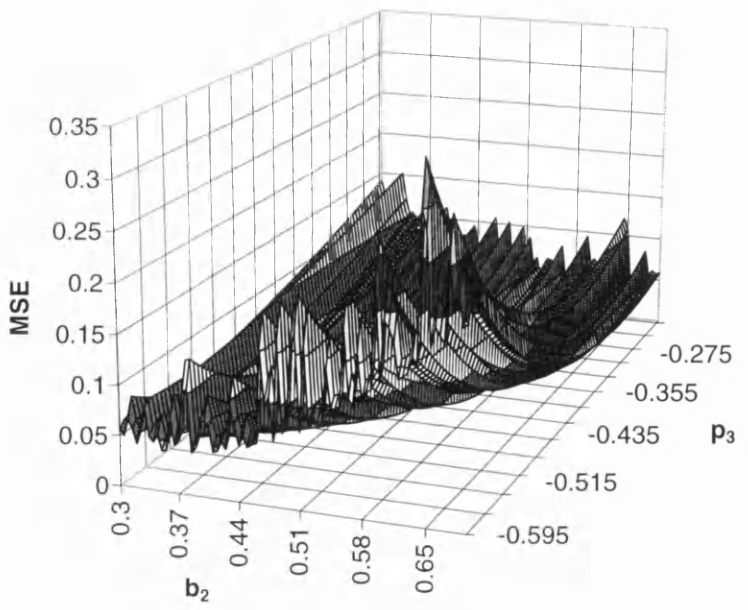

Figure 4.3: Non-logarithmic scaling of MSE.

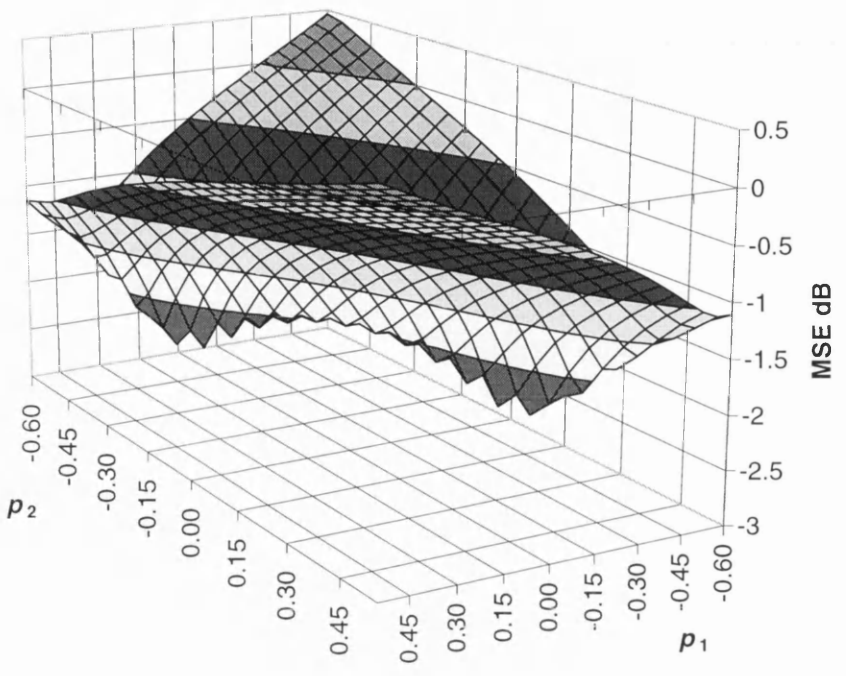

Figure 4.4: 3-D plot showing the MSE surface while modelling a second-order all pole system using an adaptive filter. Mean squared error is shown for various choices of poles of the adaptive filter. Two global minima exist by interchanging the poles within the filter.

functions:

$$G(z) = \frac{1}{(1 - 0.2z^{-1})(1 + 0.4z^{-1})} \tag{4.8}$$

$$A(z) = \frac{1}{(1 - p_1 z^{-1})(1 + p_2 z^{-1})} \tag{4.9}$$

where p_1 and p_2 are the poles of the adaptive system. The plant has two separate poles, which are arbitrary chosen as 0.2 and -0.4 respectively. The MSE is obtained for various choices of p_1 and p_2 by applying a training set of 100 samples. It is seen from the Figure 4.4 that the MSE has two global minima, which can be obtained by swapping the poles within the filter. i.e. interchanging the poles such as $\{p_1 = 0.2, p_2 = -0.4\}$ and $\{p_1 = -0.4, p_2 = 0.2\}$ provided the same objective function value as shown in Figure 4.4. This property leads to a major drawback when crossover operations are employed within the feedback sections. The simulation results confirming this statement are illustrated in the latter part of this chapter. In this chapter we use crossovers only for evolving the feedforward sections.

4.3 Genetic Operation in Floating-point Chromosomes

Genetic algorithms manipulate the most promising chromosomes searching for improved solution. The chromosomes are finite length string structures representing a possible solution of a problem domain. Binary GAs use a chromosome string whose elements (genes) are binary [23]. These GAs require chromosomes with $4(L+M)B$ elements each of which is a binary digit. This is a prohibitively long representation and hence provides lower precision when the algorithm is employed with large domains [25].

In this work a more sophisticated method is used to represent the floating-point numbers without employing any coding techniques. Genetic algorithms with floating point representation of chromosomes of length l and fixed population size N are considered. Each individual in the population corresponds to an element of the space

$$S = \{x\}^l \tag{4.10}$$

where x is a complex number. The population space is denoted as S^N and we call S^2 the parent's space. The population can be written in the vector form as follows

$$\vec{C} = \{C_i, i = 1, \cdots N\} \tag{4.11}$$

where

$$\vec{C} \in S^N \tag{4.12}$$

where C_i denotes the i^{th} individual of $\vec{C}$ and can be given in mathematical notation as:

$$C_i \in \vec{C} \tag{4.13}$$

Each element of C_i is a parameter of the filter. The j^{th} element of C_i is related to the filter parameters in the following way

$$c_j = p_j \qquad \text{if} \qquad 0 \leq j \leq L-1 \tag{4.14}$$

where p represents the poles, and

$$c_j = b_j \qquad \text{if} \qquad L \leq j \leq (L + M - 1) \tag{4.15}$$

where b represents the numerator coefficients.

4.3.1 Making Use of Tournament Selection

Evolutionary algorithms use a selection mechanism to select individuals from the population to insert into a mating pool. Individuals from the mating pool are used to generate new offspring, with the resulting offspring forming the basis of the next generation. As the individuals in the mating pool are the ones whose genes are inherited by the next generation, it is desirable that the mating pool be comprised of better individuals. A selection mechanism in EAs is simply a process that favours the selection of better individuals in the population for the mating pool. As mentioned in chapter 3 there are many selection schemes for EAs, including ranking, tournament, and proportionate schemes, each with different characteristics. An ideal selection scheme would be simple to code and efficient for both non-parallel and parallel architectures. Furthermore, a selection scheme should be able to adjust its selection pressure so as to tune its performance for different domains [40].

Tournament selection is increasingly being used as an EA selection scheme because it satisfies all of the above criteria. Recent research work [40] on tournament selection clearly indicates that the convergence rate of a EA is largely determined by a factor called selection pressure, with higher selection pressure resulting in higher convergence rates. The selection pressure is the degree to which the better individual is favoured: the higher the selection-pressure, the more likely that the better individual is favoured. Also it is reported that simply increasing the tournament size, s, can provide increased selection pressure, as the winner from a larger tournament will, on average, have a higher fitness than the winner of a smaller tournament [40].

Unlike other selection schemes, which were discussed in previous chapter,

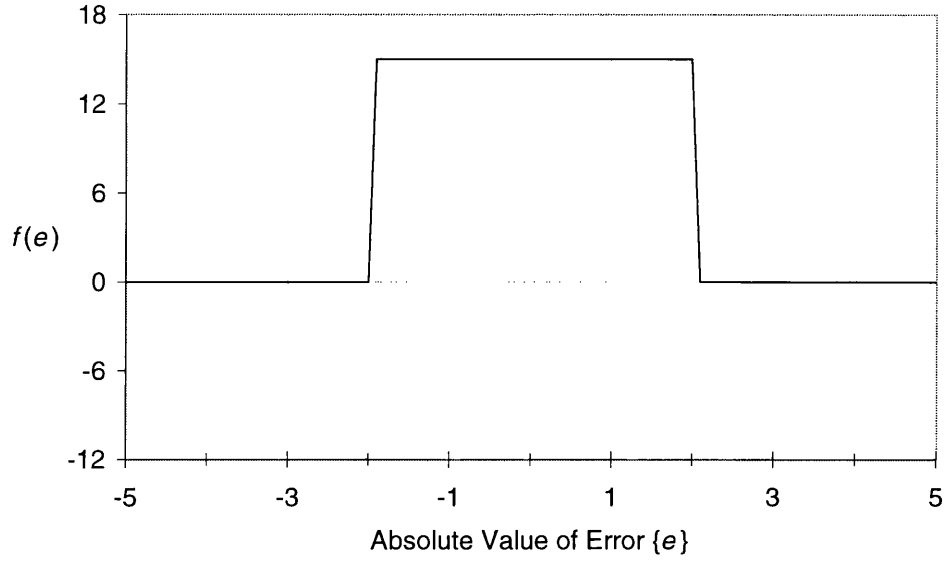

Figure 4.5: An example of a nonlinear threshold error criterion.

tournament selection can use any cost functions with ease [40]. They can either minimise or maximise a function regardless of its structure. Finding maximising functions from minimising criteria, therefore, can be avoided. It also facilitates the EAs to optimise any performance criteria that provide an acceptable band of errors in the objective function. For example, the nonlinear threshold, which is shown in Figure 4.5, can also be used in the instance, where a band of error may be acceptable. In this work, the MME criterion has been successfully employed in optimisation of adaptive IIR filters with the support of tournament selection. It is important to note here that the use of raised powers of errors, (e.g. e^2, e^4) provides good selecting capability and contributes faster convergence than using MME when using other selection schemes [47].

In this work we employ tournament selection to select the parent chromosomes. The smallest tournament-size $s = 2$ is chosen to keep the selective pressure to a lowest value with an aim to avoid premature convergence. Throughout the present work, we use the MSE cost functions unless otherwise specified. The main reason for choosing MSE criterion is that the theory of mean square optimisation is well developed, and can be used as a standard criterion to test the proposed genetic operations.

4.3.2 Split-point Uniform Crossover

Uniform crossover has been studied extensively and their results have shown a considerable improvement in convergence performance when compared to standard one or two-point crossover operators [71, 4]. Figure 4.17 compares these convergence properties when modelling an ARMA model by evolving filters using EAs. The details of this experiment are discussed in Section 4.4. In binary coded GA, the recombination process can introduce new members into the original chromosome structure since each parameter of the filter can be split into small elements called genes. Floating-point EA on the other hand represent the filter parameters as uniquely indivisible genes [17, 25], therefore the recombination cannot occur within the individual parameter, this simply results in a shuffling of the two parents.

Considering the above issues in mind, we propose a novel method of crossover for floating-point (complex) chromosomes. Even though the filter parameters are represented directly in the chromosomes, each of them can be split into small parts as in the binary GAs. This can be accomplished by introducing a third alphabet into the binary template, which incorporates split-point crossover between selected floating-point numbers of the parent chromosomes. This can be easily shown with the following example: Consider the Figure 4.6, which shows a template whose elements have three alphabets 0, 1, and 2. The elements corresponding to the alphabet 0 and 1 are swapped as the whole numbers as illustrated in Figure 3.6. The elements corresponding to the third character "2" are split into small portions and combined as follows:

$$|o_{i,1}| = p|c_{i,1}| + (1-p)|c_{i,2}| \tag{4.16}$$

$$\angle o_{i,1} = p\angle c_{i,1} + (1-p)\angle c_{i,2} \tag{4.17}$$

where $o_{i,1}$ represents the i^{th} element of offspring 1, $c_{i,1}$ and $c_{i,2}$ represent i^{th} elements of parent 1 and parent 2 respectively, and p is a uniform random number between $[0 \cdot \cdot 1]$. Similarly, i^{th} element offspring 2 can be generated as

$$|o_{i,2}| = p|c_{i,2}| + (1-p)|c_{i,1}| \tag{4.18}$$

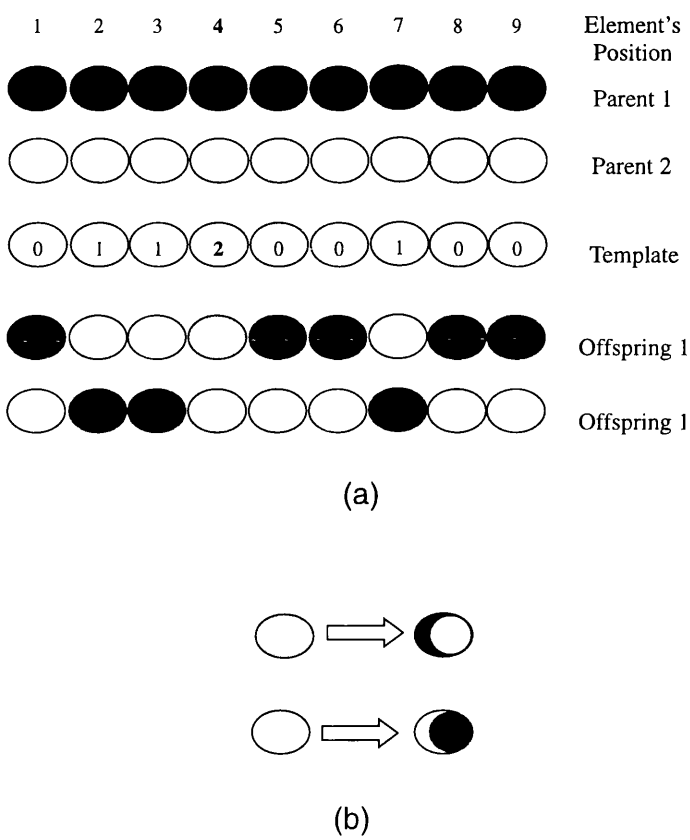

Figure 4.6: (a) Split-point crossover. The character "2" determines the position where split-point crossover to be taken place. (b) Split-point crossover occurs in the forth element of the parents

$$\angle o_{i,2} = p\angle c_{i,2} + (1-p)\angle c_{i,1} \tag{4.19}$$

This method of crossover, which is applied between parent chromosomes, has several additional features compared with the standard crossover operation shown in Figure 3.6. Split-point crossover reduces the schema growth by providing more destruction. It can be easily shown that the probability of survival of a particular schema H via the split-point crossover is

$$p_{s,split}(H) = \alpha_c \left\{\frac{1}{3}\right\}^{o(H)} \tag{4.20}$$

where $o(H)$ represents the order of the schema H. The order refers that number of fixed elements defining a schema [23]. Consequently, the probability of survival via standard uniform crossover is

$$p_{s,standard}(H) = \alpha_c \left\{\frac{1}{2}\right\}^{o(H)} \tag{4.21}$$

From the above two equations it is clear that the split-point crossover reduces the schema growth and hence produces more new members which in turn can avoid premature convergence. The best features of split-point crossovers are clearly seen from the simulation results shown in latter part of this chapter. However, this crossover scheme has not still fulfil the binary representation - the magnitude of a digit obtained through split-point crossover operation is always less than that of highest value of the parents before crossover. For example, let us assume that magnitudes of the third elements corresponding to two parents p_1 and p_2 be 0.8 and 0.6 respectively. Also assume that the portion p which is selected from a random number generator is 0.9. Recombining these elements can form two elements for offspring o_1 and o_2:

$$|o_{3,1}| = 0.9 \times 0.8 + (1-0.9) \times 0.6 = 0.78 \tag{4.22}$$

$$|o_{3,2}| = 0.9 \times 0.6 + (1-0.9) \times 0.8 = 0.62 \tag{4.23}$$

It is clear that recombination cannot produce new members with the values greater than those of original members.

As we mentioned in the previous section, adaptive filters provide multiple global optima that can be obtained when reordering the poles among the filter. This property leads to a major limitation of using the crossover operation between the feedback sections. This is because during the evolution process the poles of the adaptive filters can lie in any order within the filter structure, consequently applying crossover between feedback sections can degrade the convergence performance. Therefore, in this work we apply the crossover only to the feedforward sections, while the feedback sections are only subjected to mutation.

4.3.3 Performing Mutation on Floating-point Genes

Mutation is implemented by occasionally perturbing a random element in a chromosome. It is mainly used to introduce variations in to the chromosomes. Global or local variations can be introduced with a probability α_m. In binary-coded GA, each bit of a bit string is replaced by a randomly generated bit if a probability test is passed [23, 16]. It can be achieved by flipping the bits in the strings with the mutation probability α_m. Unlike binary coded GA, the floating-point representation, particularly an EA using complex genes, requires a more sophisticated method to achieve mutation.

In this work, we employ a novel method to introduce perturbations into the elements of chromosomes. Each element of a chromosome is split into magnitude and angle and individually perturbed with a probability α_m:- either magnitude, angles or both can be selected for mutation with the probability test. A random function can be used to perturb these parameters within a desired range. For example, if the magnitude and angle are selected for mutation, they are perturbed through

$$|c_i(t+1)| = |c_i(t)| + V_{t,\sigma 1} \tag{4.24}$$

and,

$$\angle c_i(t+1) = \angle c_i(t) + V_{t,\sigma 2} \tag{4.25}$$

where $V_{t,\sigma 1}$ and $V_{t,\sigma 2}$ are random variables generated at time t with variances $\sigma 1$ and $\sigma 2$ respectively. Choosing a suitable random function (e.g. Gaussian, Uniform, $\cdots$ etc.) and its variance are the major tasks to design the mutation operator. For example, random function that has high probability distributions at the regions closer to the operating points is often preferred in order to avoid high population diversity to be met during the search. Population diversity occurs when the population in an EA diverges arbitrary from the optimal state where most of the genetic operators produce offspring that outperform their parents.

Considering the above issues in mind, a random generator with normal distribution is chosen as a mutation operator. The probability density function (pdf) of normal distribution is thoroughly studied [55]:

$$\text{pdf}(y) = \text{pdf}(x - \mu) = \frac{1}{\sqrt[2]{2\pi d}} \exp \frac{(-x - \mu)^2}{\sigma^2} \tag{4.26}$$

where μ and σ^2 are the mean and the variance respectively. The probability density curves of a normal distribution for various μ and d, where $d = \sigma$, are plotted in Figure 4.7. This bell-shaped curve is perfectly symmetric about a line perpendicular to the x-axis through the mean μ. This line bisects the bell exactly. Changing the mean μ merely translates the curve to the right or the left. Varying the standard deviation changes the shape of the bell. Thus changing the mean and standard deviation change the location and the shape of the curve, but it still remains a normal curve. It can be easily shown that, with $\mu = 0.0$ and $d = 0.5$, a x variation of approximately $\{-1 \leq x \leq 1\}$ can be obtained. The probability of generating a value, which lies below 0.5 is greater than that of a value that lies above 0.5.

Performing mutation on feedback parameters (either poles or zeros) is a much trickier process than doing on feedforward parameters. For example, in this work we wish to ensure the poles lie inside the stable region throughout the genetic search. An improper design can cause the poles to fall outside the stable region and will necessitate the use of stability monitoring techniques.

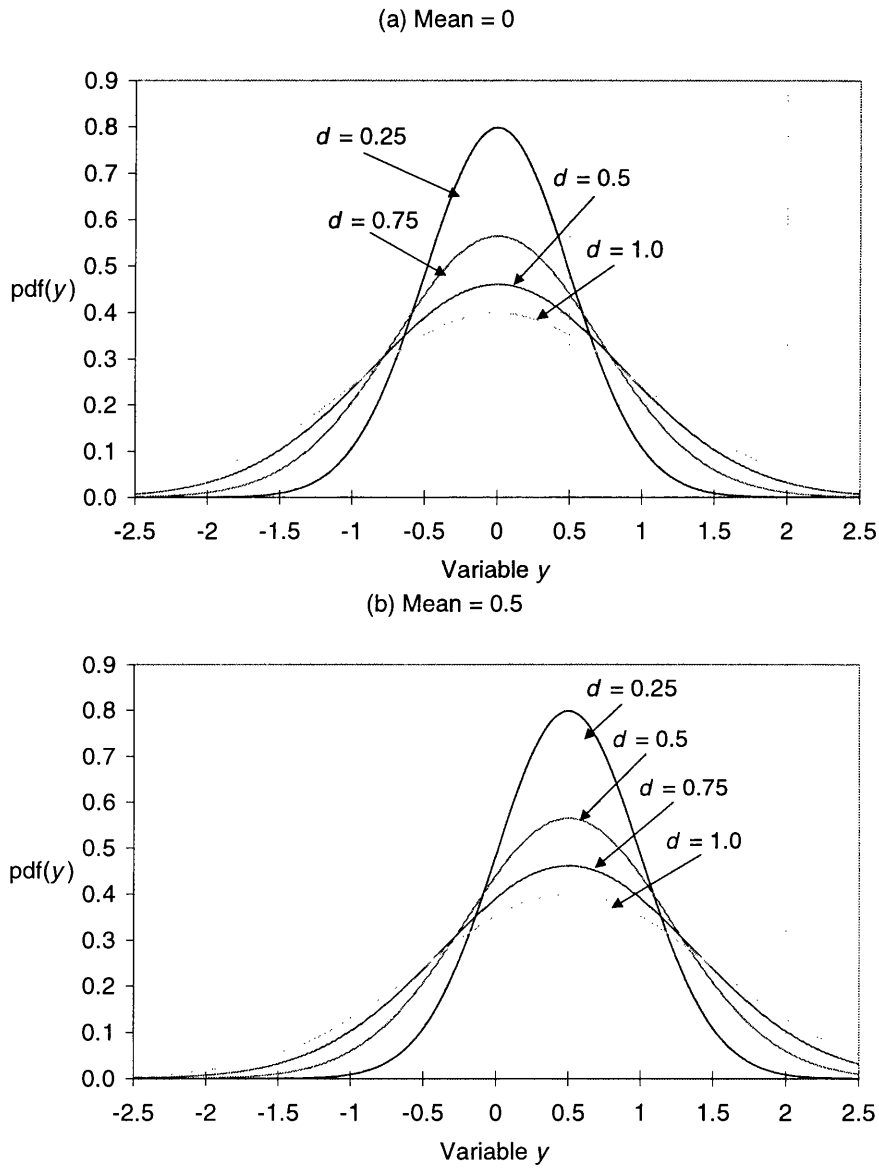

Figure 4.7: Probability density function (pdf) of Gaussian variable, y (where $y = x - \mu$), for various standard deviations d, where $d = \sigma$. (a) Gaussian distribution when $\mu = 0$ (b) Gaussian distribution when $\mu = 0.5$.

Considering the filter stability, we define the mutation operation on a parameter q as follows

$$\begin{aligned} |q_{t+1}| &= |q_t| + (q_{max} - |q_t|)g_t(0, \sigma 1) \quad \text{if } g_t(0, \sigma 1) \geq 0 \\ &= |q_t| + |q_t| g_t(0, \sigma 1) \quad \text{if otherwise} \end{aligned} \tag{4.27}$$

$$\begin{aligned} \angle q_{t+1} &= \angle q_t + (\pi - \angle q_t)g_t(0, \sigma 2) \quad \text{if } g_t(0, \sigma 2) \geq 0 \\ &= \angle q_t + (\angle q_t + \pi)g_t(0, \sigma 2) \quad \text{if otherwise} \end{aligned} \tag{4.28}$$

where $g_t(0, \sigma)$ is a Gaussian function with zero mean and a standard deviation σ at time t, $|q_t|$ and $\angle q_t$ represent the magnitude and the phase angle of the complex parameter q estimated at time t, and q_{max} is the maximum range of the parameter's magnitude. If q represents a pole, the maximum range q_{max} assigns the value 1. The range q_{max} could be any positive number for feedforward coefficients. However, choosing an appropriate range allows the GAs to converge with a fast rate. In this work, a value 5 was assigned to q_{max}, which limits the search space of the numerator-coefficient's magnitude within $\{0 \leq |q| \leq 5\}$. The phase angles are chosen to lie within $\{-\pi \leq \phi \leq +\pi\}$ regardless of the type of parameter (either poles or feedforward coefficients) as represented in equation . It has been shown earlier that with a standard deviation $d = 0.5$, Gaussian variables can be generated in the range $\{-1.0 \leq x \leq +1.0\}$. This ensures that the poles always lie inside the stable region. However, a variance, which is smaller than 0.5 is much preferred when the algorithms approaches the global region. This will facilitate the algorithm to fine-tune the parameters. Large variations are only needed to find the global region, which is embedded within the multiple local optima. Once the global area has been found the variance must be reduced to a smaller value. However, in this work a fixed, higher variance, $d = 0.5$, is used to avoid the occurrence of premature convergence. Premature convergence occurs when the population in a GA reaches such a sub-optimal state that most of the genetic operators can no longer produce offspring that outperform their parent [34, 9].

Filter Stability During Evolution

Note that since we use the poles directly in the chromosome, the ensuring of stability becomes trivial. In particular, this work ensures that poles positions are unaffected by mutation. Hence stability monitoring is not needed as in the case of coefficients design methods. This greatly reduces the computational costs while achieving the capability of designing complex adaptive filters.

4.4 Applications for Adaptive System Modelling

One popular application of the adaptive filter is in the area of system modelling. System modelling can be broadly classified into two types: direct system modelling and indirect or inverse system modelling [6]. In direct system modelling, the aim of the adaptive process is to model the transfer function of the system that is to be identified. Conversely, the aim of inverse system modelling is to model the transfer function of the system that is being identified by adaptive filtering. In this chapter, we show two simple examples of system modelling techniques that have been used to demonstrate the convergence properties of the proposed EA.

4.4.1 Direct System Modelling

The experimental arrangement of direct system modelling is made as shown in Figure 4.8. In direct system modelling, we use a simple second-order IIR system whose transfer function is expressed in z-domain as

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^2 - 2 + b_3 z^{-3} + b_4 z^{-4}}{(1 - p_1 z^{-1})(1 - p_2 z^{-2})(1 - p_3 z^{-2})} \tag{4.29}$$

where

$$\begin{cases} b_0 = 0.4,\ b_1 = -0.3,\ b_2 = 0.5,\ b_3 = 0.4, b_4 = 0.2, \\ p_1 = 0.6,\ p_2 = -0.5,\ p_3 = -0.4 \end{cases} \tag{4.30}$$

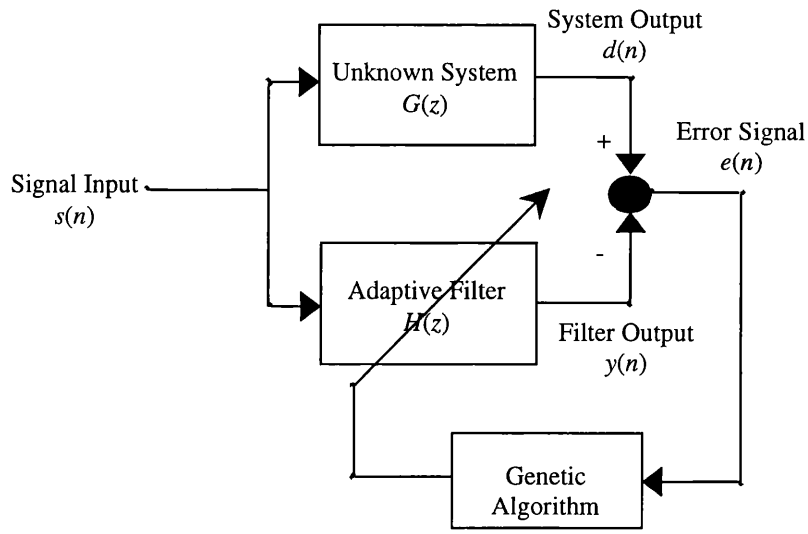

Figure 4.8: Experimental arrangement of direct system modelling.

Mean square error and MME performances of this model is investigated; the results show that the surface has several multiple local optima. Gradient algorithms often fail to optimise this surface. Figure 4.9 illustrate the local error surfaces plotted against b_2 and p_3 for various choices of b_4 using 100 training samples.

A spectrally white Gaussian noise,

$$s(n) = g_n(0, 1) \tag{4.31}$$

with zero mean and unity variance is applied as an input to model the unknown system $H(z)$. Timing waveform of the input signal is shown in Figure 4.10.

4.4.2 Inverse System Modelling

Figure 4.11 illustrates the experimental arrangement of the inverse system modelling. In this experiment we use a fourth-order IIR channel whose z-transfer function, $C(z)$, is

$$C(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})(1 - p_3 z^{-1})(1 - p_4 z^{-1})} \tag{4.32}$$

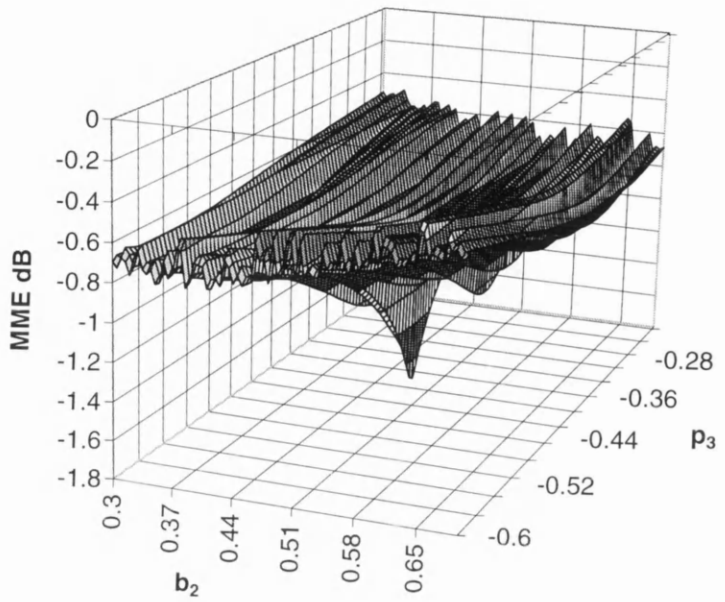

Figure 4.9: Local MME surfaces while modelling the system represented by equation (4.29).

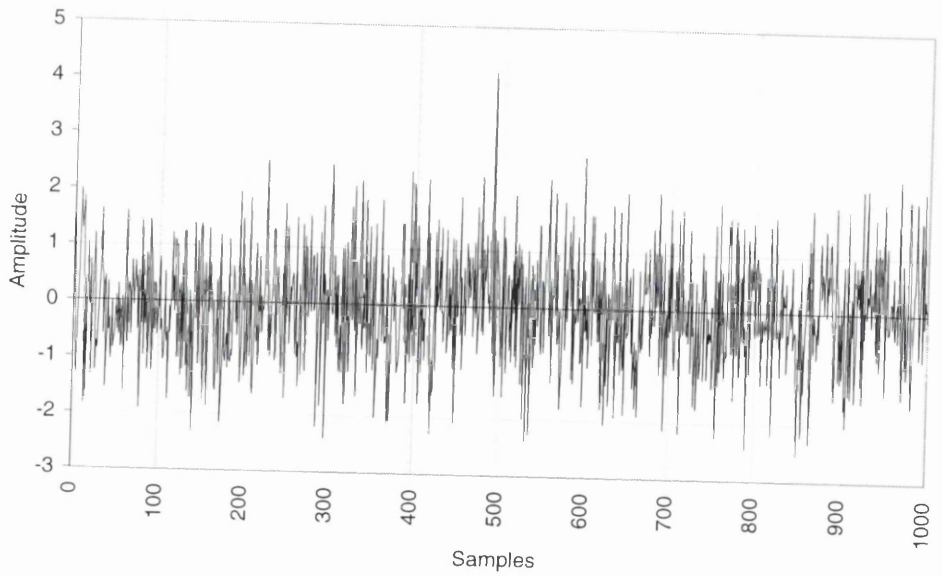

Figure 4.10: Input signal applied to the direct system modelling. The signal is a Gaussian distribution function with zero mean and unit variance.

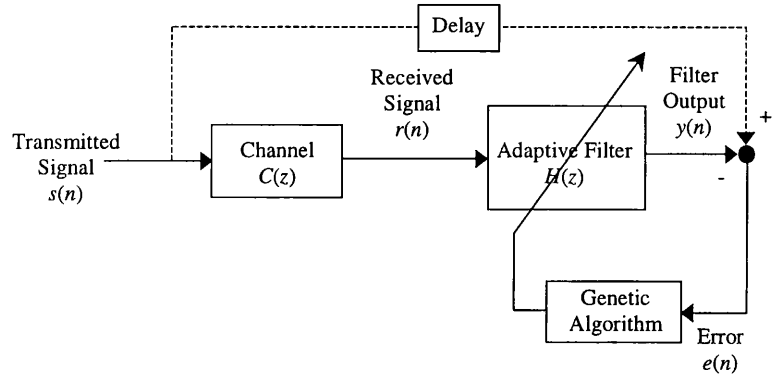

Figure 4.11: Experimental arrangement of inverse system modelling.

where $\{b_i,\ i = 0 \cdots 2\}$ and $\{p_j,\ j = 1 \cdots 4\}$ are complex parameters and are assumed as

$$\begin{cases} b_0 = 1.0, b_1 = 0.6e^{j0.5\pi}, b_2 = 0.6e^{j1.2\pi}, p_1 = 0.86e^{-j0.5\pi}, \\ p_2 = 0.75e^{-j1.2\pi}, p_3 = 0.8e^{-j0.75\pi}, p_4 = 0.7e^{j1.5\pi} \end{cases} \tag{4.33}$$

This kind of channel with complex response can be seen in the transmission of QAM signals through modem channels. In such areas, the actual signal is down converted into lower frequency in order to reduce the signal bandwidth and hence the sampling rate. The transmitted signal in this example is a complex FM signal

$$s(n) = Ae^{\{j\omega_c T + \theta(n)\}} \tag{4.34}$$

where $s(n)$ is complex-valued, A is the real amplitude, ω_c is the centre frequency, T is the sampling interval, and $\theta(n)$ is the baseband signal, $\theta(n) = 0.25\sin(0.01\pi n)$. The signal $s(n)$ has a constant envelope since A is a constant and the exponential has a modulus of unity. For the sake of simplicity, it is assumed in the experiments that the training signals are available during evolution and the signals of our interest have no noise impairments.

4.4.3 Verification

A series of simulations is carried out to test the convergence performance of the proposed EA techniques with parameters such as population-size N, probability

of mutation α_m, and probability of crossover α_c to model direct and inverse systems. The aim of the experiment is six folds:

1. Investigate the effect of crossover on EA's convergence.
2. Compare the split-point crossover operation with the standard uniform crossover.
3. Compare the convergence performance of Gaussian mutation distribution against uniform distribution.
4. Investigate the effect of crossover when applied within feedback sections.
5. Investigate the effect of mutation rate on EA's convergence.
6. Compare the convergence performance of various cost (objective) values which can be used in genetic search approach.

The experiments presented in this chapter use a fixed population size $N = 50$ along with the tournament selection scheme with a tournament-size $s = 2$. Gaussian distribution is used throughout this work unless otherwise specified and the training signals of 100 samples are used to obtain the fitness function. The EA is allowed to run over 32000 generations and the MSE of the best member in the population is measured against the generation number. The convergence curves which shows the variations of objective function values against the generation number are called learning curves.

Experiment 1: This experiment is carried out to show the necessity of the crossover operation in genetic evolution. Standard uniform crossover is employed along with $\alpha_m = 0.02$ and $\sigma = 0.5$. A set of learning curves is obtained when α_c is set to 0 and 1.0 respectively. Figure 4.12 shows these learning curves which are obtained while modelling direct and inverse systems. These curves clearly show the necessity of crossover operation in genetic evolution. An evolutionary algorithm, which uses the crossover operation, converges faster than an EA without the crossover. The EA without a crossover operator is analogous to EP in that it takes longer evolution time to reach a better solution.

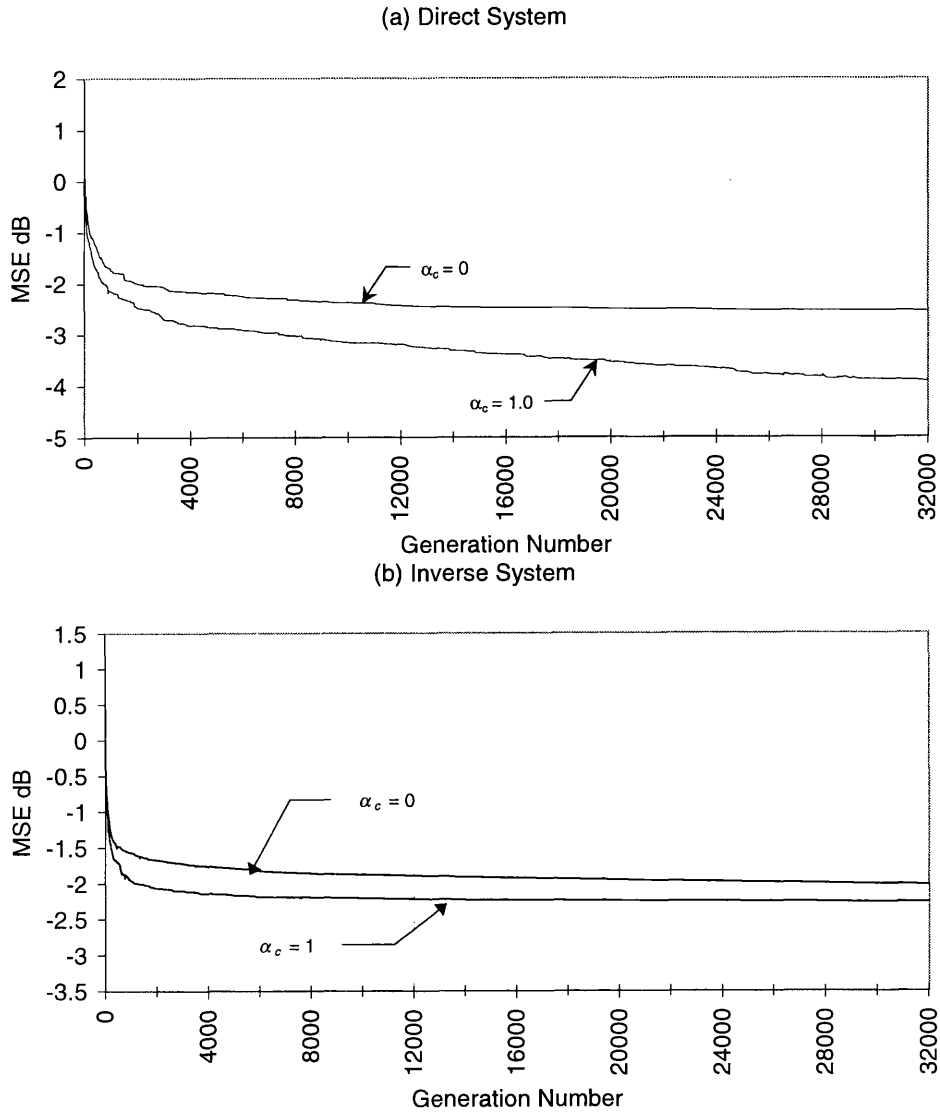

Figure 4.12: Effect of crossover operation on EA's convergence.

Parameter	Original Values	$\alpha_c = 1.0$	$\alpha_c = 0$
b_0	$0.4e^{j0}$	$0.3435e^{-j2.78x10^{-3}}$	$0.1786e^{j0.53}$
b_1	$0.3e^{j\pi}$	$0.2525e^{j3.1382}$	$0.2167e^{j3.04}$
b_2	$0.5e^{j0}$	$0.4375e^{-j4.916x10^{-4}}$	$0.6413e^{-j0.0128}$
b_3	$0.4e^{j0}$	$0.327e^{-j2.17x10^{-4}}$	$1.136e^{j1.4588}$
b_4	$0.2e^{j0}$	$0.1433e^{j4.067x10^{-5}}$	$0.1576e^{j0.0171}$
p_1	$0.6e^{j0}$	$0.4242e^{j3.1414}$	$0.5474e^{j3.6536x10^{-4}}$
p_2	$0.5e^{j\pi}$	$0.3412e^{j3.162}$	$0.4521e^{j2.9459}$
p_3	$0.4e^{j\pi}$	$0.5741e^{j1.2927x10^{-6}}$	$0.4961e^{j3.1388}$

Table 4.1: The best parameters as obtained from EAs when modelling direct system of parameters (4.30).

The crossover operation provides a better local search capability, which is much appreciated when the algorithm approaches the global region.

Table 4.1 shows the parameters of the best evolved filter as obtained from EAs when modelling the direct system of parameters (4.30).

Experiment 2: The experiment is aimed to compare the performance of the proposed split-point crossover with the standard uniform crossover. This experiment is conducted by setting the EA's parameters $\alpha_c = 1.0$, $\alpha_m = 0.02$ and $\sigma = 0.5$. A series of learning curves is obtained while modelling the direct and inverse systems. Figure 4.13 shows these learning curves.

The learning curves obtained when using split-point crossover converges to a minimum point, which is lower than that of standard uniform crossover. As split-point crossover produces more new members than the standard operator does, it is able to find a better solution very quickly. Table 4.2 compares the best parameter values as obtained from this experiment with those of standard methods used in experiment 1.

Although the convergence curves of direct system modelling are closer to each other, it provides distinct solutions. The parameters which are obtained from split-point crossover are very closer to the optimum parameters given in (4.30).

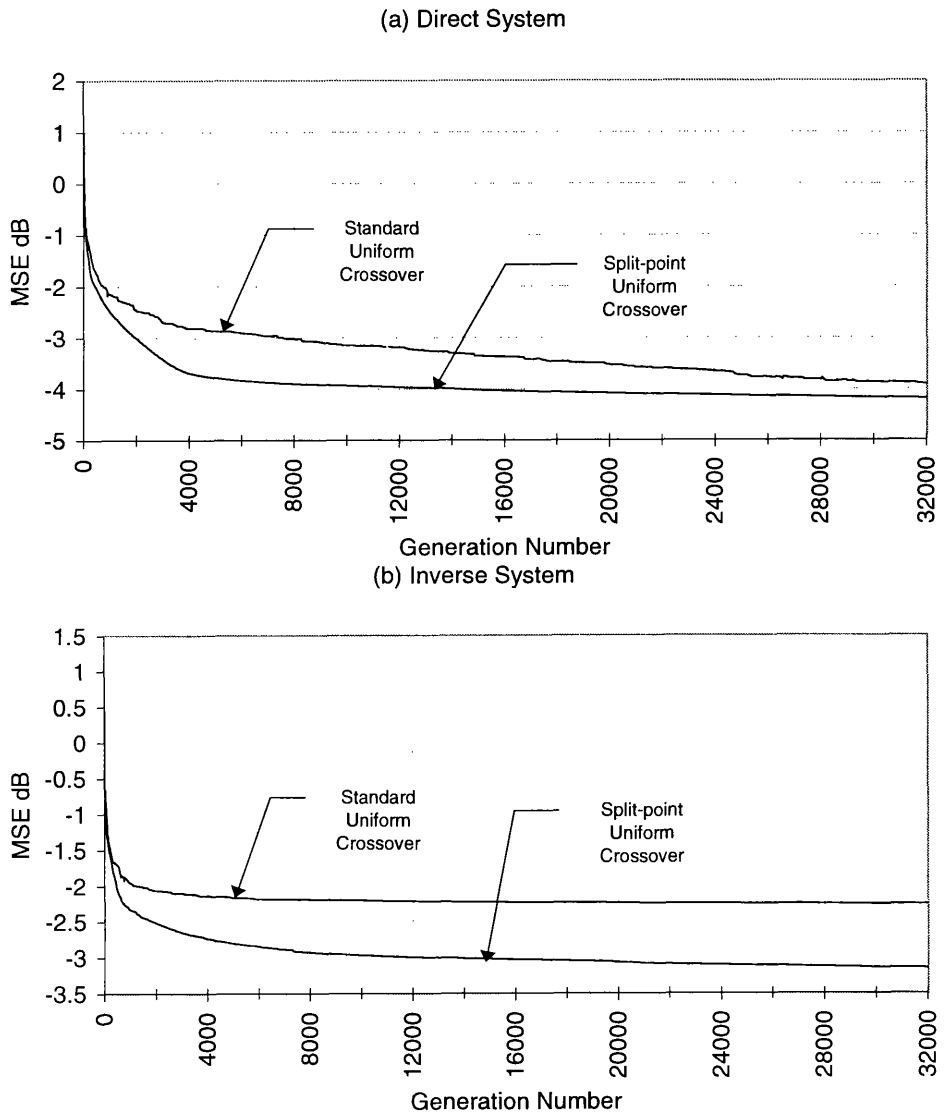

Figure 4.13: Learning curves showing the effect of split-point crossover on EA's convergence.

Parameter	Original Values	Split-point Crossover	Standard Crossover
b_0	$0.4e^{j0}$	$0.3706e^{j3.3169x10^{-4}}$	$0.3435e^{-j2.78x10^{-3}}$
b_1	$0.3e^{j\pi}$	$0.2645e^{j3.112}$	$0.2525e^{j3.1382}$
b_2	$0.5e^{j0}$	$0.467e^{j1.27x10^{-5}}$	$0.4375e^{-j4.916x10^{-4}}$
b_3	$0.4e^{j0}$	$0.368e^{3.92x10^{-4}}$	$0.327e^{-j2.17x10^{-4}}$
b_4	$0.2e^{j0}$	$0.214e^{j1.02x10^{-5}}$	$0.1433e^{j4.067x10^{-5}}$
p_1	$0.6e^{j0}$	$0.487e^{j3.143}$	$0.4242e^{j3.1414}$
p_2	$0.5e^{j\pi}$	$0.388e^{j3.112}$	$0.3432e^{j3.152}$
p_3	$0.4e^{j\pi}$	$0.591e^{j2.3x10^{-5}}$	$0.5441e^{j1.2927x10^{-6}}$

Table 4.2: Parameters of the best evolved filter as obtained from EAs when split-point crossovers are applied to evolve the direct system of parameters (4.30)

Another important observation is that performing crossover between feedback sections degrades the algorithm's convergence performance. This is because during evolution the poles can lie within the filters in any manner and give the same fitness values. Figure 4.14 illustrates the learning curves when the crossover operations are employed to the feedback sections.

Experiment 3: This experiment is aimed to compare the convergence behaviour of uniform mutation distribution against Gaussian distribution. Split-point uniform crossover was employed along with $\alpha_m = 0.02$, $\alpha_c = 0.85$ and $\sigma = 0.5$. Figure 4.15 shows the learning curves, which are obtained, while modelling direct and inverse systems.

Gaussian distribution provides a faster convergence than uniform distribution.

Experiment 4: This experiment is conducted to show the effect of mutation probabilities on the EA's convergence. A set of mutation probabilities $\{\alpha_m = 0.005, 0.02, 0.06\}$ are applied with the proposed split-point crossover. The other genetic parameters such as crossover-probability α_c and the variance σ are set to 0.85 and 0.5 respectively. A set of learning curves is obtained for various mutation probabilities. The experimental results are shown in Figure 4.16. These results clearly show how the mutation probability is influenced on EA's convergence. It is observed that a moderate mutation rate ($\alpha_m = 0.02$)

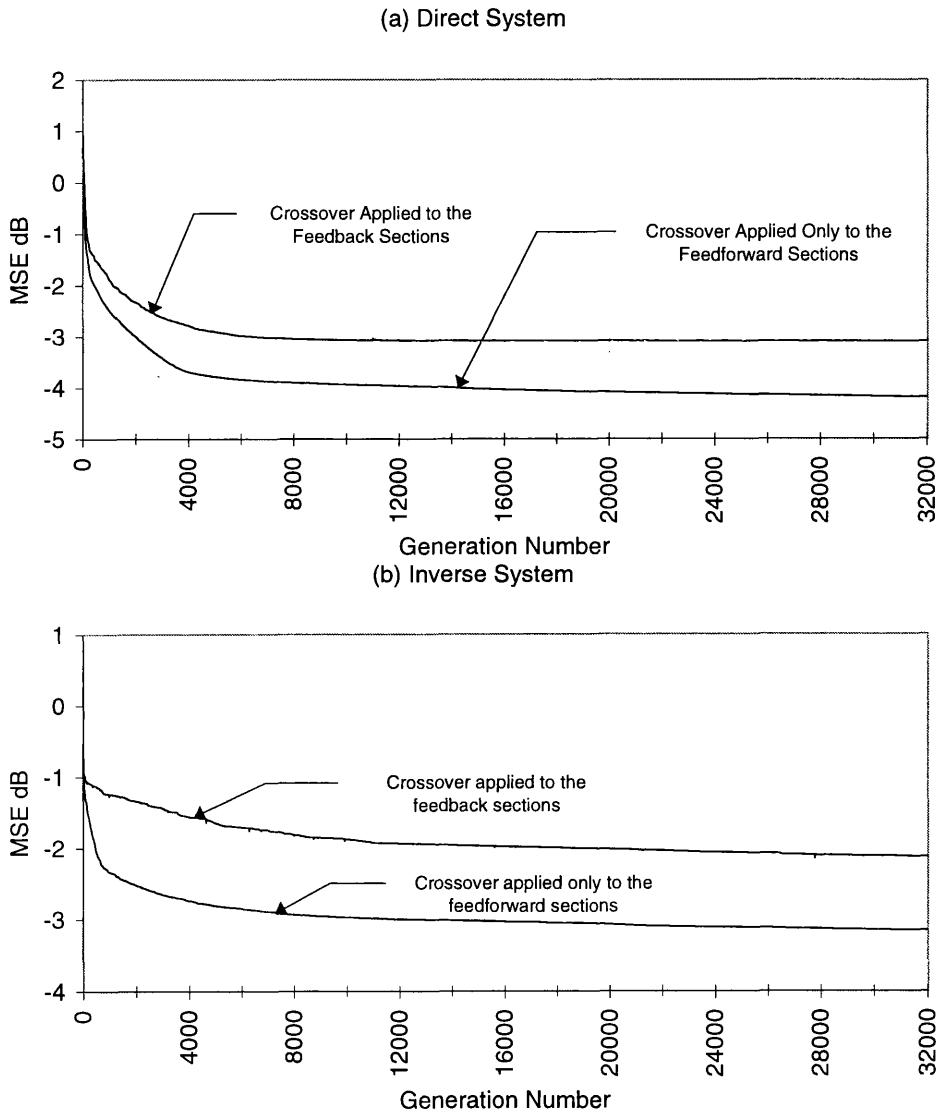

Figure 4.14: Effect of crossover when employed to the feedback sections of the IIR filters.

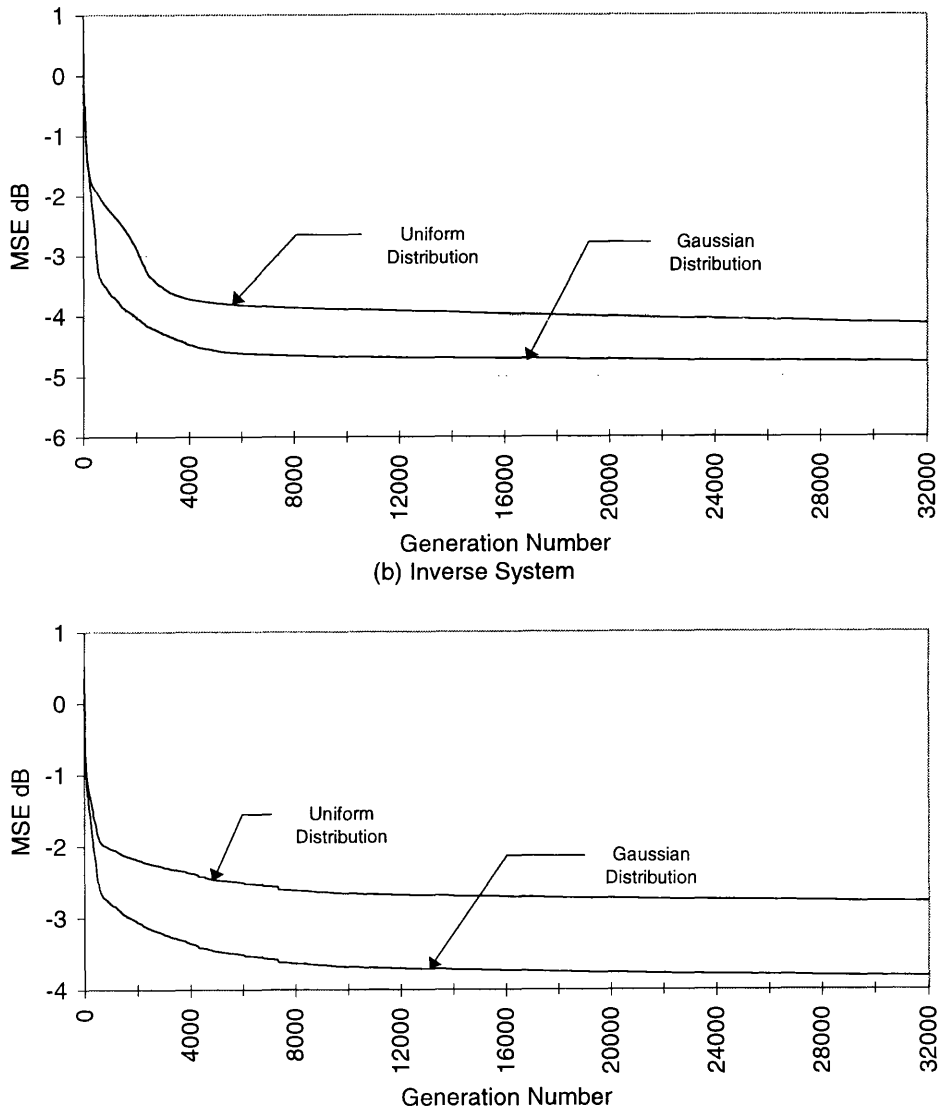

Figure 4.15: Effect of mutation distribution on EA's convergence.

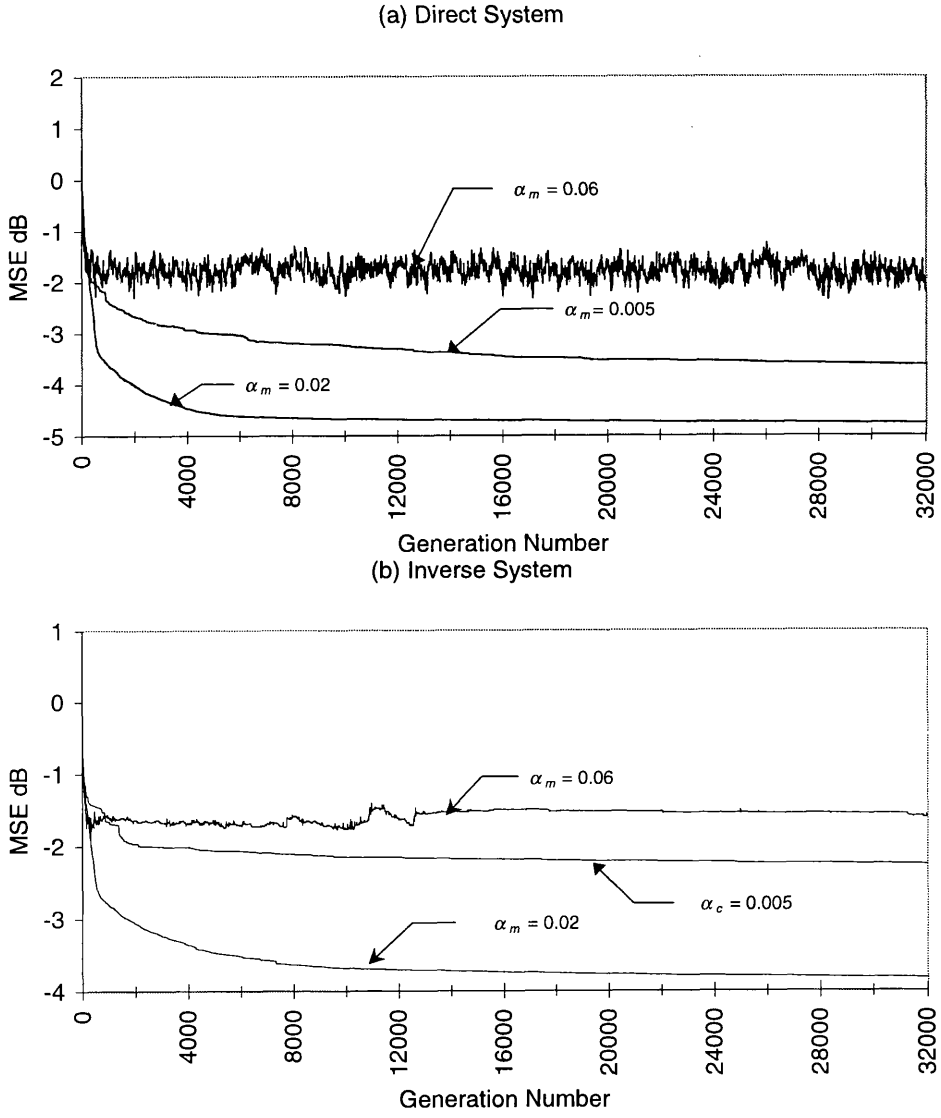

Figure 4.16: Learning curves showing the effect of mutation.

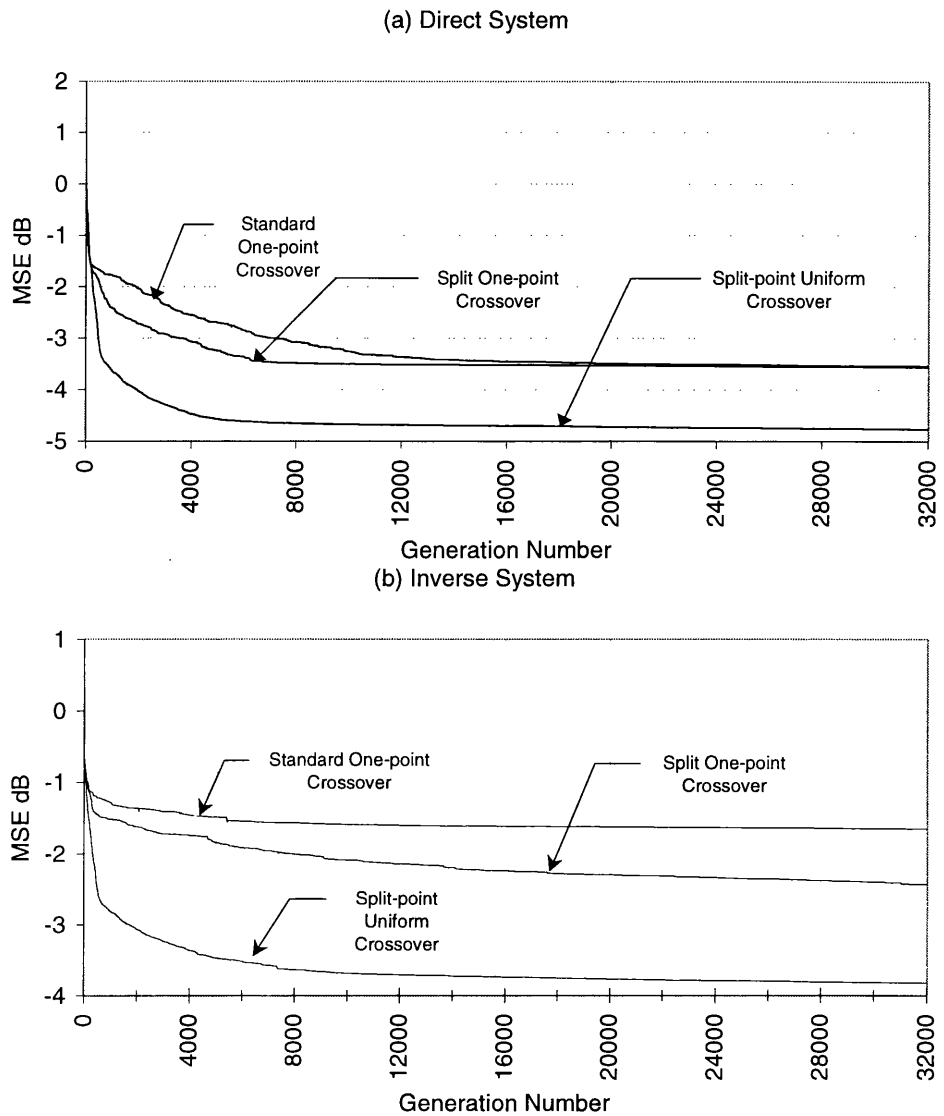

Figure 4.17: Learning curves comparing the uniform crossover with the one-point crossover.

contributes good performance, while too large or too small degrades the performance, i.e. smaller mutation rate causes premature convergence while larger values provide high population diversity.

Experiment 5: This experiment is carried out to compare the performance of one-point crossover against uniform crossover. Uniform crossover produces more new members than one-point crossover and converges faster. One-point crossover requires much more time to reach a better solution. The comparative curves showing the convergence are illustrated in Figure 4.17.

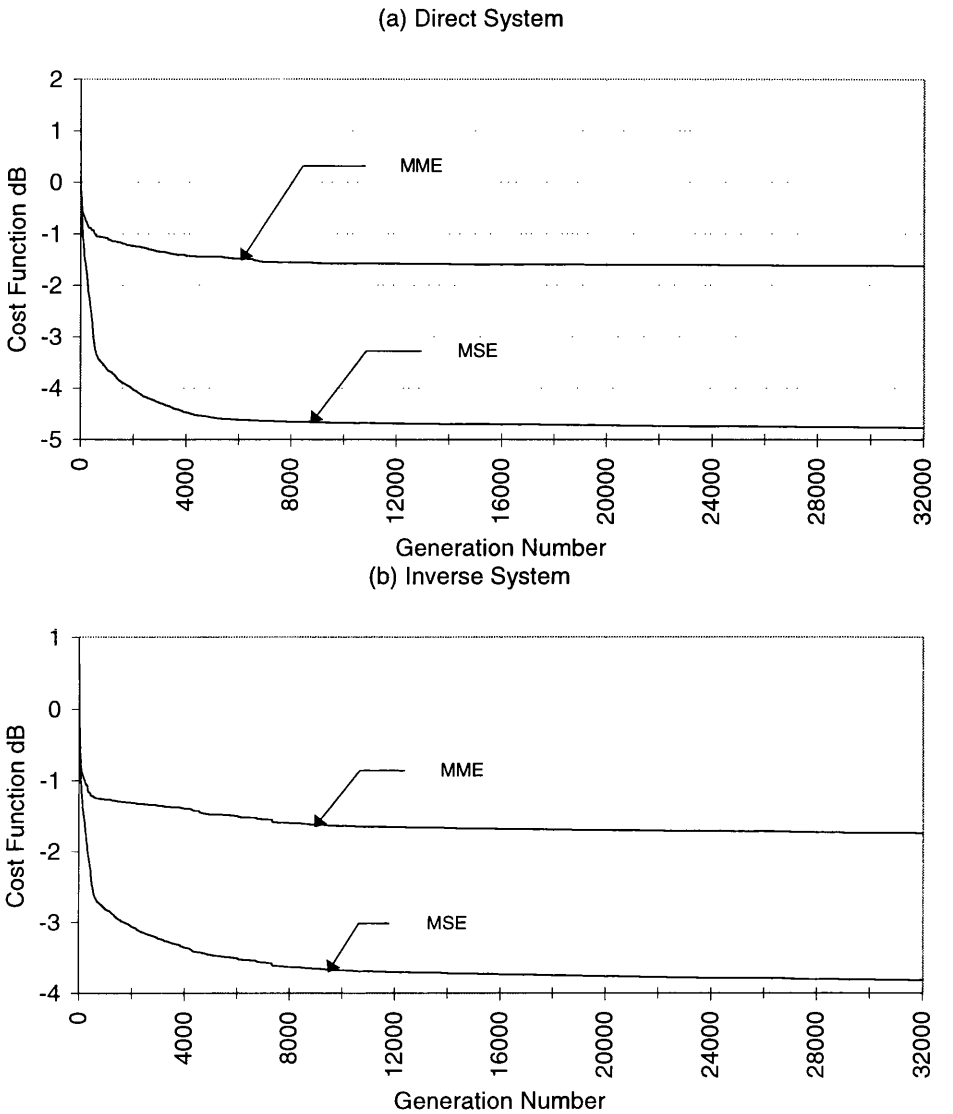

Figure 4.18: Learning curves showing the convergence performance of various cost functions.

Experiment 6: Finally, an experiment is carried out to analyse two objective functions that can be applied to the proposed EA techniques. The objective functions such as MSE and MME are used for this analyse. A set of learning curves is obtained by keeping $\alpha_c = 0.85$, $\alpha_m = 0.02$ and $\sigma = 0.5$. The optimum filter parameters estimated have also been tabulated for the comparison. The filter parameters of the optimum filters evolved using those objective functions are tabulated. The learning curves which show the convergence performance of each cost function are illustrated in Figure 4.18. It can be seen that the MME

Parameter	Original Values	MSE	MME
b_0	$0.4e^{j0}$	$0.3723e^{j2.31x10^{-5}}$	$0.363e^{j1.14x10^{-5}}$
b_1	$0.3e^{j\pi}$	$0.2821e^{j3.26}$	$0.2834e^{j3.21}$
b_2	$0.5e^{j0}$	$0.475e^{j9.37x10^{-6}}$	$0.462e^{j5.03x10^{-3}}$
b_3	$0.4e^{j0}$	$0.423e^{4.17x10^{-5}}$	$0.376e^{j1.16x10^{-6}}$
b_4	$0.2e^{j0}$	$0.189e^{j2.11x10^{-6}}$	$0.221e^{j0.8x10^{-6}}$
p_1	$0.6e^{j0}$	$0.587e^{j3.7x10^{-7}}$	$0.388e^{j3.11}$
p_2	$0.5e^{j\pi}$	$0.391e^{j3.16}$	$0.486e^{j3.13}$
p_3	$0.4e^{j\pi}$	$0.495e^{j3.15}$	$0.588e^{j1.11x10^{-7}}$

Table 4.3: Comparing the Parameters of the best evolved filters as obtained from EAs when various cost functions are employed to evolve the direct system of parameters (4.30)

criterion gives same amount of convergence performance as the standard MSE criterion. Table 4.3 shows the best parameter values obtained for direct system modelling.

A special feature of the evolutionary technique is that the capability of using any cost functions with ease. This property distinguishes the EAs from classical techniques in that they are strongly bounded with the cost functions. The results, which are shown in this report, are that the average values taken over 15 independent EA runs with distinct initial conditions. For example, consider the following Figure 4.19 which illustrates how the average value is obtained for 4 independent EA runs when modelling the direct system (4.29). The EA cycle applied to these experiments is summarised below:

Evolution Cycle

The initial set of parameters (magnitude and phase angles) is selected randomly. Let us consider the number of parameter set (Population size) is N. The magnitudes of the poles are selected within the range $[0 \cdot \cdot 1]$ in order to ensure that the systems lie in the stable region. The phase angles are selected randomly between $-\pi$ and $+\pi$ and the magnitudes of the feedforward coefficients are ensured within $[0 \cdot \cdot 5]$. The population is then subjected to a fitness measurement

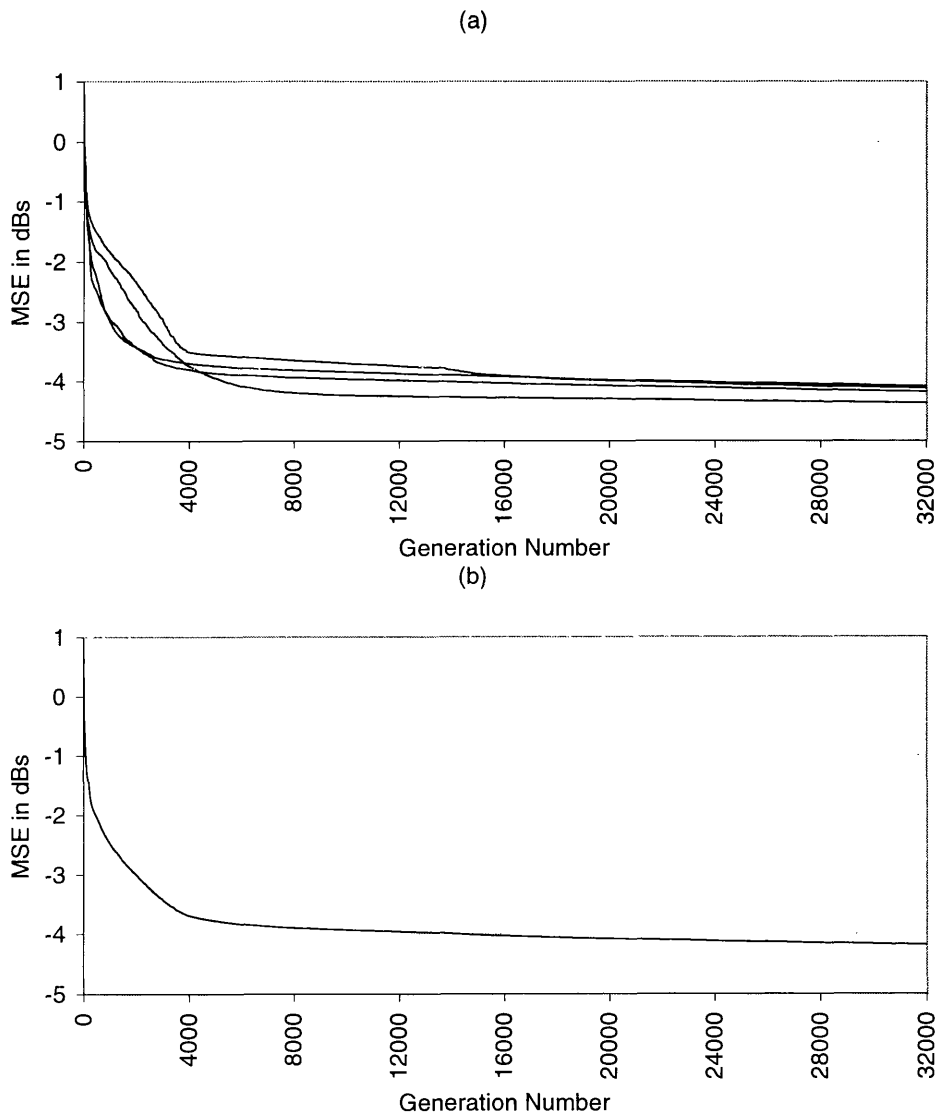

Figure 4.19: Figure illustrating how the average value is obtained from several EA runs. (a) Learning curves of 4 independent EA runs. (b) Average value of the curves shown in (a).

based on a cost function used for the IIR filtering application. Only the best chromosomes are selected and they survive to the next generation. Tournament selection with a tournament size $s = 2$, is used where two chromosomes are selected randomly from the population and the best becomes one of the next generation. This is repeated, until a complete intermediate population (parents) of size N is selected. The generation is modified using crossover and mutation and a new population (offspring) of the same size (N) is created. The offspring replaces the old population by keeping the best chromosome.

4.5 Summary and Discussion

Floating-point EAs have been developed for evolving complex adaptive IIR digital filters. In this work, direct form realisation has been considered without employing any stability monitoring techniques. It has been achieved by designing the poles through EAs. Simple algorithm has been employed to perform the conversion of poles into equivalent feedback coefficients. A new crossover scheme has been introduced which produces more new members at each generation and provides good local search capability when the algorithm approaches the global region. We also introduced a more sophisticated method to perform mutation on complex chromosomes. This approach ensures the IIR filters to lie inside the stable region during evolution. The cost functions such as MSE and MME have been successfully employed with the support of tournament selection scheme.

Along with the above salient features, the proposed techniques have two major drawbacks. First, the pole design method accommodates multiple optima with the same fitness values that can be obtained from re-ordering the poles within the filters. This property leads to slower convergence when the crossover operations were employed within the feedback sections.

Second drawback is that proposed techniques have provided poor fine-tuning capability as the algorithm approaches global region. This was the reason why

the final values as obtained from genetic evolution were not exactly matched to the optimum parameters. A major reason causing this poor performance may be of the higher variance of mutation operator. In this work, the variance σ^2 was chosen as 0.5 so as to provide a desired variations of $\{-1 \leq x \leq +1\}$ throughout the genetic search. This is considerably a higher value, which may degrade the local search capability of the EAs. Consequently, employing small variance throughout the genetic search can cause premature convergence. Even though many research outcomes have been published [44, 47] to find the optimum value of the variance with ease, they are either computationally expensive or inefficient. For example, one possible solution is to use evolvable variance for each parameter and are evolved along with the solutions. This approach increases the computational cost, therefore unsuitable for applications having large number of parameters. Alternatively, dynamic mutation can be considered to introduce high variations at the beginning of the search, and reduce along with the solution. The concept of dynamic mutation was introduced to reduce the severity of mutation as the global optimum is approached. This requires either the generation number or the fitness values as a function variable to calculate the variance of each string. Unfortunately, this approach has also been found as an inefficient method since the global optimum of an objective function cannot be determined from any methods.

The next chapter is aimed to resolve these problems with coefficient design approach.

Chapter 5

Direct Design Method via Global Evolution

In the previous chapter we have provided the design techniques of complex evolutionary algorithms. This approach facilitates the design of poles so that the adaptive IIR filters can be easily designed within the stable region. This method avoids the use of stability monitoring techniques and allows the mutation operator to move the feedback parameters within a desired search space throughout the genetic search. However, the techniques described in Chapter 4 has a major drawback, which include:

- Pole design method provides additional stationary points which are obtained when reordering the poles within the filter. Therefore, the crossover cannot be applied within the feedback sections. Applying crossover within feedback sections degrades the convergence performance of the EAs.

To resolve this problem, the coefficients can be designed directly from a population. This can facilitate the use of all the features of crossover.

The aim of this chapter is twofold:

1. To show the recent development of designing the coefficients of the adaptive IIR filters using simple stability monitoring techniques.

2. To compare the convergence performance of the coefficient design method

against pole design approach when modelling the same signals and systems which have been used in the previous chapter.

In this work the coefficients of the adaptive filters are designed directly from a population. The adaptive filters are developed as direct, cascade, parallel and lattice form structures using floating-point EA techniques as discussed in Chapter 4.

The rest of this chapter is organised as follows. In Section 5.1, the problems that can arise when designing the coefficients are described. A special feature of floating point EAs is outlined and the technique of designing mutation operator is discussed. The problem of premature convergence is delineated and an overview of various remedies is given. This section also provides the operation of new genetic cycle, which employs a new operator called immigrant with an aim to avoid the severity of premature convergence. Section 5.1.2 introduces a new correction method that will improve the convergence performance of the EAs. This section also outlines the problems of alternative realisations. Convergence behaviour while modelling various IIR models using several alternative realisations is compared and the reasons for choosing direct form filters is illustrated. Section 5.2 provides the design techniques of direct form IIR structures using simple stability ensuring methods. In this section we show how the filter stability can be determined from the estimate of filter output. This section also compares the convergence performance of the coefficient design method against pole design approach using the same systems and signals as shown in previous chapter. Finally, this chapter provides a brief summary of these works with suitable conclusions.

5.1 Stability, Design and Evolutionary Method

A unique feature of the floating-point EA techniques is that being able to search from problem space rather than from an encoded range [25]. As mentioned in the previous chapter, the crossover operator employed in floating-point EAs can no longer produce new members with values greater than those of the parents before crossover. Only mutation can move a parameter to any desired value. For example, consider a standard mutation performed on a parameter c as follows:

$$c(t+1) = c(t) + g_t(0, \sigma) \tag{5.1}$$

where $c(t+1)$ represents the parameter value after mutation and $g_t(0, \sigma)$ represents a Gaussian variable with zero mean and variance σ^2 respectively. The range of variations to be introduced into the parameter can be varied with the variance σ^2. A larger variance is needed to identify the global region of a multimodal error surface without premature convergence. Smaller variance is often required to fine-tune the parameter as the algorithm approaches the global solution. Most evolutionary algorithms, therefore, use meta-evolutionary techniques which provide dynamic mutation during adaptation [13]. These methods employ variable variance, which is calculated either from fitness values or from the generation number. For example, the variance of mutation, which is calculated from the generation number, can be defined as:

$$\sigma(t+1) = \sigma(t)\left(1 - r^{m(t)}\right) \tag{5.2}$$

where $\sigma(t+1)$ represents the variance after the parameters are evolved, r is a uniform random number from $\{0 \cdots 1\}$, $m(t)$ is a time function which is described by:

$$m(t) = \left(1 - \frac{t}{T}\right)^b \tag{5.3}$$

where t represents the generation number, T is the maximum number of generation, and b is a system parameter determining the degree of dependency on

iteration number. A major drawback of this method is to define the maximum number of generation as a function argument, which is normally an unknown value. Therefore, this method cannot be considered as a better approach to achieve a desired solution with appropriate fine-tuning as an algorithm reaches a global region.

In this work we choose a smaller value of variance for mutation in order to keep the coefficient values within the neighbourhood. As such, they often fail to converge to an optimum solution and converge to a sub optimal state causing inaccurate solution due to premature convergence. The convergence performance becomes much worse when the EAs operate within small population sizes.

5.1.1 Premature Convergence

Premature convergence or loss of population diversity before optimal or at least stationary values have been found has long been recognised as a serious failure mode for EAs [34]. The premature convergence occurs if the population sizes are small or the mutation cannot introduce enough variations to escape from a local solution. The latter becomes as a major reason causing premature convergence when designing the feedback coefficients of the adaptive IIR filters using floating-point EAs techniques. For example, consider the MSE surface shown in Figure 5.1, which is obtained while modelling a system,

$$H(z) = \frac{0.5 - 0.4z^{-1} + 0.89z^{-2} + 0.7z^{-3} - 0.9z^{-4}}{1 - 0.05z^{-1} - 0.85z^{-2}} \tag{5.4}$$

using an adaptive filter

$$A(z) = \frac{0.5 - 0.4z^{-1} + b_2z^{-2} + 0.7z^{-3} - b_4z^{-4}}{1 - 0.05z^{-1} - a_2z^{-2}} \tag{5.5}$$

The MSE is plotted against a_2 and b_4 for various choices of b_2. This local error surface has several minima in which global solution exist at $\{a_2 = 0.85, b_4 = -0.9\}$. The local optima are separated faraway from each other. So, if the algorithm stuck in one of these minima, mutation must provide a high enough variation to escape the algorithm from this inaccurate solution.

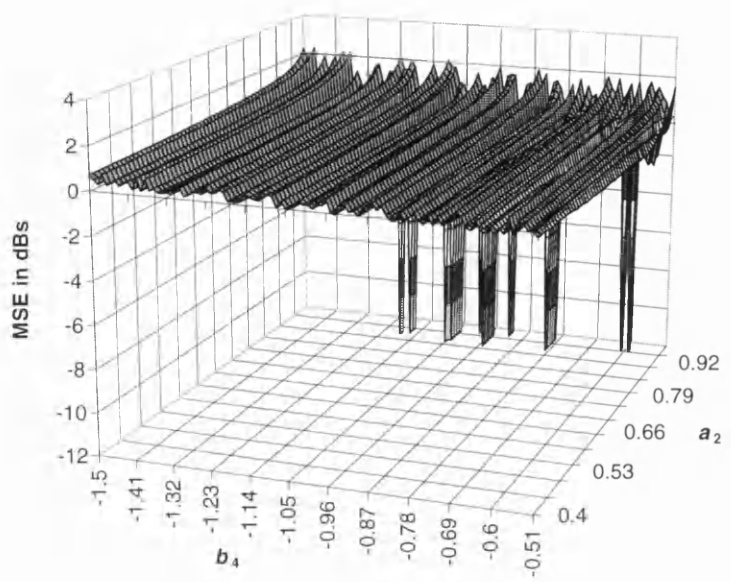

Figure 5.1: A MSE plot showing local minima separated faraway from each other.

In coded form of EAs, for example in binary GAs, each coefficient or parameter is encoded within a range, which is chosen by the GAs designer. In such a case, the coefficients or the parameter values can be moved within a wider range during evolution and probability of desolating in local optima is much less.

The problem of premature convergence cannot be avoided in EAs. Several methods have been proposed to combat premature convergence in binary GAs [9]. These include, for example, the restriction of mating procedure, recombining process, and replacement strategies [34, 9]. These techniques are summarised below:

Mating Process: Chromosomes can be selected so as to maintain population diversity. The goal of this process is to prevent similar individuals from mating [9]. Dissimilar individuals are chosen for recombination so that offspring produced by these diverse parents will tend to be more diverse [16]. Goldberg provided a sharing function, which reduces the fitness of individuals as a function of how similar they are to other individuals in the population, for an indirect mating strategies [9]. In direct mating strategies, individuals are

randomly paired, but are only mated if their hamming distance is above a certain threshold. The threshold is initially set to the expected average hamming distance of the initial population, and then is allowed to drop as the population converges. This mating strategy has a major drawback in that more schemata can be disturbed by crossover since fewer schemata are shared.

Recombination Strategies: Crossover operation can also be modified to introduce population diversity. If the crossover helps to produce offspring that are dissimilar from both parents, the resulting population will tend to be more diverse. There are several possibilities to achieve population diversity via crossover:

- increase the rate of crossover
- use a more disruptive crossover operator (e.g. uniform crossover)

Replacement Strategies: The goal of this process is to replace similar individuals in the parent population by new chromosomes [72]. De Jong first introduced this in 1989, hence known as De Jong's scheme. The second approach is that GAs can only add a new individual to the population if it is not identical to any member already in the population [72, 80].

By considering above three issues in mind we define the genetic cycle for floating-point EAs as follows:

A population of filter objects is selected randomly so that the initial values are ensured within the stable region. The population is then subjected into fitness measurement. The better individuals (parents) are selected via tournament selection scheme. The parents are shuffled before matting the individuals for crossover. Shuffling can simplify the process of pairing dissimilar individuals for recombination - adjacent individuals can be combined with ease. A set of new chromosomes replaces current population with a probability called *probability of immigration* α_i. These newly introduced chromosomes are called *immigrants* which are produced by introducing higher variations into each element of the

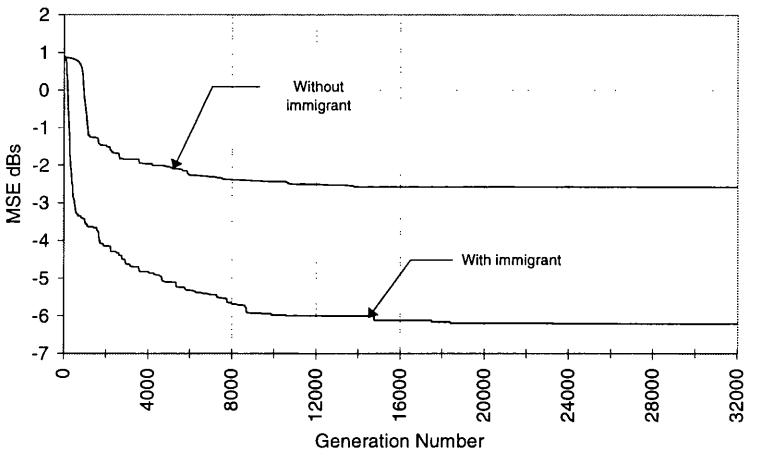

Figure 5.2: Convergence curves showing the effect of immigrants while modelling the system (5.6) by an equivalent order adaptive filter.

selected individuals. The probability of immigration determines which individual will be chosen for the immigration. The variations are introduced by a random function, in this work we use uniform random function with variation $[-0.25 \cdot \cdot 0.25]$. Split-point uniform crossover and Gaussian mutation is used to produce offspring. The standard deviation of Gaussian mutation is chosen as $\sigma = 0.005$ to provide small variations in order to keep the coefficients within their neighbourhood. The offspring replaces the old population by keeping the best member.

Figure 5.2 illustrates an example which shows the convergence behaviour of the EA while modelling a second-order IIR system of transfer function

$$H(z) = \frac{0.5 - 0.5z^{-1} + 0.89z^{-2}}{1 - 1.4z^{-1} + 0.98z^{-2}} \tag{5.6}$$

by an equivalent order adaptive IIR filter. This system has poles at $\{p_{1,2} = \pm 0.9899\}$ which are very close to the unit circle. Local error surface of this modelling problem is shown in Figure A.1. This particular modelling problem is a unique example to show severe multiple local minima in the local error surface. Convergence through conventional EAs often fail while optimising the parameters of (5.6) [3]. Figure 5.2 clearly shows that EAs without immigration operator converges to a sub-optimal state and causes inaccurate solution. Figure 5.3 illustrates another example which is obtained while modelling (5.4) by

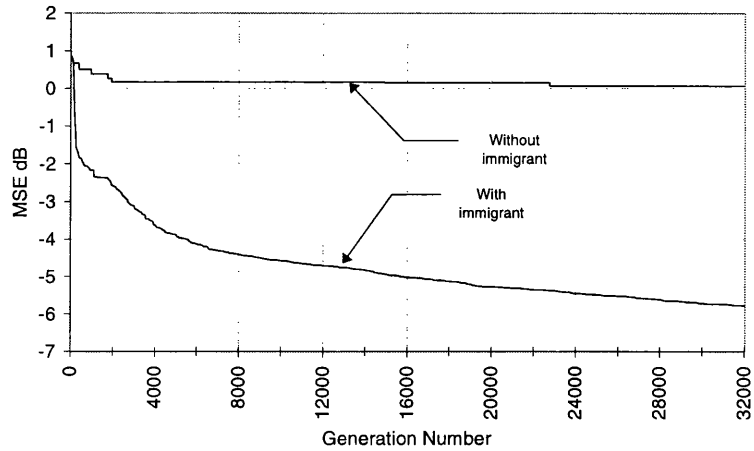

Figure 5.3: Convergence curves showing the effect of immigrants while modelling the system (5.4) by an adaptive filter (5.5)

(5.5). Local error surface while modelling this system is shown in Figure 5.1. The algorithm without immigration always converges to inaccurate estimates, which give higher MSE vales. Immigration operator provides enough population diversity and the problem of dislocating in local minima is much less.

From the knowledge of our previous examples, the experiments illustrating the convergence improvement by immigrant operator are obtained by using the following EA's parameters:

- Population size $N = 100$
- Probability of crossover $\alpha_c = 0.95$
- Probability of mutation $\alpha_m = 0.02$
- Probability of immigration $\alpha_i = 0.02$

It is important to note here that probability of immigration plays an important role in EAs convergence. Too small or too high values can degrade the convergence performance. For example, consider the Figures 5.4 and 5.5 which show the convergence behaviour for various rate of immigration while modelling (5.6) and (5.4) respectively.

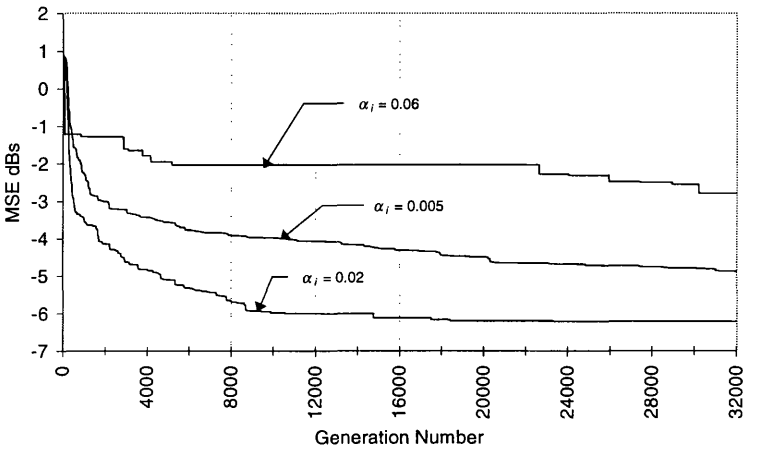

Figure 5.4: Convergence curves showing the effect of rate of immigration while modelling the system (5.6) by an equivalent order adaptive filter.

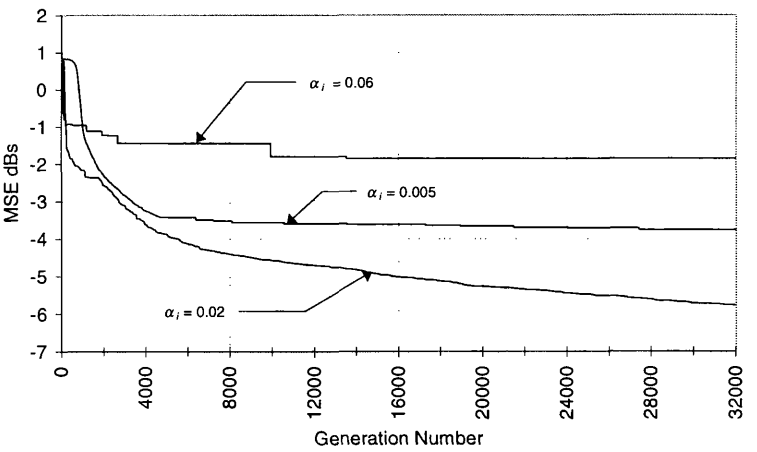

Figure 5.5: Convergence curves showing the effect of rate of immigration while modelling the system (5.4) by an adaptive filter (5.5)

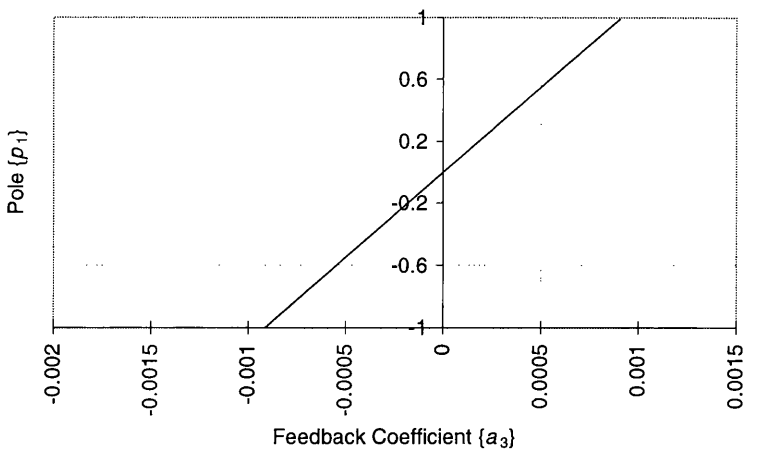

Figure 5.6: Relationship between a pole and a coefficient of a fourth-order IIR system.

5.1.2 Filter Stability

A major problem inherent in the design of an adaptive IIR filter is being able to guarantee filter stability during adaptation. The direct form parameterisation has significant drawbacks, which results from the high sensitivity of its poles to small errors in the feedback coefficients. For example, consider the Figure 5.6 where the relationship between a pole and a coefficient of a 4^{th}-order IIR system is illustrated. This figure interprets that a small change in the a_3 causes a large change in p_1. Therefore, the filters can easily move outside the stable region and provide an unbounded output, which may degrade the convergence performance of the EAs and result in an inaccurate solution or the premature termination of the algorithm can be occurred. To resolve this problem stability monitoring and correction mechanisms must be employed.

There are several methods have been proposed to ensure the stability of adaptive IIR filters [63, 26, 11, 30]. One of the simplest tests of stability is to check after each update of the algorithm that the sum of $|a_m(n)|$ is less than 1 [63]. This method can fail if the coefficient space is large. Moreover, the test will only indicate that a polynomial is unstable when in fact it is not. To ensure stability in large order systems Jury's test can be used [27]. This method can easily determine whether or not a polynomial has minimum phase,

but it does not reveal which poles are unstable. To obtain this information factorisation must be used. Factorising polynomials in higher order IIR systems is computationally expensive.

Although, several techniques have been developed to identify the unstable coefficients, but no such definite method is given to correct these parameters. Correcting the unstable parameters back into the stable region plays an important role in algorithm's convergence. A standard method of replacing these unstable parameters is to use pole projection technique [63]. In pole projection method, the particular unstable pole is reflected back into the stable region in such a way that

$$p_i(t+1) = \frac{1}{p_i(t)} \tag{5.7}$$

where $p_i(t)$ represents the unstable pole, while $p_i(t+1)$ denotes the pole after correction.

In this chapter, we show a new approach in which the feedback sections corresponding to the unstable filters are replaced with the best member of the parent population. This approach is compared with the standard method of correction when designing IIR equalisers to equalise the interference of multi-path channels [70]. In this chapter we present this comparison while modelling the system given in (5.6) by a second-order adaptive system of transfer function

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{5.8}$$

Figure 5.7 presents the learning curves while optimising the parameters of (5.6) using floating-point EAs with the following set of parameters:

- population size $N = 100$
- probability of crossover $\alpha_c = 0.95$
- probability of mutation $\alpha_m = 0.02$
- probability of immigration $\alpha_i = 0.02$

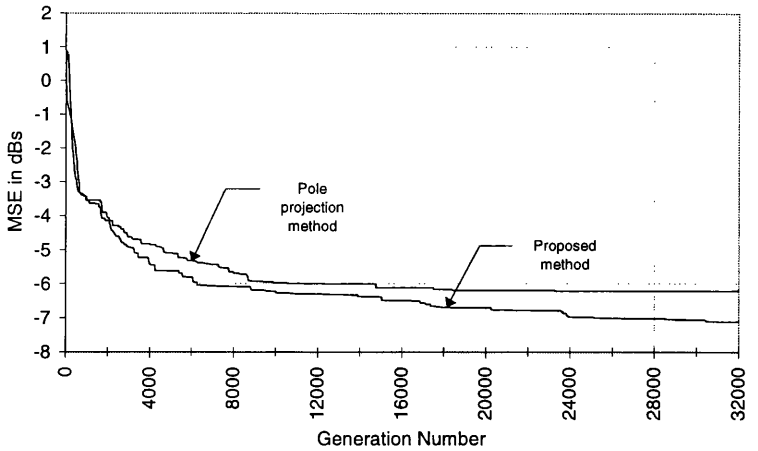

Figure 5.7: Effect of correction mechanisms while modelling the system given in (5.6).

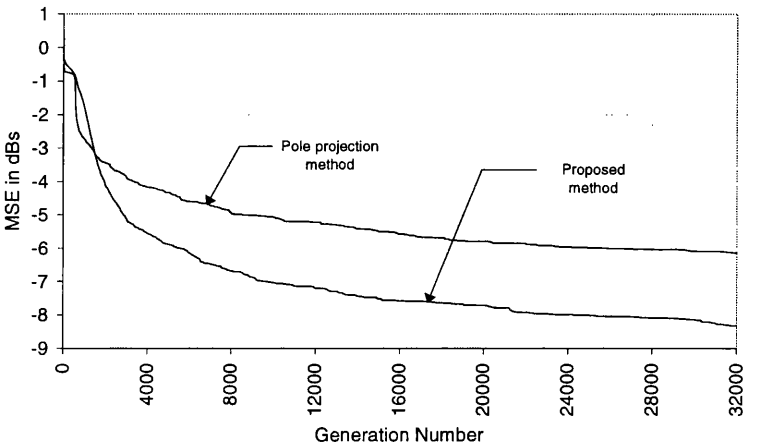

Figure 5.8: Effect of correction mechanisms while modelling the system given in (5.9).

The genetic evolution of adaptive IIR filter using the pole projection method converges slower than the proposed approach. Furthermore, the final MSE values obtained through the proposed mechanism are smaller than the standard method. Another comparison is presented in Figure 5.8. The learning curves shown in this figure are obtained while modelling a fourth-order IIR system of transfer function

$$H(z) = \frac{0.6 - 0.7z^{-1} + 0.3z^{-2} + 0.1z^{-3} + 0.5z^{-4} - 0.4z^{-5}}{(1 - 0.7z^{-1} + 0.6z^{-2} - 0.5z^{-3} + 0.8z^{-4})} \tag{5.9}$$

by an equivalent order adaptive system of transfer function

$$A(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}} \tag{5.10}$$

.

Consequently, several techniques have been proposed along with various filter realisations to simplify stability monitoring trivial. Examples of these realisations are parallel, lattice and cascade. The articles [48, 62, 64] are good references of related works that rely on the design of adaptive IIR filters using various realisations. Although alternative realisations offer simple stability monitoring, these structures introduce additional stationary points into the error surface which may affect the rate of convergence of an adaptive algorithm [63, 3]. For example, consider a MSE surface obtained while modelling a direct form IIR filter,

$$H(z) = \frac{(1 - 0.1z^{-1} + 0.4z^{-2}) + (1 - 0.1z^{-1} - 0.6z^{-2})}{(1 - 0.1z^{-1} - 0.6z^{-2})(1 - 0.1z^{-1} + 0.4z^{-2})} \tag{5.11}$$

by a second-order parallel form adaptive filter

$$A(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} + \frac{1}{1 - d_1 z^{-1} - d_2 z^{-2}} \tag{5.12}$$

For a sake of simplicity a_1 and d_1 are assigned to equal values, $a_1 = d_1 = 0.1$, and a_2 and d_2 are varied to obtain the MSE while modelling (5.11). Figure 5.9 illustrates this error surface plotted against a_2 and d_2. The MSE surface provides two global minima which can be interchanged by swapping the parallel sections given in (5.12). The convergence curves while modelling an IIR system of transfer function

$$H(z) = \frac{0.5 - 0.4z^{-1} + 0.8z^{-2} + 0.7z^{-3} + 0.6z^{-4} - 0.9z^{-5}}{(1 - 0.7z^{-1})(1 + 0.6z^{-1})(1 - 0.65z^{-1})(1 + 0.75z^{-1})} \tag{5.13}$$

by a parallel form

$$A(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} + \frac{c_0 + c_1 z^{-1} + c_2 z^{-2}}{1 - d_1 z^{-1} - d_2 z^{-2}} + \frac{e_0 + e_1 z^{-1}}{1 - f z^{-1}} \tag{5.14}$$

a cascade form

$$A(z) = \left\{\frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}\right\} \left\{\frac{c_0 + c_1 z^{-1} + c_2 z^{-2}}{1 - d_1 z^{-1} - d_2 z^{-2}}\right\} \left\{\frac{e_0 + e_1 z^{-1}}{1 - f z^{-1}}\right\} \tag{5.15}$$

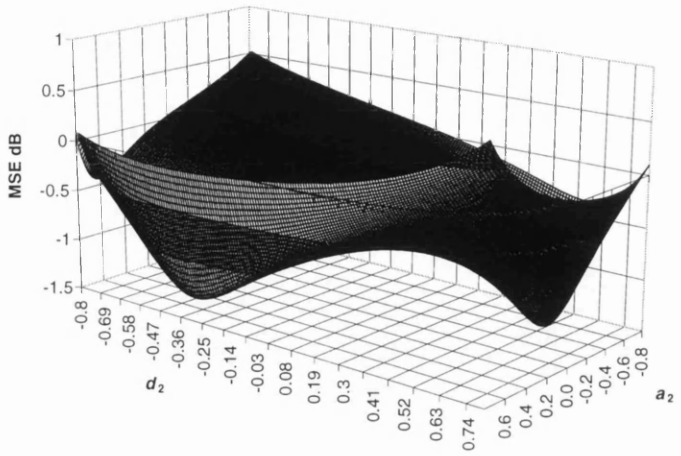

Figure 5.9: A MSE plot showing that parallel sections can be interchanged to obtain the same MSE value.

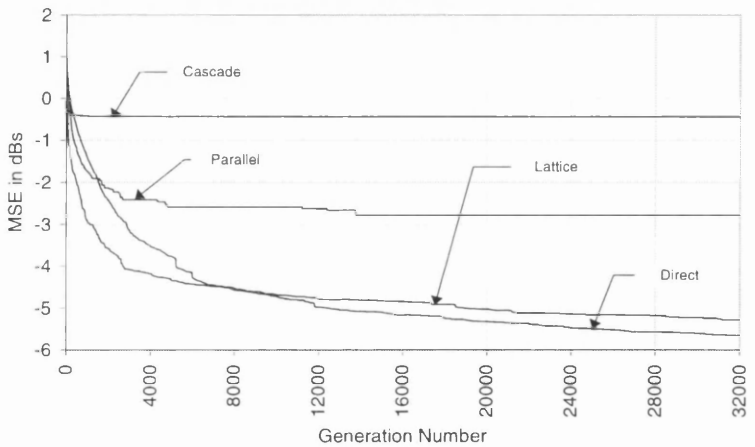

Figure 5.10: Convergence curves while modelling the system given in (5.13) by various filter realisations.

a direct form

$$A(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}} \tag{5.16}$$

and a lattice form with 6 reflection coefficients as described in Appendix F are shown in Figure 5.10. The parallel and cascade forms take much longer time to reach a minimum MSE than lattice and direct form structures. The lattice form reached to a smaller MSE than parallel and cascade forms. The direct form structure, on the other hand, provides a smaller MSE than the lattice form.

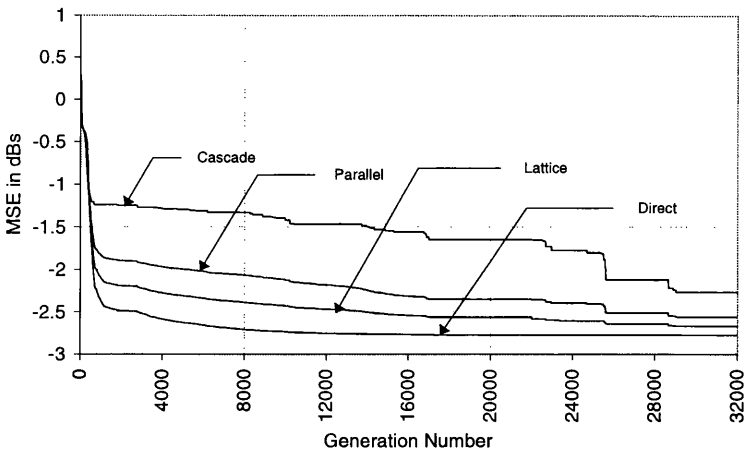

Figure 5.11: Convergence curves while modelling the system given in (5.17) by various filter realisations.

Figure 5.11 presents an another example which compares the convergence performance while modelling a 6^{th}-order IIR system of transfer function

$$H(z) = \frac{0.5 - 0.4z^{-2} - 0.65z^{-4} + 0.26z^{-6}}{1 - 0.77z^{-2} - 0.8498z^{-4} + 0.6486z^{-6}} \tag{5.17}$$

by parallel and cascade sections which are comprising 3 second-order sections, direct form filter with equivalent number of parameters and a lattice filter with 6 reflection coefficients.

The above experiments are carried out using the following set of parameters:

- population size $N = 100$
- probability of crossover $\alpha_c = 0.95$
- probability of mutation $\alpha_m = 0.02$
- probability of immigration $\alpha_i = 0.02$

From the above results it is clear that the rate of optimisation is much dependent on the structure of the filter used. Among various realisations studied, the direct form provides much faster convergence and smaller MSE than other realisations. The crossover operation mainly degrades the convergence of parallel and cascade sections. For example, consider the comparative results as shown in Figures 5.12

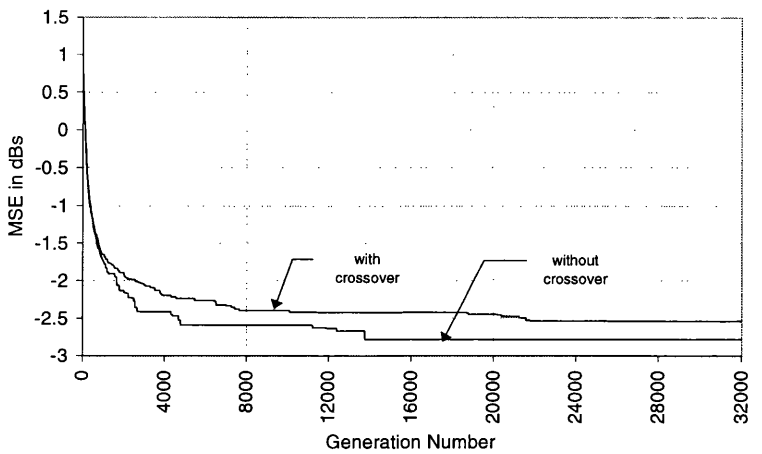

Figure 5.12: Effect of crossover while modelling the system given in (5.13) by a parallel form adaptive filter of transfer function given in (5.14).

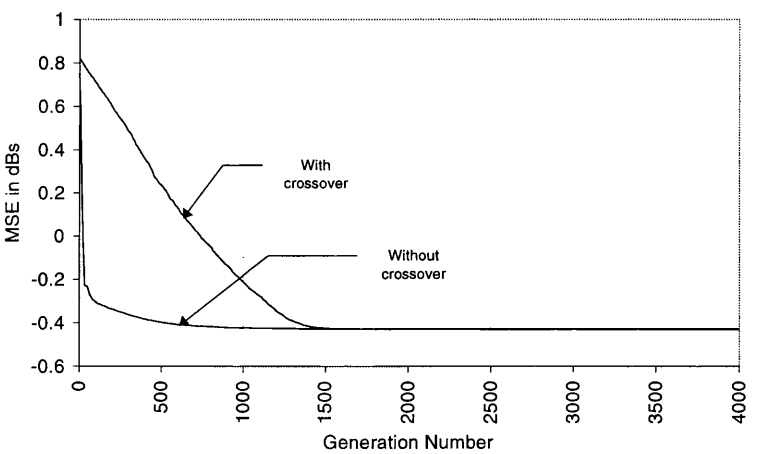

Figure 5.13: Effect of crossover while modelling the system given in (5.13) by a cascade form adaptive filter given in (5.15).

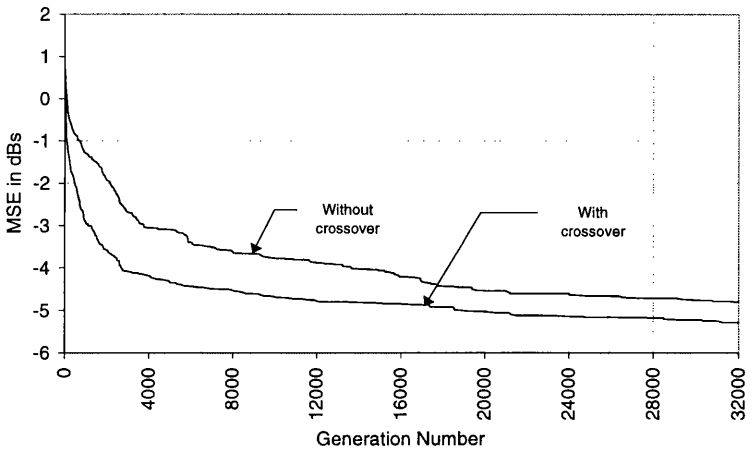

Figure 5.14: Effect of crossover while modelling the system given in (5.13) by a lattice form adaptive filter with 6 reflection coefficients as shown in Appendix F.

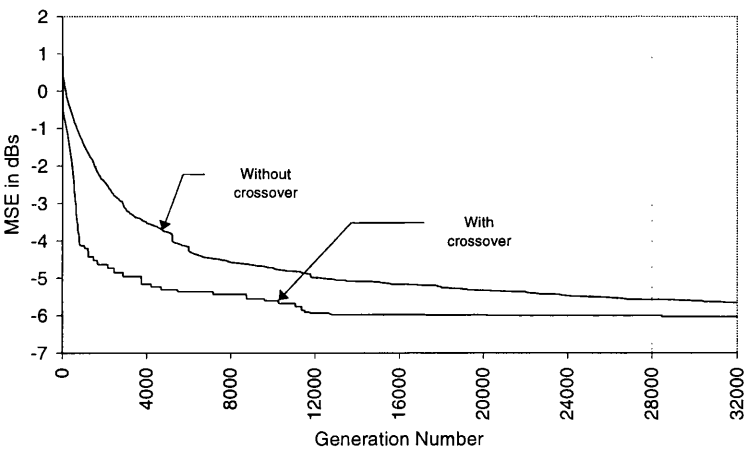

Figure 5.15: Effect of crossover while modelling the system given in (5.13) by a direct form adaptive filter given in (5.16).

- 5.15. These figures compare the learning curves which are obtained without employing crossover operation with those with the crossover operation while modelling the system (5.13) by the adaptive filters (5.14) - (5.16). From these results it is clear that cascade and parallel structures converge faster without crossover. However, the rate of convergence is much slower when compared to direct and lattice forms. Direct and lattice filters converge much faster when crossover operation is employed.

By considering the above issues in mind our aim in this chapter is to design direct form filters with simple stability ensuring techniques.

5.2 Evolving Direct Form IIR Filters

In this section we show how stability monitoring and correction can be easily achieved without employing any of those methods as stated in Section 5.1.2. When evolution is in progress, the coefficients can take any values and the filters can become outside the stable region. Unstable filters provide unbounded output, and if the training samples are too long, the simulation can be prematurely terminated. This is the reason why stability-monitoring techniques are considered when designing evolvable IIR filters.

In this work, we use a radically different approach to ensure filter stability in direct form realisations. We use a termination factor, γ which is compared with the instantaneous estimate of the filter output, $\hat{y}(n)$. If the magnitude of the output-estimate exceeds the termination factor, the filter can be discarded or can be assigned to a lower fitness value so that a filter with a higher fitness can replace this vacancy when the new population is to be selected. For example, consider the learning curves shown in Figure 5.16. This figure compares the convergence properties of the new approach against standard method of stability monitoring while modelling a fourth-order system of transfer function

$$H(z) = \frac{0.1084 + 0.5419z^{-1} + 1.0837z^{-2} + 1.0837z^{-3} + 0.5419z^{-4} + 0.1084z^{-5}}{(1 - 0.75z^{-1})(1 - 0.8z^{-1})(1 + 0.65z^{-1})(1 + 0.82z^{-1})} \tag{5.18}$$

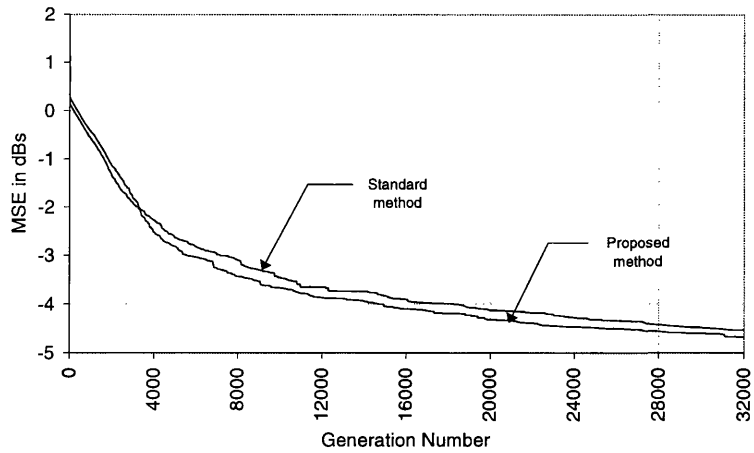

Figure 5.16: Learning curves comparing the convergence properties of the new stability monitoring approach against the standard approach as discussed in Section 5.1.2.

by an equivalent order adaptive filter of transfer function

$$A(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}} \tag{5.19}$$

The reason for choosing the above system is to show the convergence properties of a system that has poles very close to the unit circle. The given system has poles close to the unit circle - therefore stability monitoring is of paramount importance. As shown in this figure, both methods give the same amount of convergence performance, but they require different computational costs. The proposed method requires much less computation than the standard method. However, a major problem encountered in proposed technique is that it does not provide direct indication to filter instability but implicitly indicate whether or not coefficients have minimum phase. Figures 5.17 and 5.18 also give these comparisons which are obtained when modelling the following systems

$$H(z) = \frac{1.0 - 0.9z^{-1} + 0.81z^{-2} - 0.729z^{-3}}{1.0 - 0.2314z^{-1} + 0.43174z^{-2} - 0.340434z^{-3} + 0.5184z^{-4}} \tag{5.20}$$

and

$$H(z) = \frac{0.0154 + 0.0462z^{-1} + 0.0462z^{-2} + 0.0154z^{-3}}{1.0 - 1.99z^{-1} + 1.572z^{-2} - 0.4583z^{-3}} \tag{5.21}$$

by 4^{th}-order and 3^{rd}-order adaptive filters respectively. The given systems have poles very close to the unit circle. The system given by (5.20) has poles

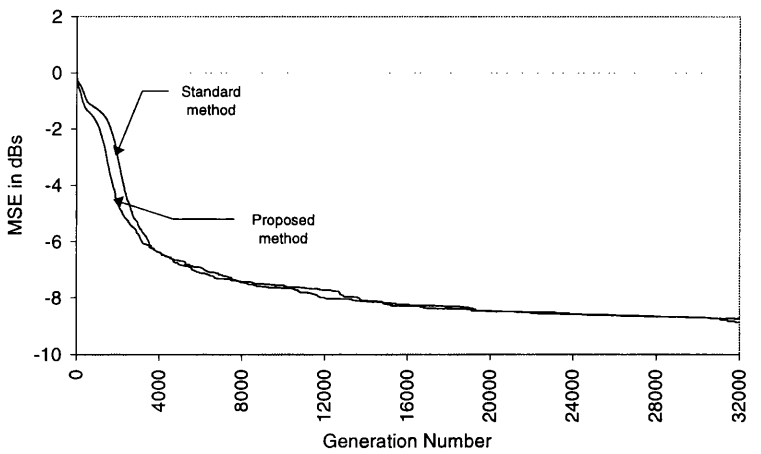

Figure 5.17: Learning curves comparing the convergence properties of the new stability monitoring approach against the standard approach as discussed in Section 5.1.2.

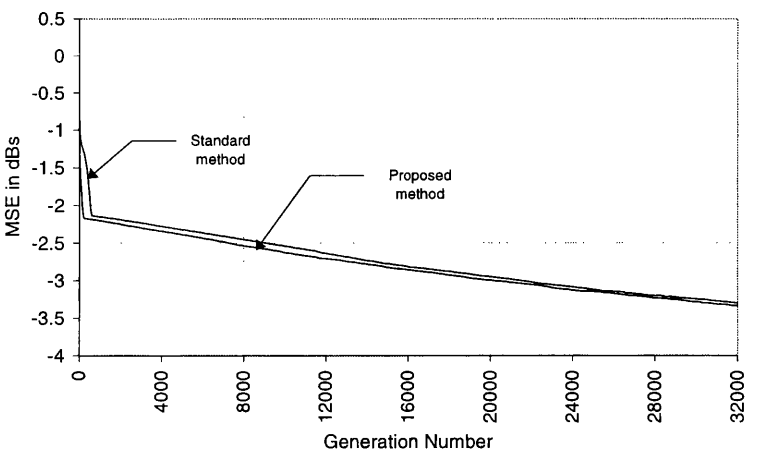

Figure 5.18: Learning curves comparing the convergence properties of the new stability monitoring approach against the standard approach as discussed in Section 5.1.2.

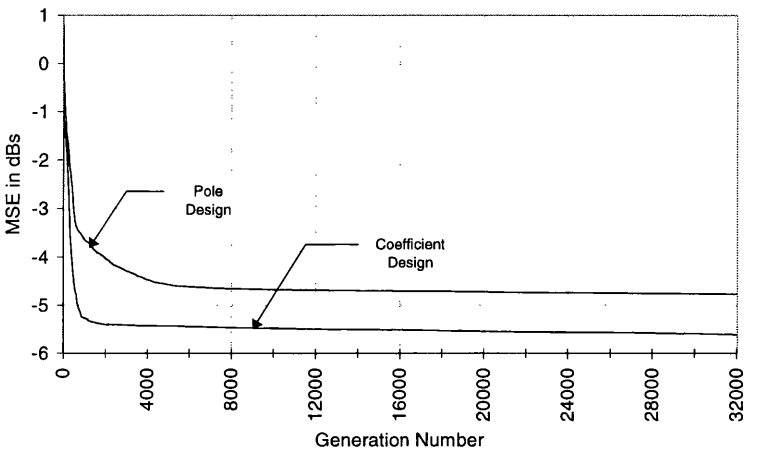

Figure 5.19: Learning curves comparing the convergence properties of the coefficient design method against pole design method.

at $p_{1,2} = 0.5\angle \pm 45$, $p_{3,4} = 0.75\angle \pm 120$, and zeros at $z_{1,2} = 0.9\angle \pm 90$, $z_3 = 0.9$, $z_4 = 0$ while the system (5.21) has its poles at $p_{1,2} = 0.833\angle \pm 45$, $p_3 = 0.6605$, and zeros at $z_{1,2} = 0.9\angle \pm 90$, $z_3 = 0.9$, $z_4 = 0$.

Finally, we compare the convergence properties of the proposed coefficient design method against pole design method using the same signals and systems as used in the previous chapter. First experiment is conducted to show the convergence curves while modelling the direct system (4.29). Genetic algorithms with the following set of parameters:

- population size $N = 100$
- probability of crossover $\alpha_c = 0.95$
- probability of mutation $\alpha_m = 0.02$
- probability of immigration $\alpha_i = 0.02$

is used to obtain filter estimates. The adaptive filter used in this experiment has 5 feedforward coefficients and 3 feedback coefficients as the original plant. Figure 5.19 shows the convergence curves. Figure 5.20 compares this performance while modelling the inverse system used in the previous chapter.

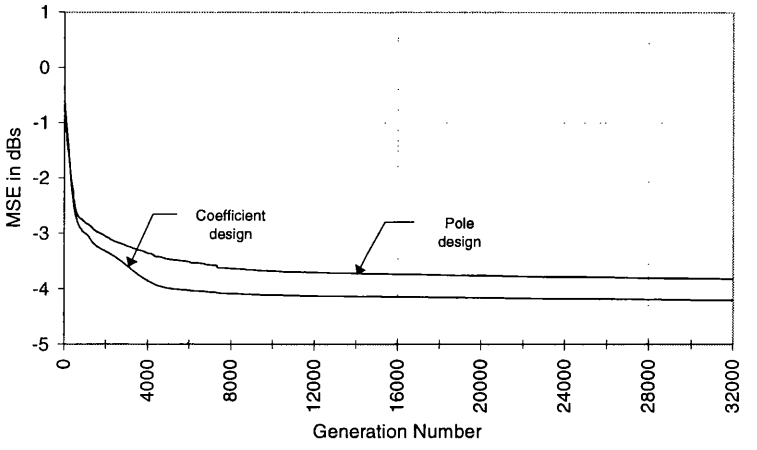

Figure 5.20: Learning curves comparing the convergence properties of the coefficient design method against pole design method.

5.3 Summary and Discussion

This chapter has provided the design techniques of direct form parameterisation using simply stability monitoring techniques. Direct forms parameterisation has given faster rates of convergence than other realisations such as parallel, cascade and lattice forms. A major problem inherent in designing direct form filters using floating-point EAs is that it often fails to converge to an optimum solution due to premature convergence. This is mainly due to small variance of mutation and small population sizes. Premature convergence can be avoided by using one of the remedies described in Section 5.1.1. However, these methodologies often fail if the error surface has severe multimodal performance. Higher mutation rate can reduce the severity of premature convergence, but it can degrade the convergence performance. To resolve this problem, this work has provided a new operator called immigrants, which introduces new chromosomes that are obtained by perturbing each element of parent chromosomes with higher variations. Number of immigrants introduced into the population play an important role in EA's convergence. Moderate values gave better performance which is clearly seen from Figures 5.4 and 5.5.

The way of correcting unstable coefficients back into the stable region played

an important role in EA's convergence. This chapter has shown a new methodology for correcting unstable parameters, which in turn improves the rate of convergence of floating-point EAs. Pole projection method that has been used as a standard correction mechanism was compared with the proposed method. Pole projection method requires an exact knowledge of unstable coefficient so that it can be projected back into the stable region. To get the information pole polynomials must be factored. Factorising polynomials is computationally expensive. Proposed method does not require such information and therefore computationally inexpensive. Another important observation has been made from our simulations is that proposed method improves the convergence of EAs. This new approach provides faster rate of convergence when compared to pole projection method.

Even though the proposed method of correction is computationally inexpensive, but it needs stability monitoring techniques which require high computation. To simplify stability monitoring, we have provided a new methodology, which uses a termination factor that determines the filter stability implicitly. This method has been tested while modelling various IIR systems that gives good performance as standard approaches.

Finally, our conclusion is that coefficient design method always converges faster than pole design method. Coefficient design gives better fine-tuning capability and provides final MSE values which are smaller than pole design approach.

Chapter 6

Hybrid Adaptive Approach and System Modelling for Adaptation

In the previous chapters, the design techniques of IIR filters towards global optimality have been described. A general conclusion is that evolutionary techniques provide a global search capability but is poor in fine-tuning, which has been a major drawback. Moreover, evolutionary algorithms require much longer time than classical methods and therefore are unsuitable for applications requiring tracking time varying changes. For example, channel equalisation of multipath channels strictly requires an efficient adaptive algorithm to track the time varying characteristics of the channel environment [82, 6, 76, 28].

This chapter develops design techniques for adaptive IIR digital filters using a hybrid approach. This work uses EA as a major search tool to find global regions of IIR error surfaces and employs a gradient-based algorithm where appropriate for fine-tuning the coefficients. When a global region has been found, the gradient algorithm can be switched on to track any time-variant changes without the help of an evolutionary algorithm. This method resolves the problems of achieving fine-tuning and tracking time varying performance by evolutionary approaches.

The rest of this chapter is organised as follows: In Section 6.1, fine-tuning and hybrid methodologies are described. This section describes how the parameters of time-invariant systems can be fine-tuned using the new method. Applications and validation to various system modelling are illustrated. Section 6.2 briefly describes the modelling of time-invariant and time-variant systems. Simulation results illustrating convergence performance are shown. In Section 6.3 provides modelling of time-variant systems is shown and simulation results showing the convergence performance illustrated. Section 6.4 presents adaptive inverse modelling in nonstationary environment. Finally, Section 6.5 provides a brief summary and discussion of this chapter.

6.1 Fine-tuning and Hybrid Methodology

In this section we describe how the parameters of time-invariant systems can be fine-tuned using hybrid methodology. Evolutionary algorithm and the normalised RPE algorithm are chosen for this hybrid search approach. A set of training data of w samples is used to train the adaptive filters. The proposed floating-point EA with a fixed population size N is employed and the best member (an individual having the lowest cost) of the population is subjected to fine-tuning via normalised RPE algorithm, which is discussed in Chapter 2. The same training set and the same number of training data are used to update the normalised RPE algorithm. Adaptation through normalised RPE algorithm can fail if the path contains multiple local optima or the coefficients are updated outside the stable region. Filter can be checked for stability using the technique discussed in the previous chapter. At each generation, the coefficients obtained via normalised RPE algorithm are used to calculate the MSE value of the instantaneous error and compared with the best member of the EA's population. The winner of this competition is used to update the normalised RPE algorithm and replaces a member of the parent population before the next generation to be evolved. This cycle is repeated until the MSE of the normalised RPE reach

to a steady state value. A flowchart representing this cycle is shown in Figure 6.1.

6.2 Applications and Validation to System Modelling

In this chapter, we consider two types of system modelling problems, which include the modelling of systems, whose parameters are

1. Time-invariant
2. Time-variant

We use the following notation

$$g = \{b_0, b_1, \cdots, b_{M-1}, a_1, a_2, \cdots, a_N\} \tag{6.1}$$

to represent time-invariant systems and use

$$g(n) = \{b_0(n), b_1(n), \cdots, b_{M-1}(n), a_1(n), a_2(n), \cdots, a_N(n)\} \tag{6.2}$$

to represent time-variant systems. The index M and N used in (6.1) and (6.2) represent the number of numerator and denominator coefficients respectively. In our simulations, we assume that the parameters defined (6.2) change with time with a frequency α_f which is considerably slower than that of adaptation rate. For example, let us assume that the parameter b_1 is varied in such a manner

$$b_1(n) = b_1 + \{1 - \exp(-\alpha_f n)\} \tag{6.3}$$

where b_1 is the original value and n denotes the time. Our aim in this chapter is to model direct and inverse of such systems in the direct form adaptive IIR filters. We further assume that models can be trained by a set of training signals.

A series of simulations is carried out to show the fine-tuning capability of the proposed approach. The system described by (6.1) is used in all simulations

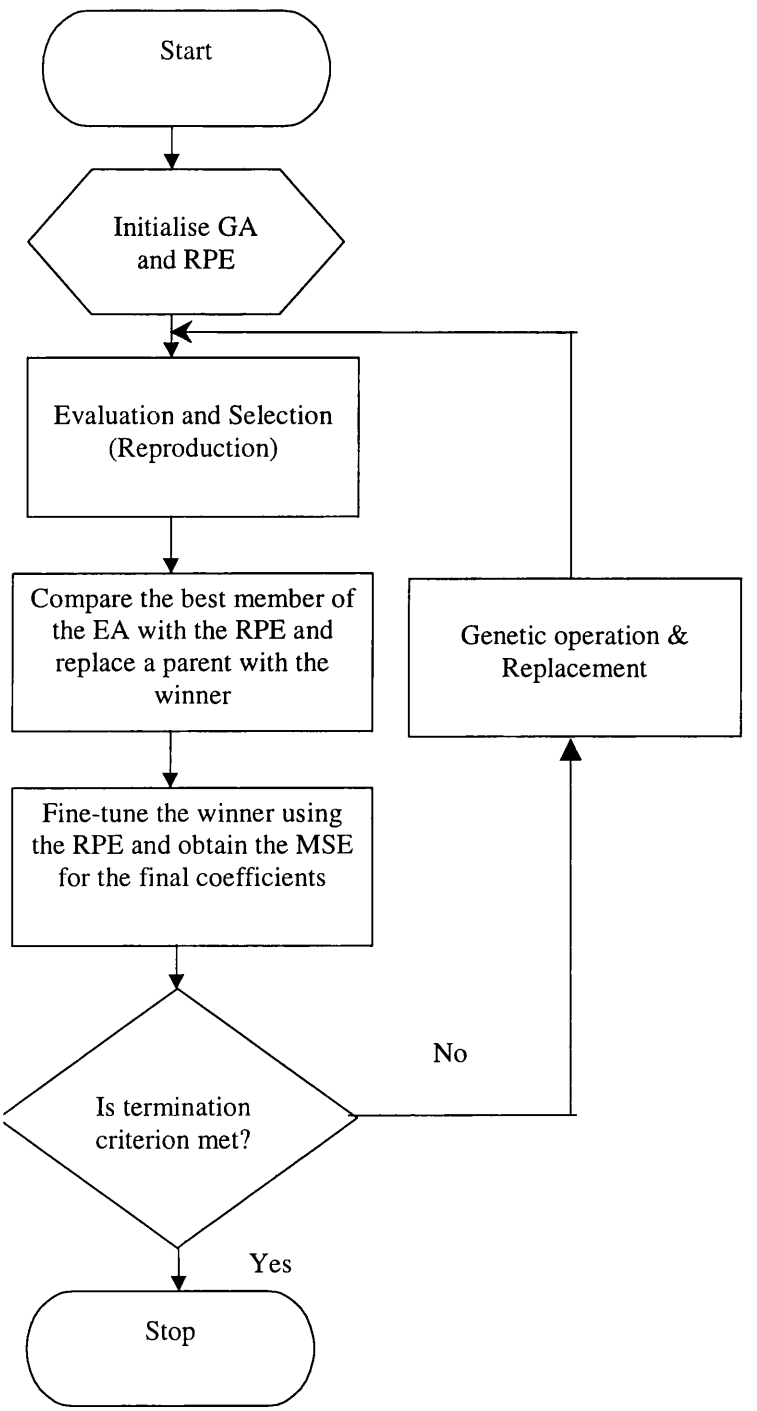

Figure 6.1: Flowchart illustrating the hybrid approach.

of this section, and the input signal $s(n)$ applied to this system is a white random process with a normal distribution and unit variance. For the sake of simplicity, the experiments presented in this chapter are carried out in noise free environments. An estimate of the MSE at each generation is obtained by averaging

$$J = \frac{1}{w} \sum_{i=0}^{w-1} |e(i)|^2 \tag{6.4}$$

over 15 independent computer runs, where w represents the number of training signals. The number of training signals and the population size used in all experiments are 100 and 50 respectively. From the knowledge of our previous experiments carried out in the previous chapter, we chose the following values as genetic parameters:

- Probability of crossover $\alpha_c = 0.95$
- Probability of mutation $\alpha_m = 0.02$
- Probability of immigration $\alpha_i = 0.01$

The step size μ of normalised RPE is set equal to 0.001 unless otherwise specified. The reasons for chosen this value have been discussed in Chapter 2.

To make a clear conclusion on the convergence performance of the proposed approach, we show three kinds of direct system modelling of (6.1) such as *exact-modelling* (order of an adaptive filter and the system being modelled are equal), *over-modelling* (order of an adaptive filter is greater than the system being modelled) and *poles close to the unit circle* [64]. In all three cases the adaptive filter has the general form:

$$A(z) = \frac{c_0 + c_1 z^{-1} + \cdots + c_{M-1} z^{-(Q-1)}}{1 - d_1 z^{-1} - \cdots - d_N z^{-P}} \tag{6.5}$$

where Q and P represents the numerator and denominator coefficients respectively.

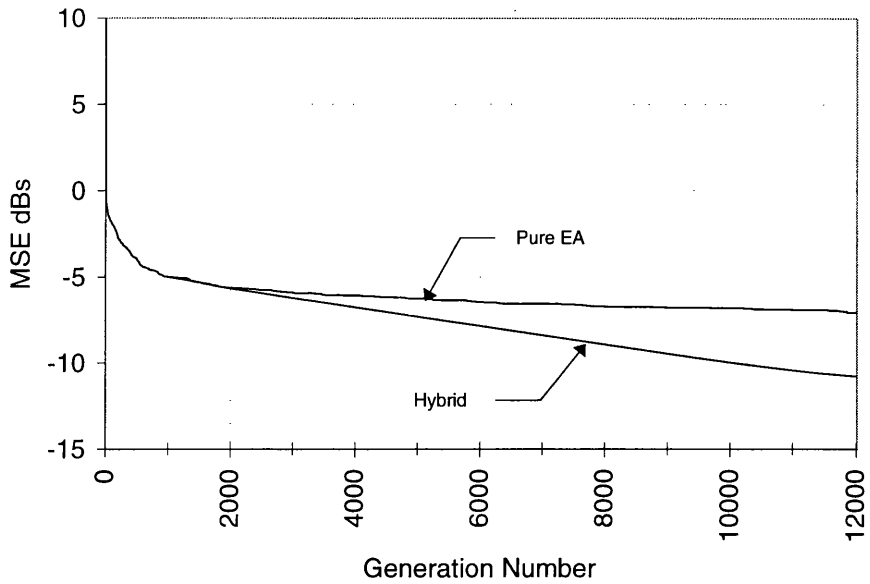

Figure 6.2: Exact-modelling with fine-tuning performance. MSE learning curves of the pure EA and the hybrid approach

6.2.1 Exact Modelling

In this case, g is a fourth-order system with distinct poles located well inside the unit circle. The coefficients of g are

$$\begin{cases} b_0 = 1.0,\ b_1 = -0.9,\ b_2 = 0.81,\ b_3 = -0.729, \\ a_1 = -0.04,\ a_2 = -0.2775,\ a_3 = 0.21012,\ a_4 = -0.14 \end{cases} \tag{6.6}$$

which give poles at $p_{1,2} = 0.5\angle \pm 45$, $p_{3,4} = 0.75\angle \pm 120$, and zeros at $z_{1,2} = 0.9\angle \pm 90$, $z_3 = 0.9$, $z_4 = 0$. Figure 6.2 shows the MSE learning curves of the pure EA and the hybrid algorithm. Evolutionary algorithm finds the shortest root to normalised RPE algorithm. The global region is found within 1000 generations and normalised RPE algorithm takes another 110000 generations to fine-tune the coefficients. Figure 6.3 shows the trajectories of a_1 and b_0. From these results it is clear that the above two coefficients reach to its original values within 500 - 1000 generations. The convergence performance of normalised RPE algorithm itself is attempted to show in this chapter, but in all attempts the results are unsatisfactory due to multiple local minima and potential instability of direct form adaptive IIR filters. Therefore, they are not presented.

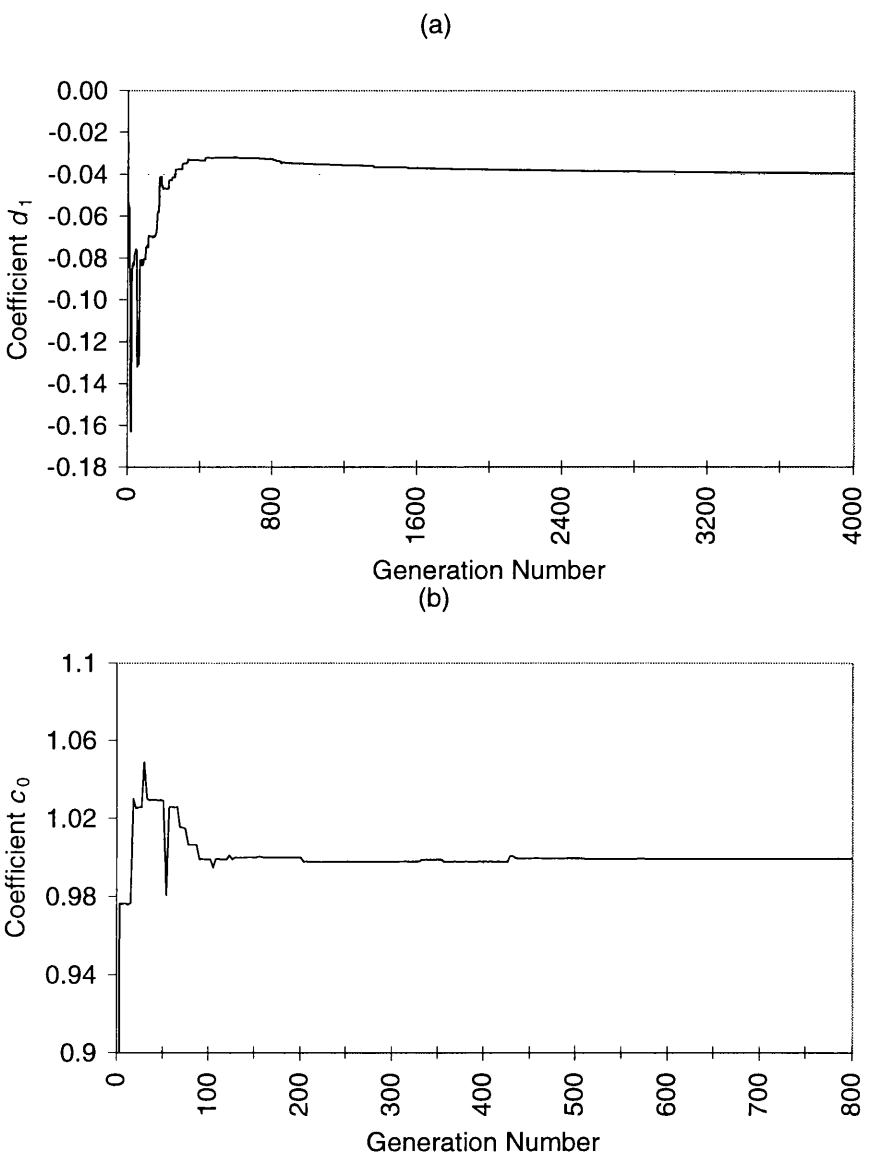

Figure 6.3: Exact-modelling with fine-tuning performance. Trajectories of (a) d_1 (b) c_0

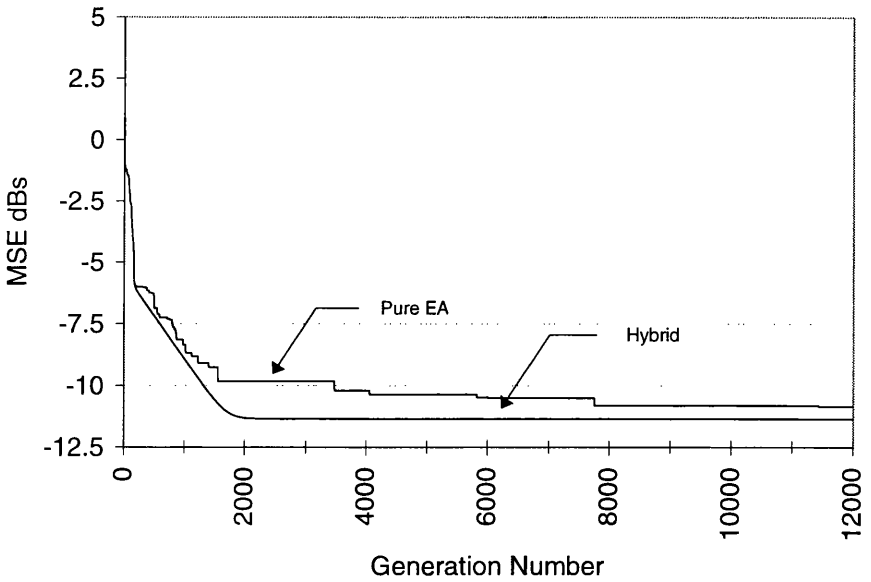

Figure 6.4: Over-modelling with fine-tuning performance. MSE learning curves of the pure EA and the hybrid approach

6.2.2 Overmodelling

In this simulation, g is a second-order system of parameters

$$\begin{cases} b_0 = 1.0,\ b_1 = -0.9,\ a_1 = 0.71,\ a_2 = -0.25 \end{cases} \tag{6.7}$$

with distinct poles at $p_{1,2} = 0.5\angle \pm 45$ and zeros at $z_1 = 0.9$, $z_2 = 0$. The adaptive filter has $P = 3$ and $Q = 2$, i.e. $P > N$. Figure 6.4 shows the MSE learning curves of pure EA and the hybrid algorithm. The EA finds the global region within 200 generations and normalised RPE takes another 1800 generations for fine-tuning. Figure 6.5 shows the trajectories of d_1 and c_0. The coefficient trajectories clearly shows that d_1 reaches its original value within 200 generations while c_0 takes 100 generations.

6.2.3 Poles Close to the Unit Circle

A fourth-order system is used as in the exact-modelling case, except the poles are at $p_{1,2} = 0.8\angle \pm 45$ and $p_{3,4} = 0.9\angle \pm 120$. The coefficients corresponding to

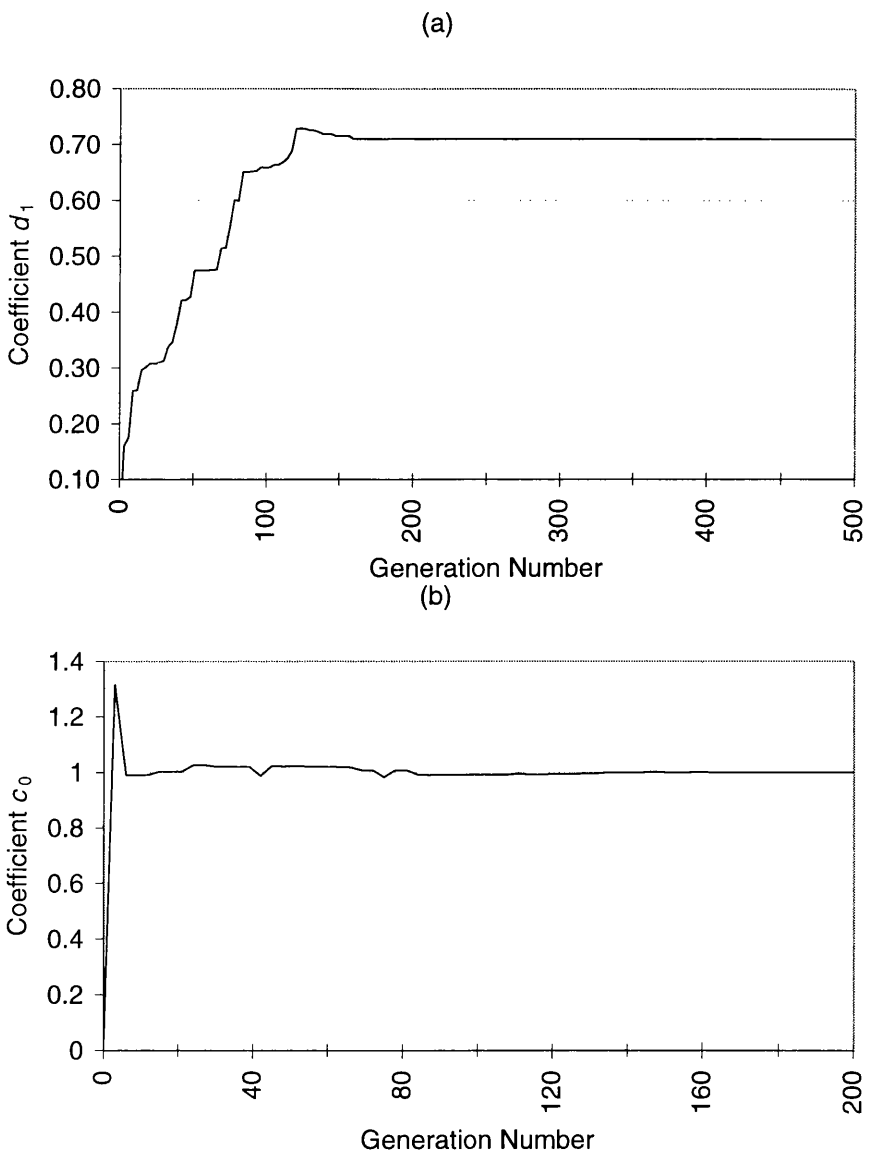

Figure 6.5: Over-modelling with fine-tuning performance. Trajectories of (a) d_1 (b) c_0

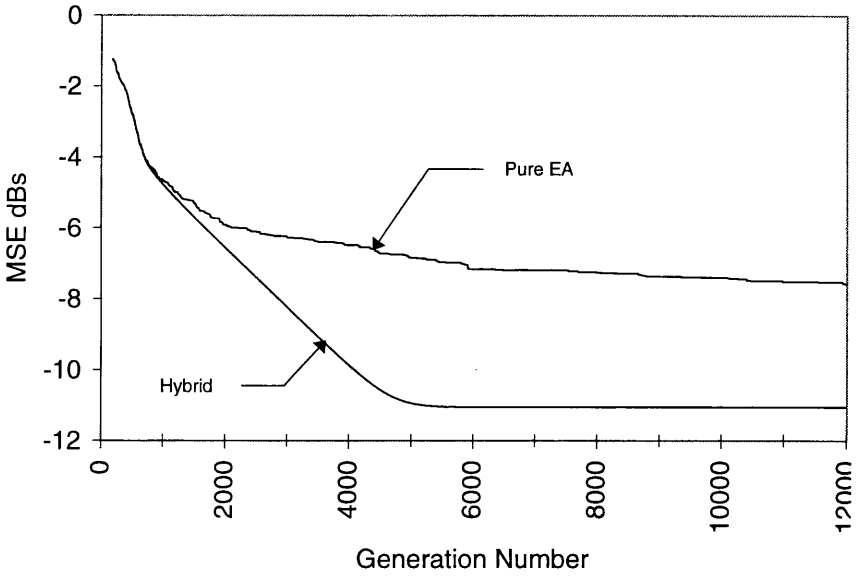

Figure 6.6: Modelling with poles close to the unit circle to show fine-tuning performance. MSE learning curves of the pure EA and the hybrid approach

these poles are

$$\begin{cases} b_0 = 1.0,\ b_1 = -0.9,\ b_2 = 0.81,\ b_3 = -0.729, \\ a_1 = 0.2314,\ a_2 = -0.43174,\ a_3 = 0.340434,\ a_4 = -0.5184 \end{cases} \tag{6.8}$$

Figure 6.6 illustrates the learning curves while Figure 6.7 shows the coefficients trajectories. In this case the EA takes 1000 generations to find the global region and normalised RPE takes 5000 generations to fine-tune the coefficients. However, the coefficients d_1 and c_o are converged to its original values within 500-1000 generations.

6.3 Modelling in a Nonstationary Environment

This section applies the modelling techniques to time-variant systems. Time variant systems have parameters have parameters that can be varied with time. Therefore modelling of such systems cannot be achieved with an offline process such as EAs. In this case, the EAs can only be used to find a better root for an algorithm that track time varying performance. An algorithm such as normalised RPE can be used to achieve the adaptation of time varying changes.

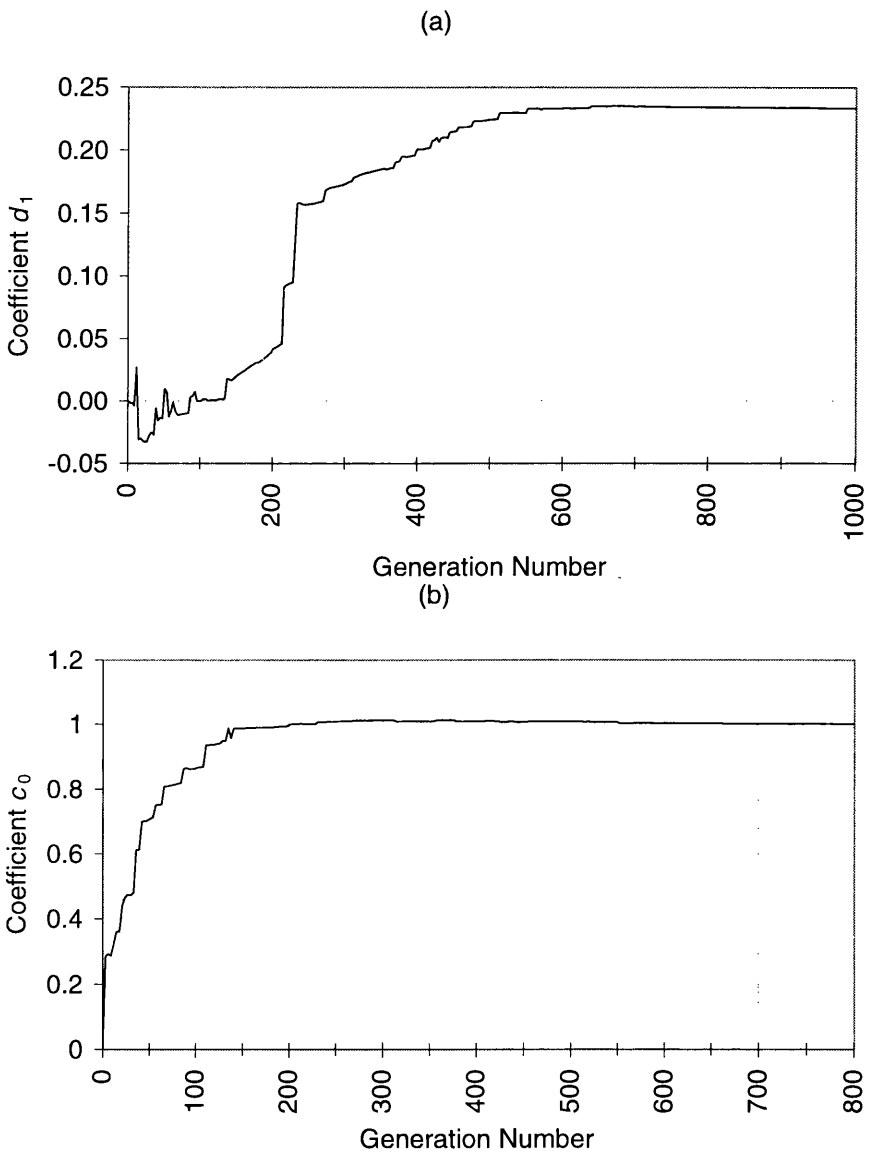

Figure 6.7: Modelling with poles close to the unit circle with fine-tuning performance. Trajectories of (a) d_1 (b) c_0

We run the hybrid algorithm to obtain a best solution for a particular training set and then employ normalised RPE algorithm to adapt the time varying parameters.

To illustrate the convergence performance of the proposed approach, we consider a variety of direct and inverse modelling of $g(n)$. The aim of these experiments is to show the capability of the proposed method when modelling systems in nonstationary environments.

In this case we consider the same systems used in Section 6.2, except that the parameters are assumed to be time-variant. Exact modelling is considered unless otherwise specified. The genetic parameters and population size are chosen as those used in the previous experiments. Firstly, we consider a fourth-order IIR system of parameters

$$\begin{cases} b_0(n) = 1.0 + 0.5\{1 - \exp(-0.001n)\},\ b_1 = -0.9, \\ b_2(n) = 0.81 + 0.4\{1 - \exp(-0.001n)\}, \\ b_3(n) = -0.729 + 0.25\{1 - \exp(-0.01n)\}, \\ a_1 = -0.04,\ a_2 = -0.2775,\ a_3 = 0.21012,\ a_4 = -0.14 \end{cases} \tag{6.9}$$

where the parameter $b_0(n)$, $b_2(n)$ and $b_3(n)$ of (6.9) are varied exponentially at a rate 0.001, 0.001 and 0.01 respectively. Other parameters such as $\{b_1, a_1, \cdots, a_4\}$ are assumed to be time-invariant. Initially, a training set of 10000 samples is generated with these parameters. Among 10000 samples generated, only 100 are used to obtain the best member via offline. The hybrid algorithm is employed and run until the normalised RPE reach to a steady state value. The best parameters obtained through offline process are used as the initial parameters for the normalised RPE to track time varying changes. Figure 6.8 shows the MSE leaning curve of hybrid algorithm while modelling the system using an equivalent order adaptive filter. The hybrid algorithm gives the best solution within 1000 generations. Once the best solution for the particular training set is found, the normalised RPE algorithm is allowed to run to track time varying changes. Figure 6.9 shows how the coefficients $c_2(n)$ and $c_3(n)$ are converged during adaptation via normalised RPE. normalised RPE algorithm

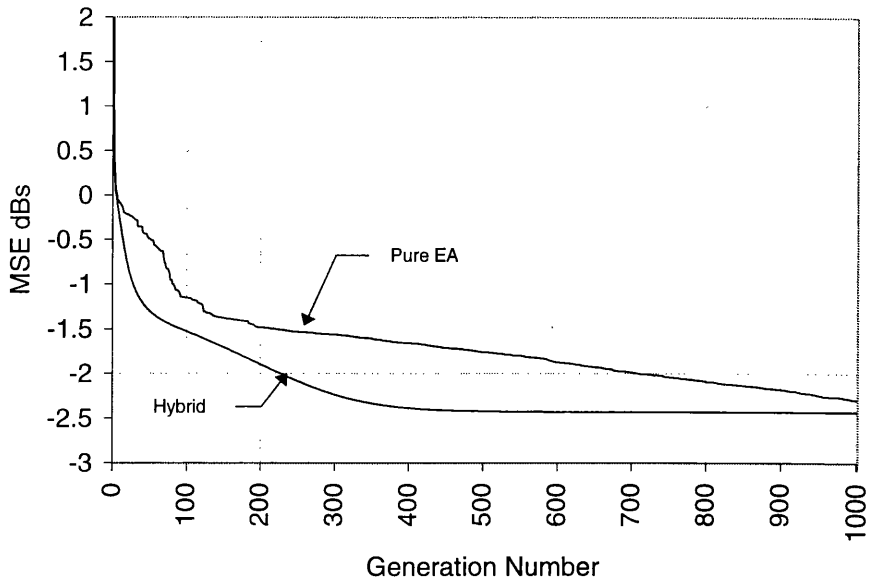

Figure 6.8: Adaptation with model (6.9) using an equivalent order adaptive filter. MSE learning curves of pure EA and hybrid approach.

could track the time varying changes, however, the adaptive filter coefficients and the corresponding original values are not exactly matched to each other. The reason for this poor performance is that the model parameters are varied with a higher rate compared to that of adaptation, i.e. $\mu = 0.001$.

Next, we consider the modelling of the same system, except the parameters $\{b_0(n), b_2(n), b_3(n)\}$ are varied much slower than that of (6.9). i.e. $b_0(n)$, $b_2(n)$ and $b_3(n)$ are varied in such a way that

$$\begin{cases} b_0(n) = 1 + 0.5\{1 - \exp(-0.0001n)\}, \\ b_2(n) = 0.81 + 0.4\{1 - \exp(-0.0001n)\}, \\ b_3(n) = -0.729 + 0.25\{1 - \exp(-0.0001n)\} \end{cases} \tag{6.10}$$

In this case the normalised RPE algorithm easily tracks these variations as new data are being captured. Figure 6.10 clearly confirms the statement stated in previous paragraph by illustrating the convergence of $c_2(n)$ and $c_3(n)$ against iteration number. It is easily seen that the original coefficient values and the coefficients obtained through normalised RPE are closer to each other.

Another experiment is carried out to show the convergence performance when the adaptation rate is $\mu = 0.01$. Figure 6.11 illustrates the coefficients

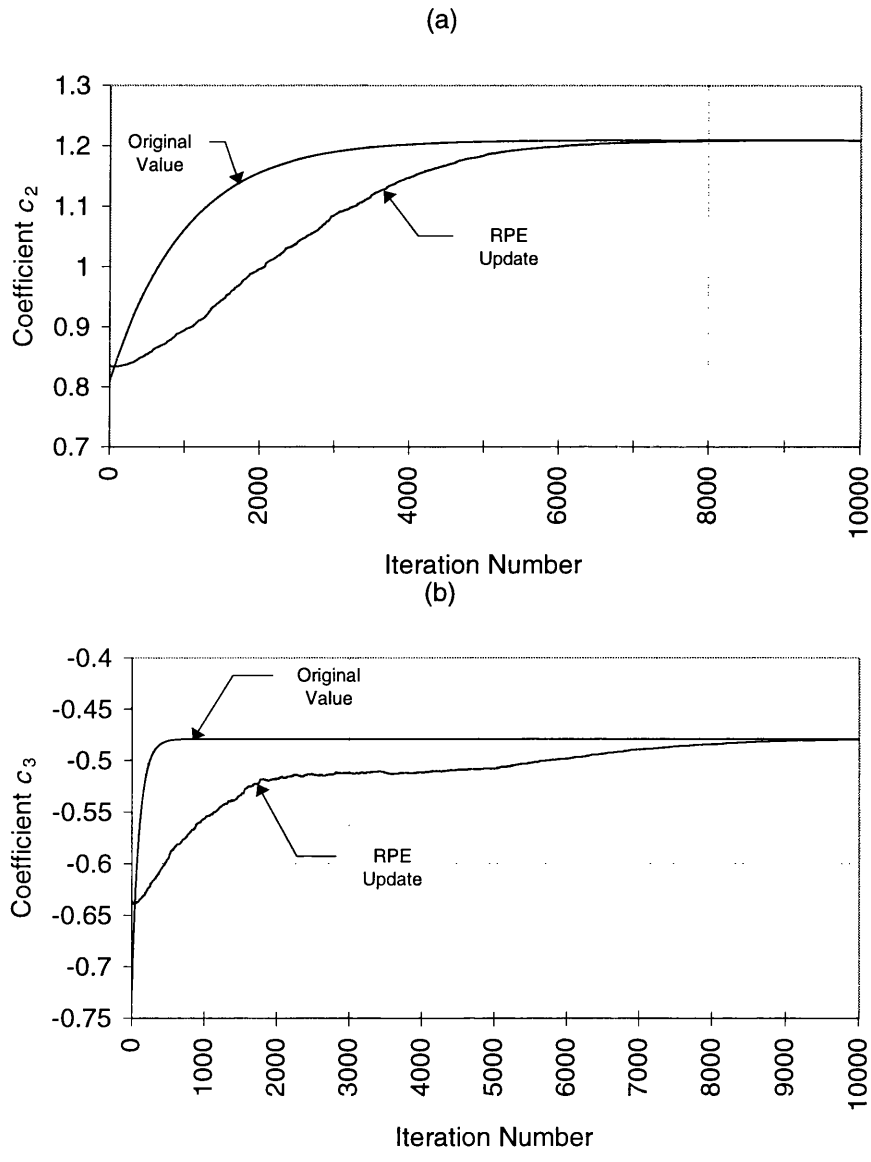

Figure 6.9: Adaptive modelling of (6.9) using an equivalent order adaptive filter. Trajectories of (a) $c_2(n)$ (b) $c_3(n)$

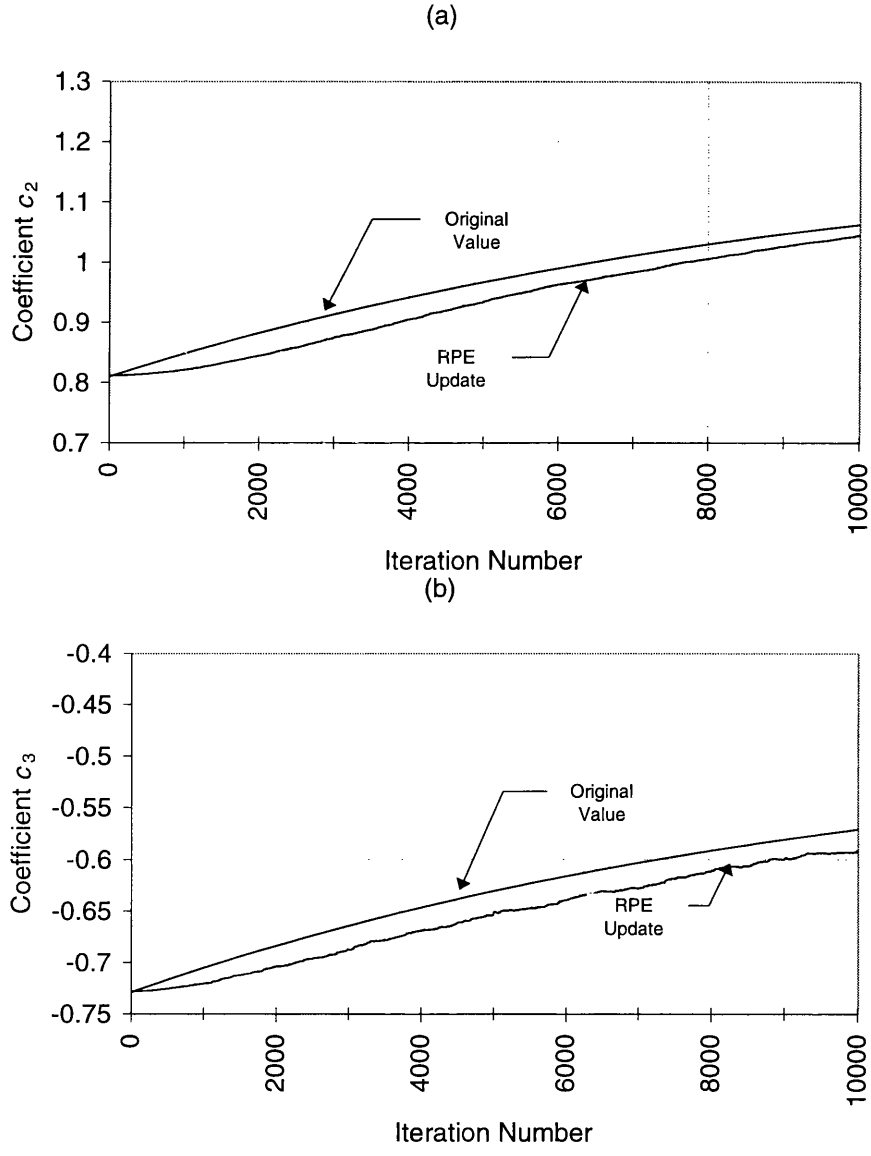

Figure 6.10: Adaptive modelling of (6.10) using an equivalent order adaptive filter. Trajectories of (a) $c_2(n)$ (b) $c_3(n)$

trajectories while modelling the (6.9) using an equivalent order adaptive filter with the new adaptation rate. The convergence curves of c_2 clearly shows the improvement - in this case adaptation rate is higher than the parameter variation.

Therefore, an important conclusion, which can be made from above discussion, is that normalised RPE algorithm can give a better convergence performance if the rate of channel variations is smaller than that of adaptation rate.

In the second modelling example, we consider an over-modelling problem with a second-order system of parameters

$$\begin{cases} b_0(n) = 1.0 + 0.5\{1 - \exp(-0.001n)\},\ b_1 = -0.9, \\ a_i(n) = 0.71 + 0.25\{1 - \exp(-0.001n)\},\ a_2 = -0.25 \end{cases} \tag{6.11}$$

In this case $b_0(n)$ and $a_1(n)$ are assumed to be time-invariant. Also the adaptation rate and the rate of parameter variations were equal. Figure 6.12 shows the MSE learning curves of hybrid and pure EA while offline adaptation. Figure 6.13 shows the trajectories of the time varying coefficients obtained during adaptation. The original values and the normalised RPE updates are not closer to each other. Figure 6.14 illustrates the coefficients trajectories when the rate of parameter variations are set to $\alpha_f = 0.0001$, which is smaller than the adaptation rate $\mu = 0.001$. i.e.

$$\begin{cases} b_0(n) = 1.0 + 0.5\{1 - \exp(-0.0001n)\},\ b_1 = -0.9, \\ a_1(n) = 0.71 + 0.25\{1 - \exp(-0.0001n)\},\ a_2 = -0.25 \end{cases} \tag{6.12}$$

In our third example we consider a system with the poles close to the unit circle. In this case exact-modelling is considered. Firstly, we assume these parameters are

$$\begin{cases} b_0(n) = 1.0 - 0.5\{1 - \exp(-0.001n)\}, \\ b_1(n) = -0.9 + 0.25\{1 - \exp(-0.01n)\}, \\ b_2(n) = 0.81,\ b_3(n) = -0.729 + 0.5\{1 - \exp(-0.001n)\}, \\ a_1(n) = 0.2314 + 0.05\{1 - \exp(-0.001n)\},\ a_2 = -0.43174, \\ a_3 = 0.340434,\ a_4 = -0.5184 \end{cases} \tag{6.13}$$

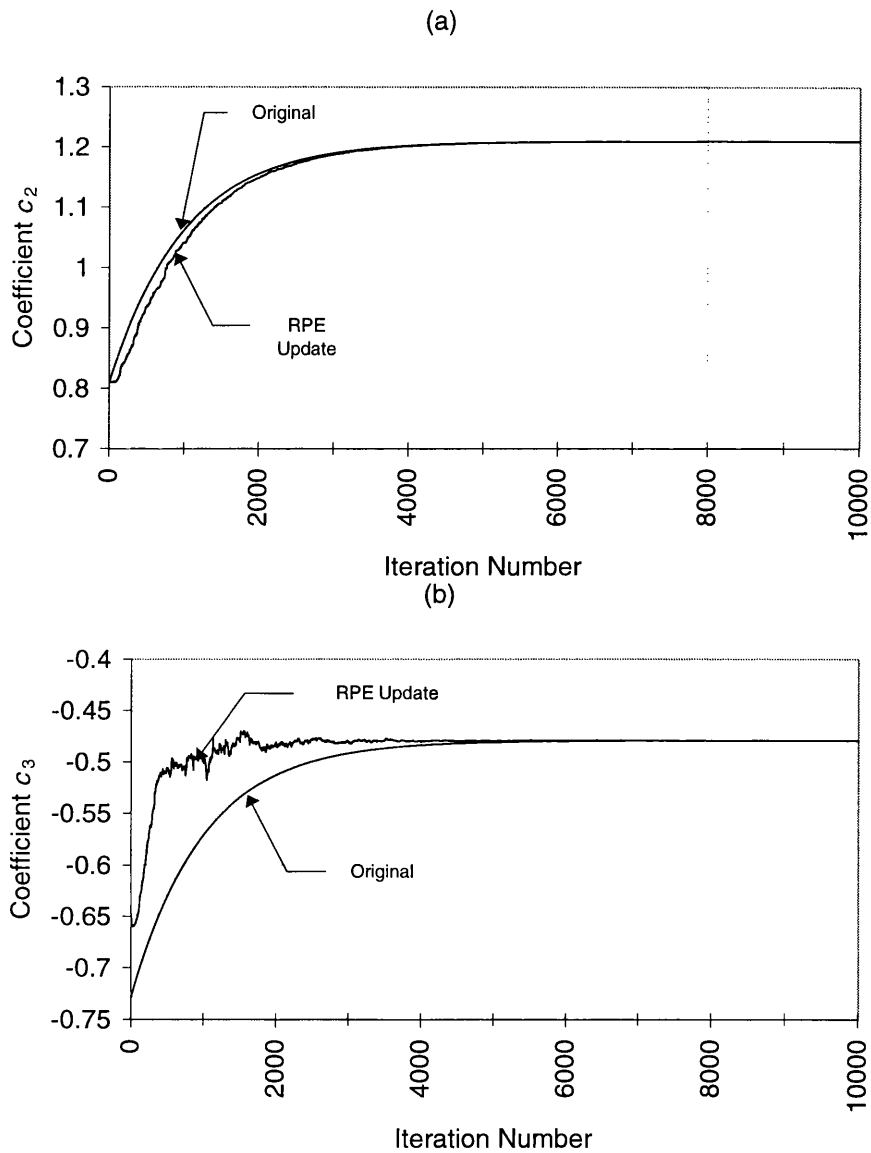

Figure 6.11: Adaptive modelling of (6.9) using an equivalent order adaptive filter. Trajectories of (a) $c_2(n)$ (b) $c_3(n)$

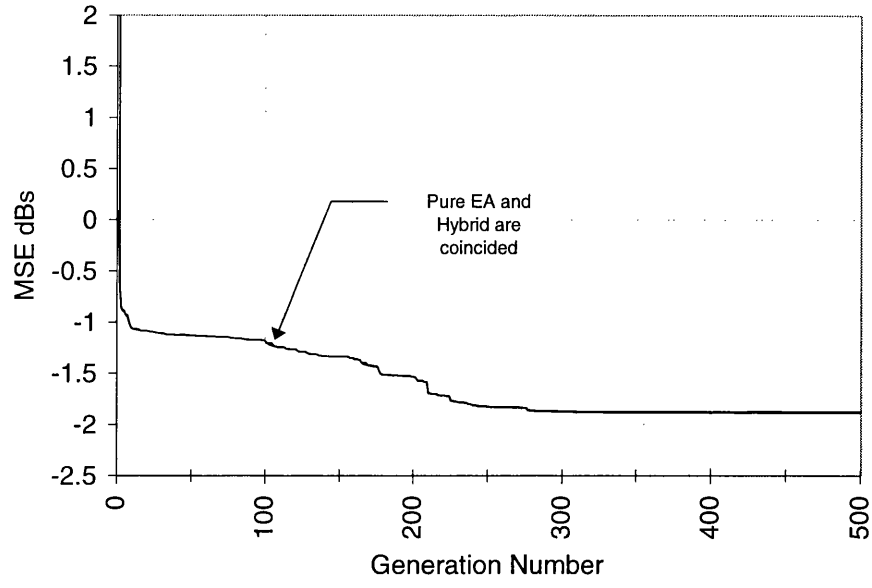

Figure 6.12: Offline adaptation while overmodelling (6.11) using an adaptive filter with $P = 3$ and $Q = 2$. MSE learning curves of pure EA and hybrid approach.

The parameters $\{a_2, a_3, a_4\}$ are assumed to be time-invariant. Figure 6.15 shows the MSE learning curves of hybrid and pure EA while offline adaptation. Figure 6.16 shows the trajectories of the time varying coefficients obtained during adaptation. Again the coefficients trajectories are not exactly matched to the original values due to the higher rate of parameter variations. Figure 6.17 shows the coefficient trajectories while modelling the same system except the parameters are varied slower rate than that of (6.13). i.e. in this case the parameters of the system to be modelled are

$$\begin{cases} b_0(n) = 1.0 - 0.5\{1 - \exp(-0.0001n)\}, \\ b_1(n) = -0.9 + 0.25\{1 - \exp(-0.0001n)\}, \\ b_2 = 0.81,\ b_3(n) = -0.729 + 0.5\{1 - \exp(-0.0001n)\}, \\ a_1(n) = 0.2314 + 0.05\{1 - \exp(-0.001n)\},\ a_2 = -0.43174, \\ a_3(n) = 0.340434,\ a_4 = -0.5184 \end{cases} \tag{6.14}$$

A major observation, which is made from above simulations is that normalised RPE algorithm perfectly tracks the time varying changes if the rate of parameter variations is much smaller than that of adaptation rate. The adaptation rate

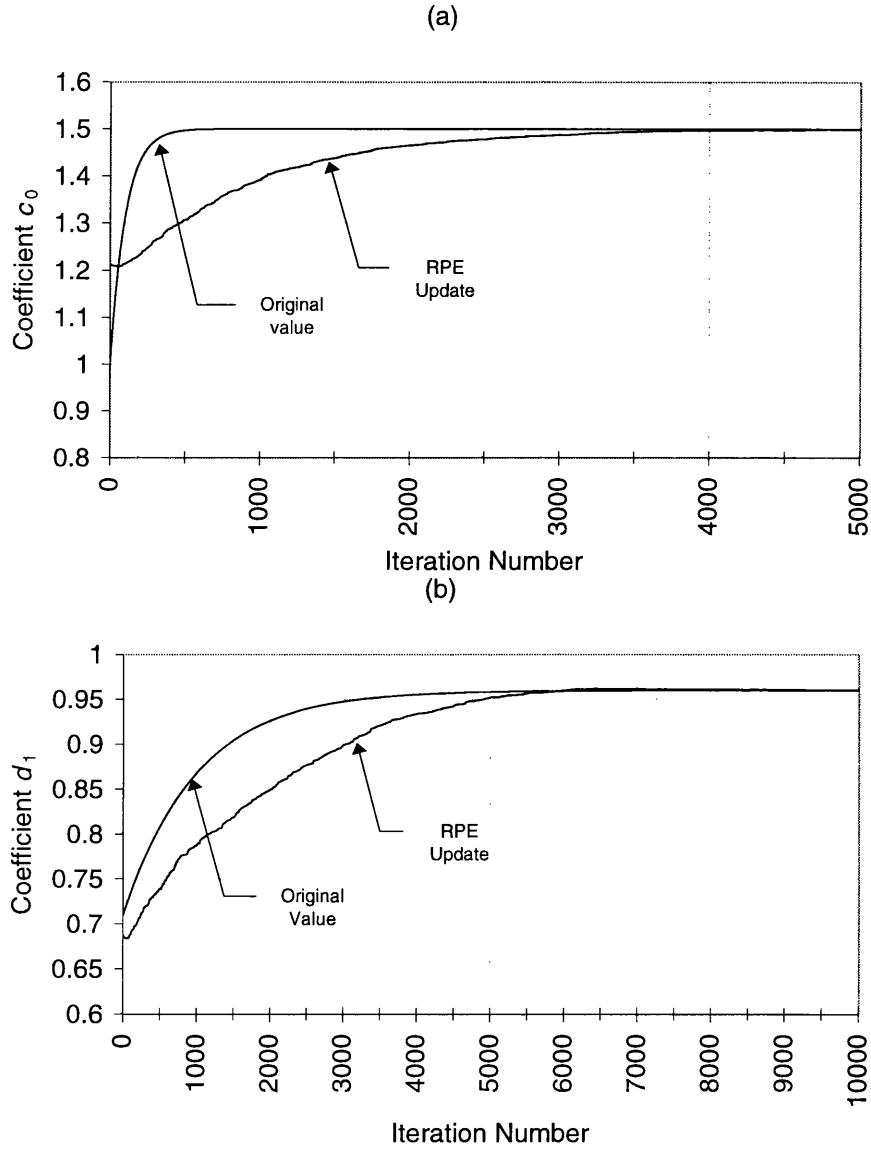

Figure 6.13: Adaptive modelling of (6.11) using an adaptive filter with $P = 3$ and $Q = 2$ (overmodelling). Trajectories of (a) $c_0(n)$ (b) $d_1(n)$

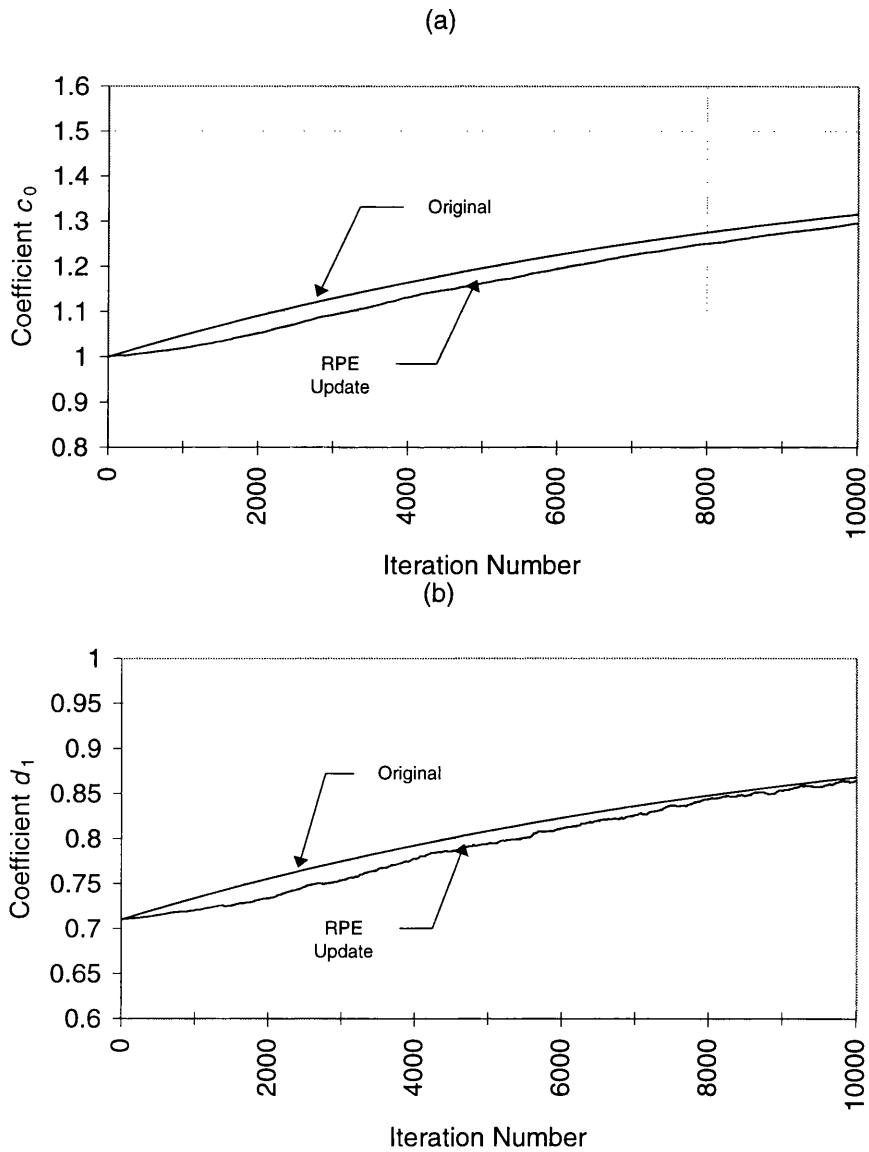

Figure 6.14: Adaptive modelling of (6.12) using an adaptive filter with $P = 3$ and $Q = 2$ (overmodelling). Trajectories of (a) $c_0(n)$ (b) $d_1(n)$

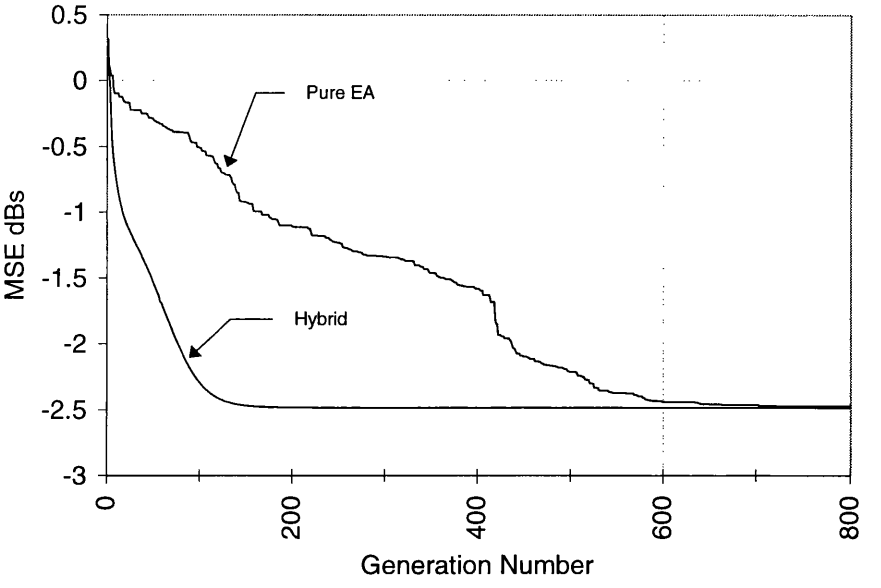

Figure 6.15: Offline adaptation while modelling (6.13) using an equivalent order adaptive filter. MSE learning curves of pure EA and hybrid approach.

can be controlled with a step size μ. Higher the step size can bring the adaptive algorithm into improper direction and can cause inaccurate estimates [63].

6.4 Adaptive Inverse Modelling

In the first case, we consider a FIR channel, which has the following parameters

$$\begin{cases} b_0(n) = 1.0 + 0.5\{1 - \exp(-0.0001n)\}, \\ b_1(n) = -1.4 + 0.05\{1 - \exp(-0.0001n)\}, \\ b_2(n) = 0.98 + 0.05\{1 - \exp(-0.0001n)\} \end{cases} \tag{6.15}$$

The parameters b_0, b_1 and b_2 are assumed to be time-variant. Inverse of the above channel is an IIR which has following z-transfer function

$$H(z) = \frac{1}{b_0(n) + b_1(n)z^{-1} + b_2(n)z^{-2}} \tag{6.16}$$

From (6.15) and (6.16) it is clear that even though the channel is a FIR, the resulting inverse filter will be an IIR which is strictly bounded with the stability criterion. Therefore it is evident from the above discussion that a linear inverse system can be found by a linear adaptive process if the channel has minimum

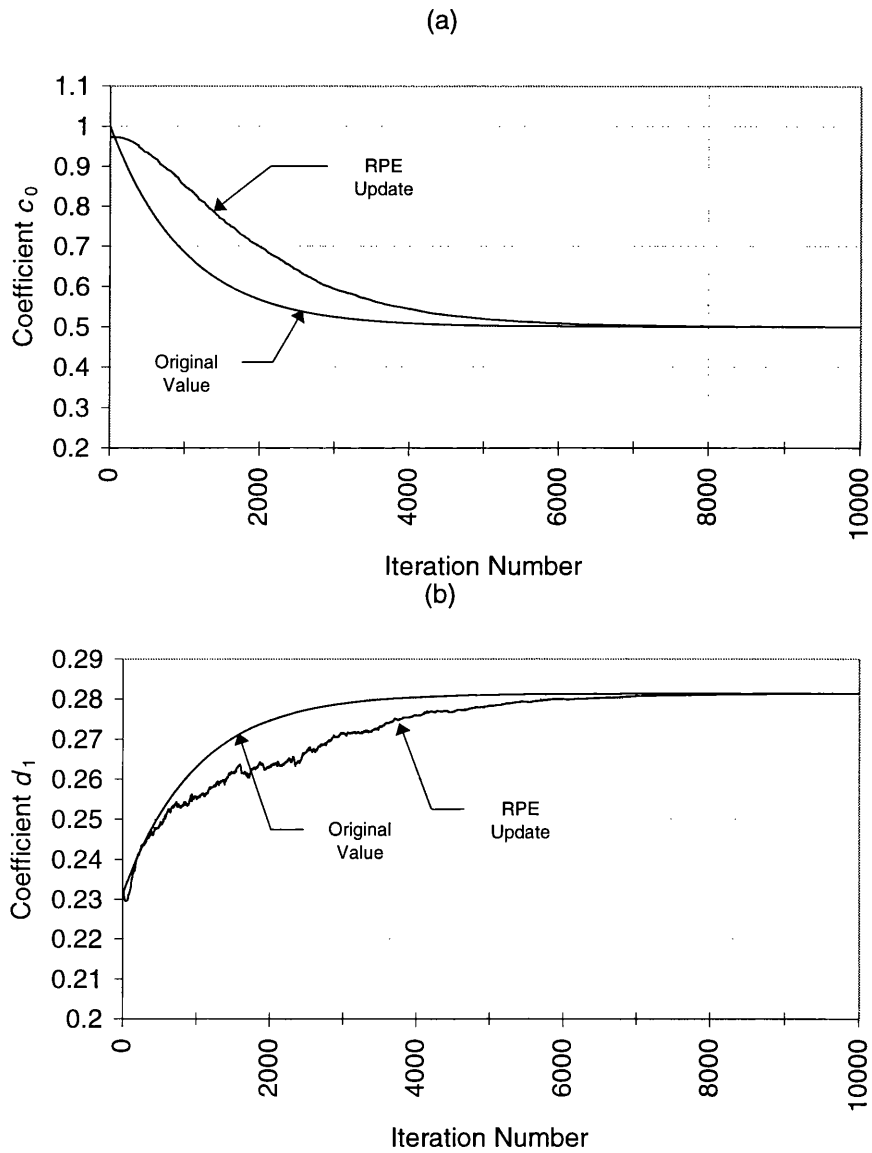

Figure 6.16: Adaptive modelling of (6.13) using an equivalent order adaptive filter. Trajectories of (a) $c_0(n)$ (b) $d_1(n)$

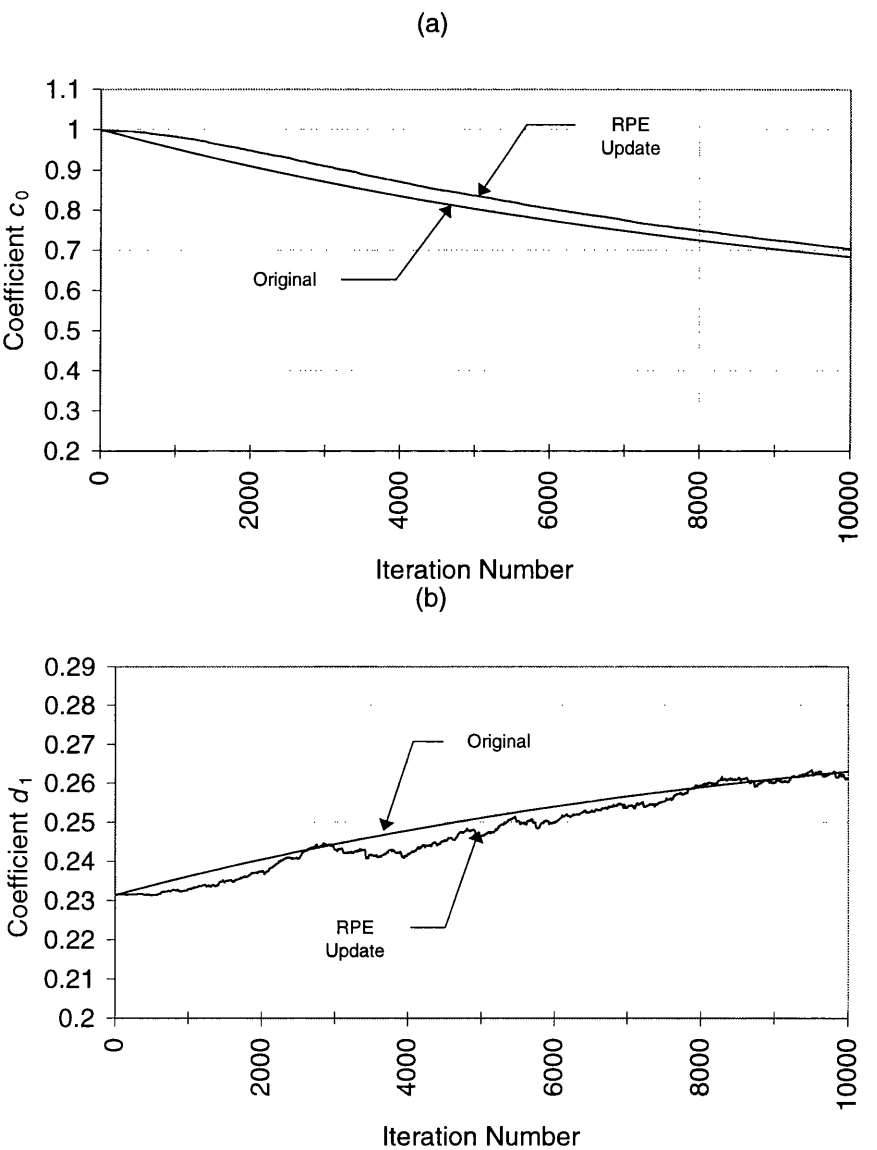

Figure 6.17: Adaptive modelling of (6.14) using an equivalent order adaptive filter. Trajectories of (a) $c_0(n)$ (b) $d_1(n)$

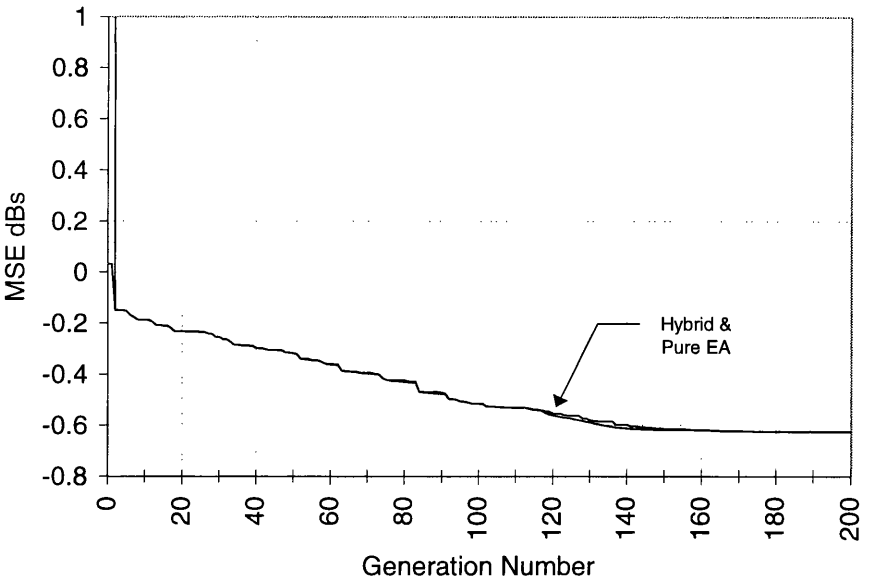

Figure 6.18: Offline adaptation. MSE learning curves of pure EA and hybrid approach while modelling the inverse of the non-minimum phase FIR channel of parameters (6.17) by the second-order adaptive filter given in (6.18)

phase, i.e. zeros lie inside the unit circle. For example, consider the above channel by ignoring the time-variant parameters, i.e. $b_0 = 1$, $b_1 = -1.4$, $b_2 = 0.98$. The inverse of this channel has poles that are very close to the unit circle, $p_{1,2} = \pm 0.9899$. The time varying parameters can easily bring this channel into non-minimum phase. Therefore the optimum inverse filter can become unstable. Let us assume that the channel parameters reach the following:

$$b_0 = 1, b_1 = -1.4, b_2 = 1.1 \tag{6.17}$$

at a time n and stay in the same state for a long time. Clearly, the channel with above parameters is a non-minimum phase and hence the optimum inverse filter is unstable. Figure 6.18 illustrates the MSE learning curves while modelling the inverse of the channel of parameters (6.17) using the hybrid approach and the EA itself with a second-order adaptive IIR filter of transfer function

$$A(z) = \frac{c_0}{1 - d_1 z^{-1} - d_2 z^{-2}} \tag{6.18}$$

Both algorithms fail to reach to the global minimum which lies at $\{b_0 = 1, a_1 = -1.4, a_2 = 1.1\}$, but they find a best solution in the stable region at $\{b_0 =$

$0.763368, a_1 = -1.27424, a_2 = 0.98813\}$. Figure 6.19 shows the trajectories of the coefficients of (6.18) during offline adaptation. The coefficients reach steady state values within 200 generations.

In summary, the application of linear filtering approach is unable to identify systems that have poles outside the unit circle. For example, inverse of a linear system modelling of the non-minimum phase channels given in (6.15) with the parameters (6.17) could not be achieved with the linear adaptive IIR filtering. The simulation results confirming this statement is illustrated in Figures 6.18 and 6.19. In this case, the algorithm fails to identify the optimum inverse filter, but find an inverse filter that provides smallest MSE of the stable region. An important point to note here is that EAs may find the global solution, but it is not always guaranteed.

By considering these issues the rest of this chapter will consider modelling of minimum phase channels. Consider the same channel, in this case we assume that all the parameters are fixed except b_1,

$$b_0 = 1,\ b_1(n) = -1.4 - 0.5\{1 - \exp(-0.0001n)\},\ b_2 = 0.98 \qquad (6.19)$$

It can be easily shown with a stability triangle [6, 63] that the channel with the above parameters can always lie within minimum phase. Before the tracking process is switched on, the global minimum of particular training samples is found by using hybrid approach via offline process. Figure 6.20 shows the MSE learning curves provided by the hybrid algorithm. As shown in this figure EA finds the global solution within 100 generation and normalised RPE algorithm takes another 100 generation for fine-tuning. After the global solution is found, the normalised RPE is switched on to track the time varying characteristics. The trajectory of the d_1 during adaptation is also shown in Figure 6.21. Figure 6.22 shows the input, output waveforms taken during adaptation. Figure 6.23 shows the modulus of the instantaneous error obtained at each updates.

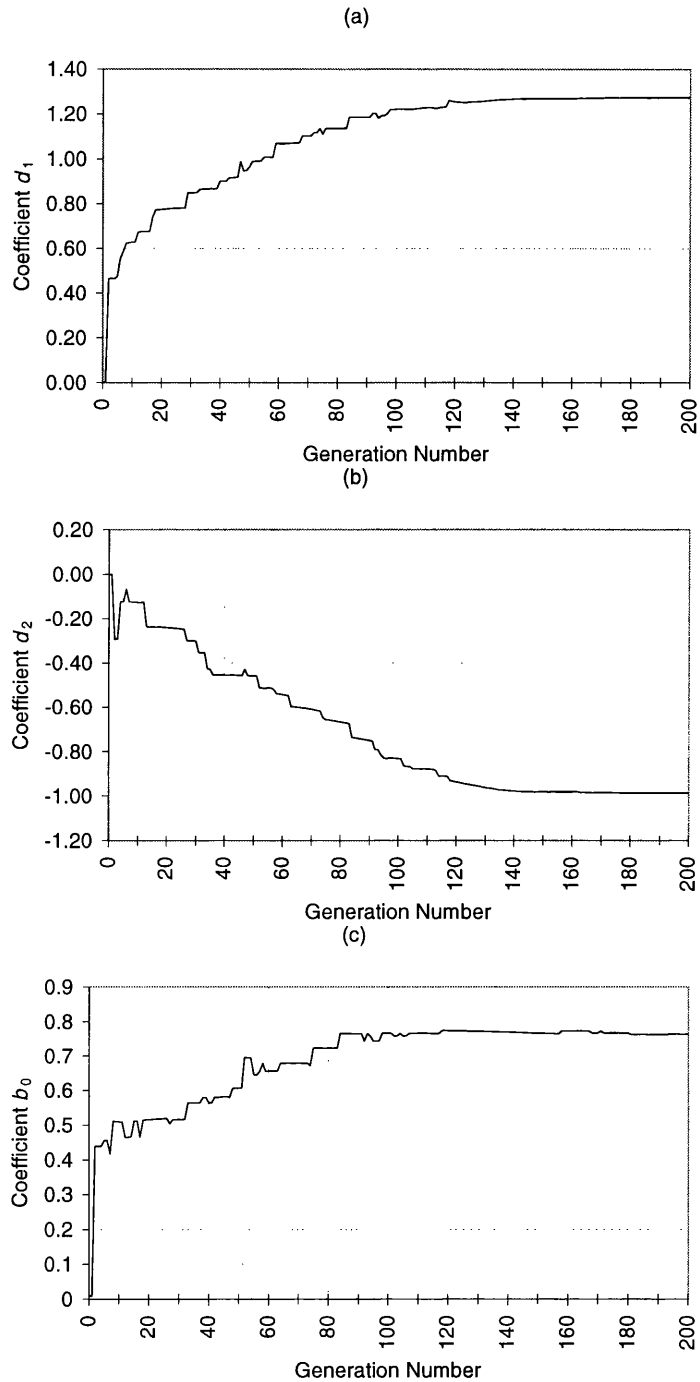

Figure 6.19: Inverse system modelling of a non-minimum phase channel of parameters (6.17) via offline adaptation. Trajectories of (a) d_1 (b) d_2 (c) c_0

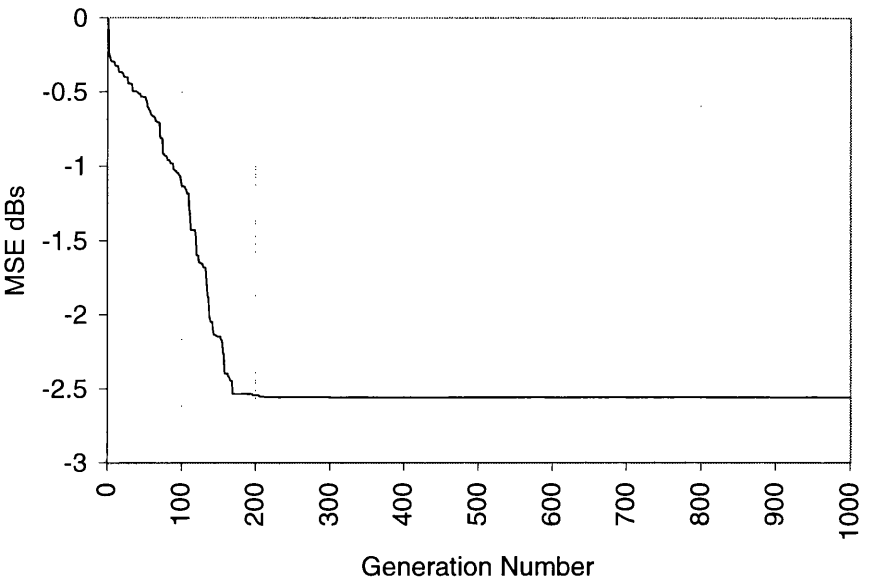

Figure 6.20: MSE learning curves of the hybrid approach while modelling the inverse of the FIR channel of parameters (6.19) by the second-order adaptive filter given in (6.18)

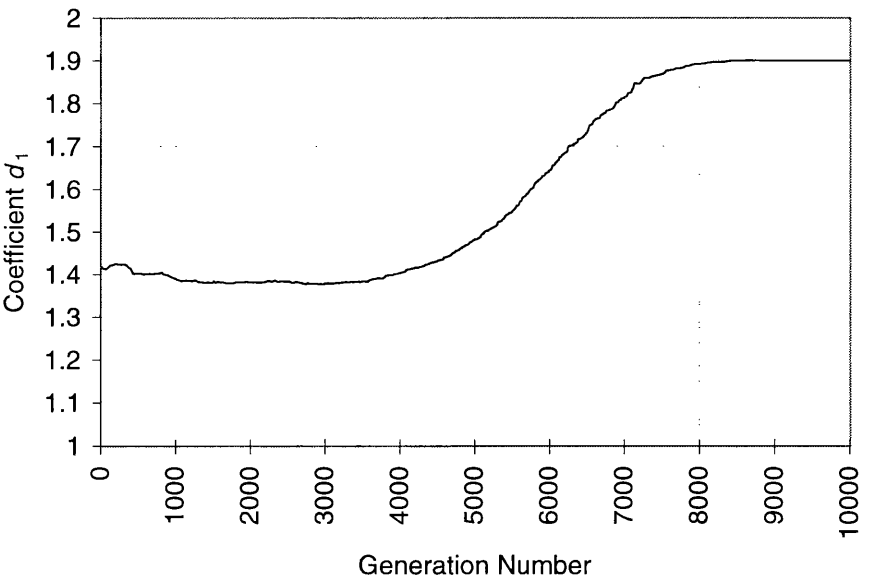

Figure 6.21: Adaptive inverse system modelling of the FIR channel of parameters (6.19). Trajectory of d_1

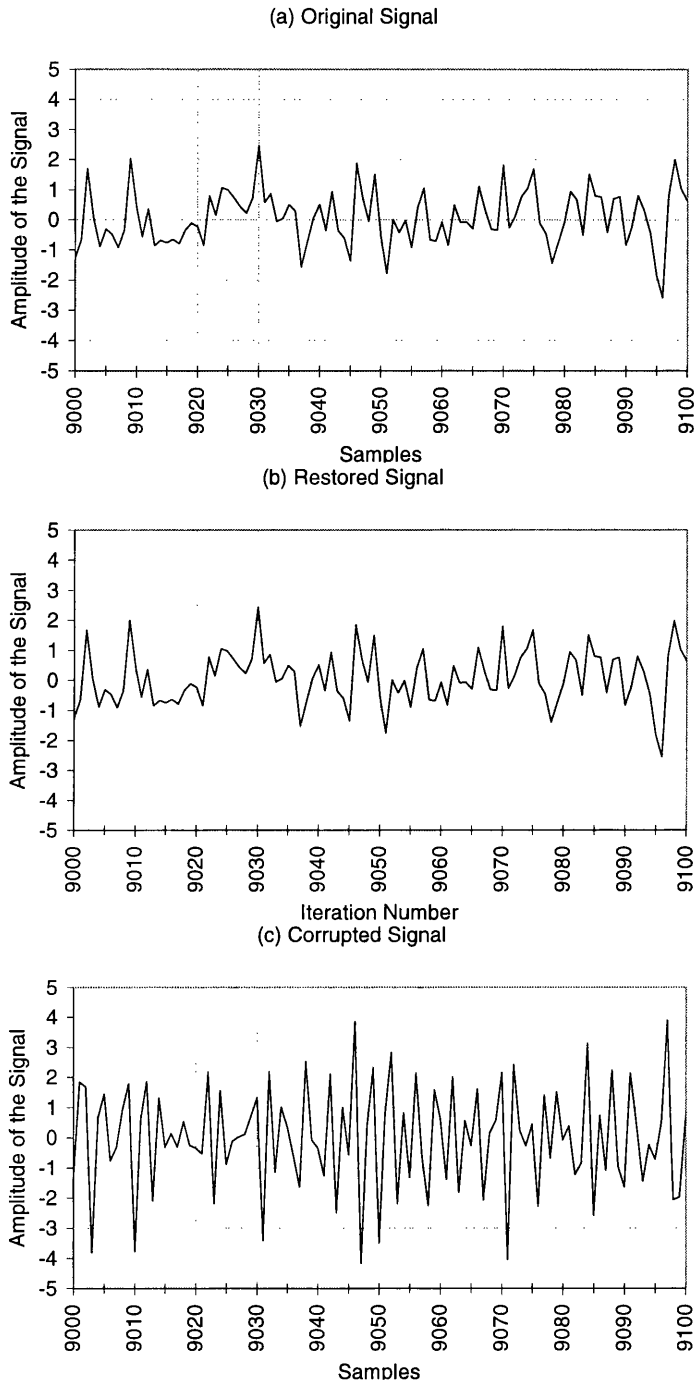

Figure 6.22: Adaptive inverse modelling. Input, output waveforms while modelling the inverse of the FIR channel of parameters (6.19) by the second-order adaptive filter given in (6.18). (a) Original signal (b) Restored signal (c) Corrupted signal (Channel output)

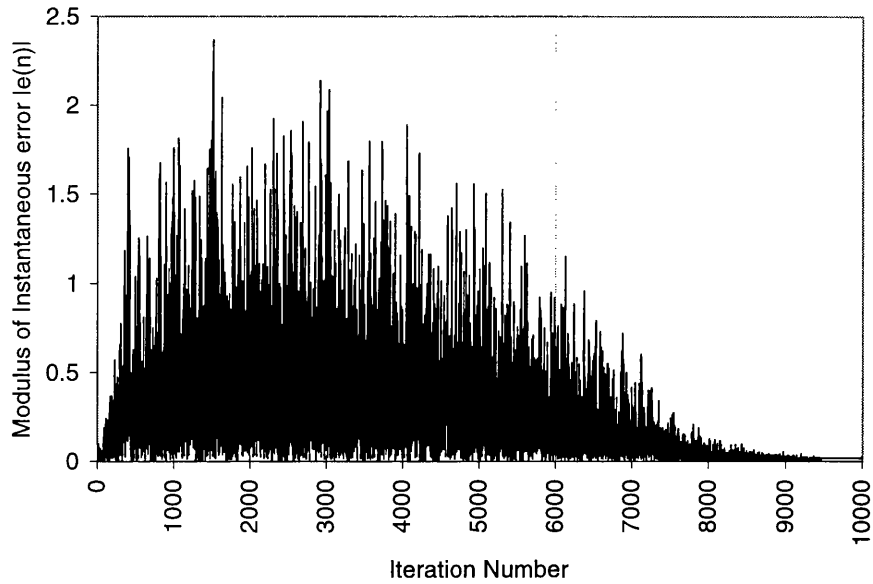

Figure 6.23: Adaptive inverse modelling of the FIR channel of parameters (6.15). Modulus value of the instantaneous error produced by normalised RPE.

Finally we consider a fourth-order FIR channel with the following parameters:

$$\begin{cases} b_0 = 1,\ b_1(n) = -0.9,\ b_2 = 0.81, \\ b_3(n) = -0.729 - 0.1\{1 - \exp(-0.0001n)\} \end{cases} \tag{6.20}$$

All the parameters except b_3 are stationary. The optimum inverse filter has z-transfer function

$$H_o(z) = \frac{1}{1 - 0.9z^{-1} + 0.81z^{-2} - 0.829z^{-3}} \tag{6.21}$$

It can be easily shown that the above inverse filter is stable. A fourth-order adaptive system of structure which is similar to (6.21) is used to model the inverse of the (6.20). Figure 6.24 shows the MSE learning curves of the pure EA and the hybrid algorithm when optimising the filters via offline. Evolutionary algorithm finds the global area within 100 generations. normalised RPE takes another 100 generations to fine-tune the coefficients. Once the global area is found, the algorithm is continued with normalised RPE to track time varying changes. Figure 6.25 shows the trajectories of d_3 while adapting the parameters using normalised RPE. The coefficient d_3 is adapted according to the changes.

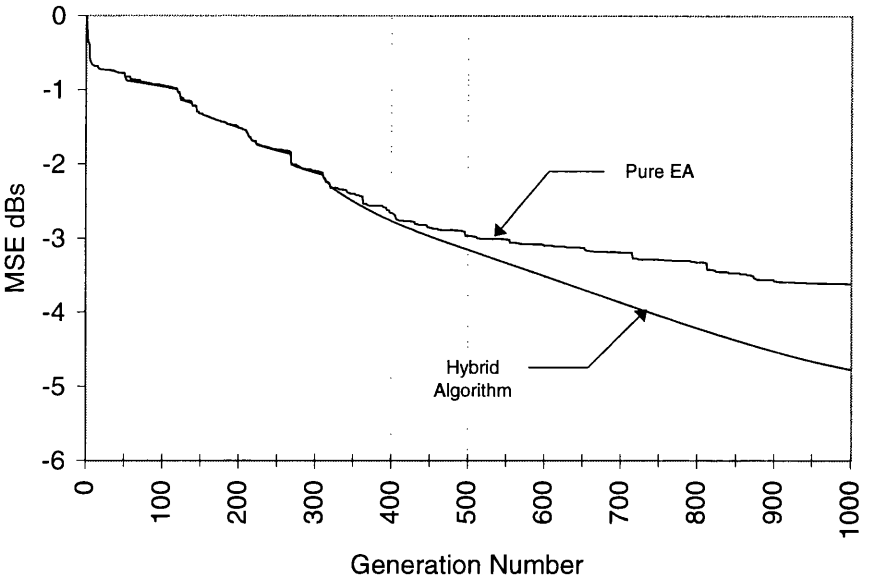

Figure 6.24: Inverse modelling of the FIR channel of parameters (6.20) via offline adaptation. The MSE learning curves of the pure EA and the hybrid algorithm

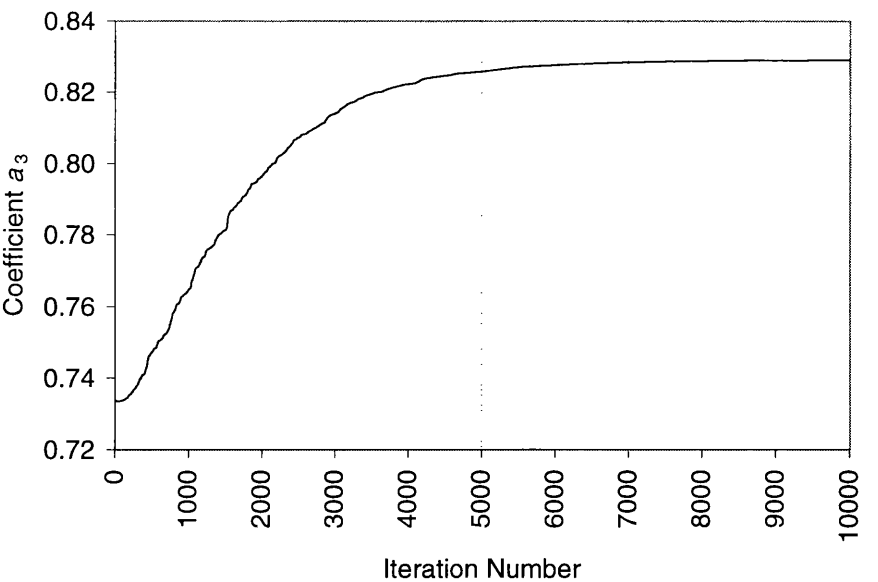

Figure 6.25: Adaptive inverse modelling of the FIR channel of coefficients (6.20). Trajectory of d_3.

Figure 6.26 shows the input, output and corrupted waveforms of the above experiment. Figure 6.27 shows the modulus of the instantaneous error obtained at each updates.

6.5 Summary and Discussion

In this chapter we have shown that hybrid methodology is very efficient and optimal for adaptive IIR filtering. We have successfully identified a wide variety of IIR linear models using the hybrid approach that employs EA and normalised RPE as searching tools. We have also shown how the time varying parameters of a linear system can be tracked with the help of a normalised RPE, which has been discussed in Chapter 2. In general, this chapter provided the techniques of designing direct form adaptive IIR digital filters in nonstationary environments.

These techniques can be used in many areas of signal processing where applications will demand adaptive IIR filtering. For example, equalising filters in telecommunication often demand IIR structures to remove interference introduced by those communication channels. Most communication channels are nonstationary and require adaptive equalisers. In such cases, hybrid approach can be used via offline to globalise the parameters and normalised RPE can be switched on to track the time varying changes.

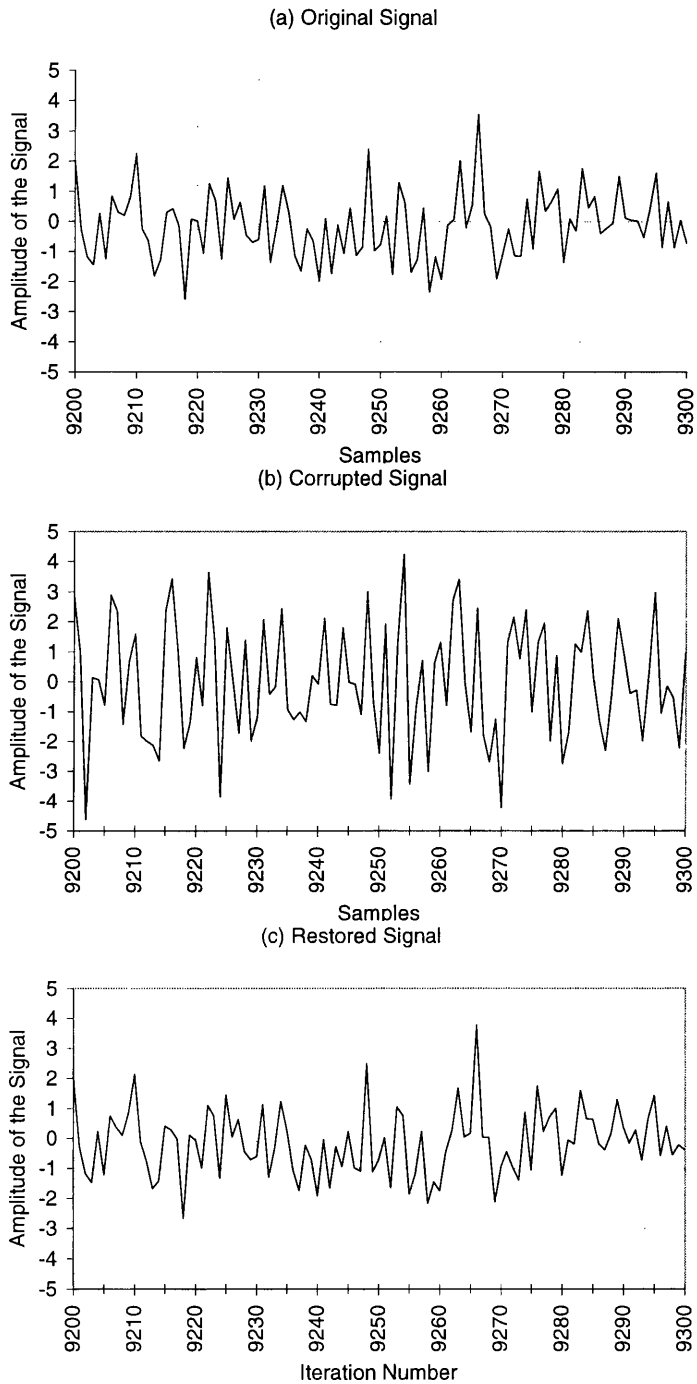

Figure 6.26: Adaptive inverse modelling. Input, corrupted and restored waveforms while modelling the inverse of the FIR channel of parameters (6.20) by a third-order adaptive filter of similar structure given in (6.21). (a) Original signal (b) Corrupted signal (Channel output) (c) Restored signal.

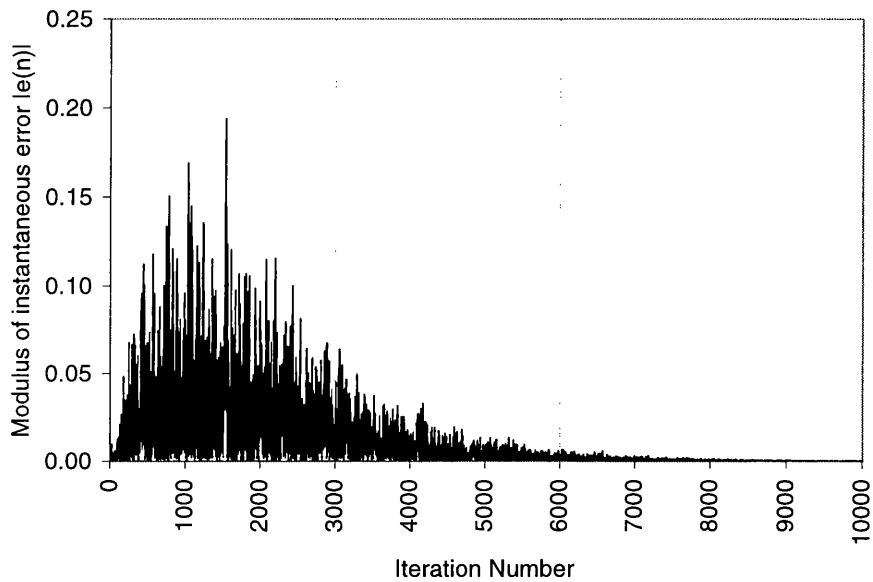

Figure 6.27: Adaptive inverse modelling of the FIR channel of parameters (6.20). Modulus value of the instantaneous error produced by normalised RPE.

Chapter 7

Adaptive Channel Equalisation and Noise Cancellation

In the last two chapters of this dissertation, the theoretical development of evolutionary techniques for the design of adaptive IIR filters has been presented along with validation using modelling applications. This chapter aims to show practical problems using these techniques and adaptive filters. These are predominantly in the areas of adaptive channel equalisation [56] and noise cancellation [81]. These areas have been active areas of research in which applications of adaptive IIR filters are of paramount importance. However, the use of adaptive IIR filtering techniques in these areas are relatively low due to the problems encountered in existing adaptive filtering approaches.

This chapter shows how the proposed IIR filtering technique can be used in the areas with less computational costs. In particular, this chapter presents several comparative results, which compare the frequency and bit error rate (BER), the ratio of misclassified to correct symbols at the receiver, performances of low order IIR filters against various taps FIR filters when designing these filters for communication channel equalisation. Moreover, the necessity of adaptive IIR filtering is clearly outlined by describing the concepts of Noise Cancellation technique. The applications of the proposed techniques are presented with

the theoretical models developed from C++ programming language. The simulation study has been carefully conducted by considering some of major real problems into the account and the simulations were carried out in an Intel-P133, 110 MHz Pentium processor.

The rest of this chapter is organised as follows: In Section 7.1 the problems in communication channels are discussed. Also, this section describes the need of channel equalisation. Section 7.2 illustrates adaptive equalisation. This section first introduces trained equalisers in QAM systems and then provides some results obtained when designing various IIR equalising filters using the techniques discussed in the previous chapters. The frequency responses of the designed IIR equalisers are compared with various-taps FIR filters optimised through classical algorithms. Bit error rate performance is also compared with FIR equalisers. This section then illustrates blind equalisation. The results of blind equalisation are shown when equalising various multipath channels in nonstationary environments. In Section 7.3, noise cancellation is given. Finally, in Section 7.4 a brief summary of this work is given with discussion.

7.1 Channel Equalisation

A major problem in digital communication is that signal fading due to channel distortions [28]. Fading is defined as any variation in signal strength, relative phase or polarisation of any of the frequency components of a received radio signal due to the characteristics of the propagation path. In addition to channel distortions, the transmitted symbols are subject to other impairments such as thermal noise, impulse noise, and nonlinear distortions arising from the modulation and demodulation process, cross talk interference and the use of amplifiers. If the communication channel is linear, a model of the received signal is therefore:

$$R(z) = C(z).S(z) + V(z) \tag{7.1}$$

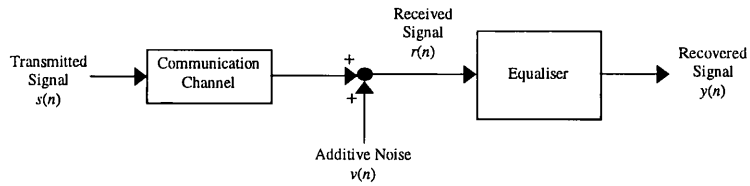

Figure 7.1: Received signal model of a communication system.

where $R(z)$, $S(z)$, $C(z)$ and $V(z)$ are the z-domain representation of the received signal $r(n)$, transmitted signal $s(n)$, channel impulse response $c(n)$, and additive noise $v(n)$ respectively. Received signal model of a communication system is illustrated in Figure 7.1.

Removal of interference in communication channels can be achieved by various approaches. For example, space and frequency diversity systems are currently being used to remove channel interference of multipath propagation path. In recent years, the improvement in adaptive signal processing has encouraged the study of adaptive channel equalisation techniques. As a result, adaptive FIR filtering approach has gradually been replacing the existing non-DSP techniques such as space and frequency diversity systems to remove channel interference and noise [76]. Unfortunately, these first generation adaptive system have certain performance limitations due to high computational requirements and poor performance of the FIR structures as discussed in the earlier chapters.

7.2 Adaptive Channel Equalisers

Channel equalisation problems are categorised into two main classes: *linear* and *nonlinear* equalisation [56]. Linear adaptive filters can provide equalisation of linear channels, where they use inverse filtering approach to cancel the interference. A linear adaptive equaliser cannot restore signals corrupted by nonlinear mechanisms. In such cases nonlinear adaptive filters [28] can be used, but they are beyond the scope of this research.

Adaptive equalisers are further classified into two categories: *trained equaliser*

or *blind equaliser*. Trained equalisers use the originally transmitted sequence during adaptation, while in blind equalisation, the adaptation of the equalising filter is attempted in a way to match the statistics of the output of the equaliser to those of the transmitted sequence [28].

7.2.1 Trained Equalisation: Adaptive Equalisers in QAM Systems

Quadrature amplitude modulation is an efficient technique to reduce system band-width, where two double-sideband suppressed-carrier amplitude-modulated signals can be superimposed, and separated at the receiver, using quadrature or orthogonal carriers for modulation and demodulation [76, 55]. Most high speed baseband modems, e.g. 56000 bits/s, employ QAM to reduce the band-width requirements of a voice-band channel [56]. As well as linear distortions, speech-band channels generally introduce frequency offset and phase jitters on to the data signal. Linear distortions cause inter symbol interference (ISI) on each quadrature component and cross coupling interference (CCI) between the two baseband channels [56]. The modem receivers use some form of carrier-phase tracking circuitry to remove frequency offset and reduce phase jitters. Quadrature amplitude modulation is as efficient in bits per second per hertz as vestigial or single-sideband amplitude modulation, yet enables a coherent carrier to be derived and phase jitters to be tracked using easily implemented decision-directed carrier recovery techniques [76, 56]. A timing waveform with negligible timing jitters can also be easily recovered from QAM signals. However, an equalisation process can only remove the linear distortions introduced into the quadrate components.

A model of an equivalent baseband channel can be drawn as the cross connected networks shown in Figure 7.2. A very convenient way of representing this is to regard the two data inputs (and outputs) as real and imaginary components and then the equivalent baseband response may be represented by a

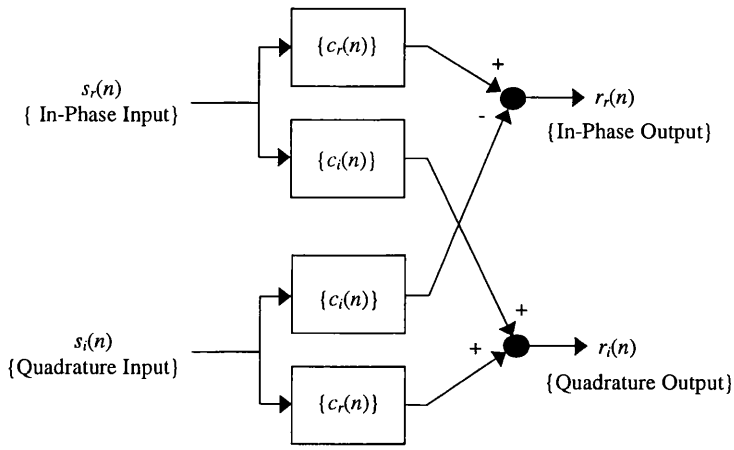

Figure 7.2: Equivalent complex channel.

complex impulse response. Consequently the model of the received signal can be represented by a single operation:

$$r^\star(n) = s^\star(n) * c^\star(n) + v(n) \tag{7.2}$$

where $r^\star(n)$, $c^\star(n)$, $s^\star(n)$ and $v(n)$ are the impulse responses of the received signal, channel response, input, and noise interference respectively. The $\star$ symbol is shown to represent the complex impulse responses,

$$s^\star(n) = s_r(n) + s_i(n) \tag{7.3}$$

$$c^\star(n) = c_r(n) + c_i(n) \tag{7.4}$$

$$\begin{aligned} r^\star(n) &= r_r(n) + r_i(n) \\ &= \{s_r(n) * c_r(n)\} + \{s_i(n) * c_i(n)\} \\ &\quad + \{s_r(n) * c_i(n)\} + \{s_i(n) * c_r(n)\} \end{aligned} \tag{7.5}$$

where subscripts r and i represent real and imaginary components. The concept of a complex channel response is therefore very useful for the design of QAM modems. To equalise or cancel a complex channel response, a complex adaptive filter is required which can use 4 such real operations in one structure which is clearly seen from the equations (7.2) - (7.5). Figure 7.3 presents a typical QAM communication system, which employs a complex adaptive filter to remove the channel interference.

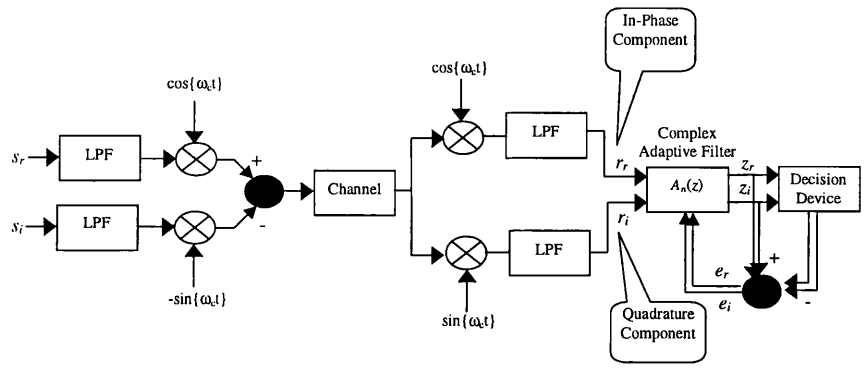

Figure 7.3: QAM system with baseband complex adaptive equaliser.

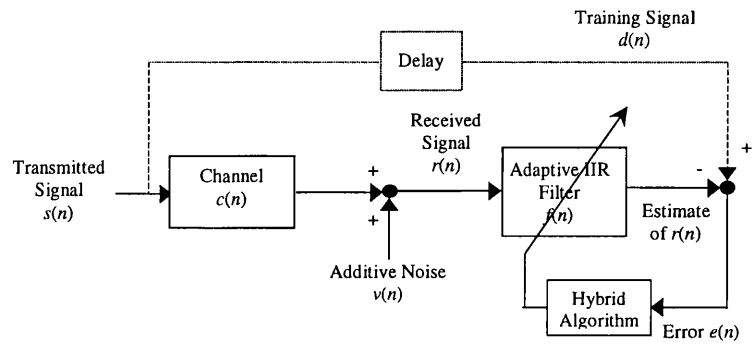

Figure 7.4: An experimental arrangement of channel equalisation.

7.2.2 Results and Validation

In this study, a linear channel with complex response is chosen to represent QAM system. Figure 7.4 illustrates an experimental arrangement of channel equalisation, where $c(n)$ and $f(n)$ are assumed to represent the impulse responses of the channel and the adaptive filter respectively. The channel transfer function is assumed to be of the form

$$C(z) = \left\{ \frac{\prod_{i=1}^{M}(1 - b_i e^{j\phi_i} z^{-1})}{\prod_{k=1}^{L}(1 - a_k e^{j\theta_k} z^{-1})} \right\} \tag{7.6}$$

where $\{a_k, b_i\}$ and $\{\phi_i, \theta_k\}$ are the factors representing the ISI and CCI of the QAM channels. The input signal $s(n)$ considered is assumed to be

$$s(n) = A\exp(j\omega_c n + x(n)) \tag{7.7}$$

where A represents the amplitude of the signal, ω_c denotes the centre frequency, and $x(n)$ is a baseband signal. This signal has a constant envelope property with an amplitude A. In addition to the linear distortion, a zero mean white noise is added at the output of the channel to represent possible noise interference, which appears during signal transmission. The standard deviation is chosen so as to meet equivalent signal to noise ratio (SNR) of at least 30 dB. This is considerably a higher value, but can be expected in communication systems due to the interference caused by amplifiers and converters. Our aim is to show how the channel interference can be removed in noisy environments, but not to show cancelling of noise. Appendix D shows how the SNRs are calculated for these signals. Trained adaptation is used and assumed that the original signal is available at the receiver during adaptation. Considering these real problems into account, the parameters of the channel (7.6) are assumed to be

$$\begin{cases} a_1 = 0.97,\ a_2 = 0.95,\ a_3 = 0.98,\ a_4 = 0.85, b_1 = 0.9,\ b_2 = 0.94, \\ \phi_1 = 0.3\pi,\ \phi_2 = -1.2\pi,\ \phi_3 = -0.75\pi,\ \phi_4 = 1.5\pi, \\ \theta_1 = 0.3\pi,\ \theta_2 = -0.2\pi \end{cases} \tag{7.8}$$

to represent a channel with poles closer to the unit circle, and

$$\begin{cases} a_1 = 0.86,\ a_2 = 0.75,\ a_3 = 0.8,\ a_4 = 0.7, b_1 = 0.75,\ b_2 = 0.8, \\ \phi_1 = -0.5\pi,\ \phi_2 = -1.2\pi,\ \phi_3 = -0.75\pi,\ \phi_4 = 1.5\pi, \\ \theta_1 = 0.6\pi,\ \theta_2 = -0.2\pi \end{cases} \tag{7.9}$$

were assumed to represent a channel with the poles well inside the unit circle. The signal $s(n)$ is generated with

$$\{\omega_c = 0.1rad,\ A = 1,\ x(n) = 0.05\sin(0.001n)\} \tag{7.10}$$

The work in this dissertation is to find the optimum setting of parameters rather than finding the model structures. Various modal selection criteria are reported in [55, 5] to find the structure of the linear modals (modal order) with ease. For example, *Akaike information criterion* (AIC) can be used to select the order of the numerator and denominator coefficients of an adaptive IIR filter

[55]. Assuming that the filter orders can be found from the above mentioned techniques, a population of adaptive filters each having 5 numerator coefficients and 2 poles are chosen to model the inverse of the channel (7.6) of parameters (7.8) and (7.9) respectively.

Original data sets of 10000 samples, $\{s(n)\}_{n=0}^{9999}$, are generated and applied to the channel to obtain channel output. The output of the channel is then added to the Gaussian noise so as to obtain received signal, $\{r(n)\}_{n=0}^{9999}$. Among 10000 samples, a training set of 100 pairs of $\{s(n)\}$ and $\{r(n)\}$ are taken to train the adaptive filters as shown in Figure 7.4. For the given training set, EA is used to evolve a population of complex IIR filters with the population size = 50. The MME value, ε, averaged over the window size, $w = 100$ (number of training data), is minimised through the hybrid algorithm. Minimising the MME, thus

$$\min\{\varepsilon\} = \min \mathrm{E}|e(n)| = \min \mathrm{E}\,|d(n) - \hat{r}(n)| \tag{7.11}$$

where $\hat{r}(n)$ is an estimate of $r(n)$ given by

$$\begin{aligned} \hat{r}(n) &= r(n) * \hat{f}(n) \\ &= \{d(n) * c(n) + v(n)\} * \hat{f}(n) \\ &= \{d(n) * c(n) * \hat{f}(n) + v(n) * \hat{f}(n)\} \end{aligned} \tag{7.12}$$

where

$$r(n) = \{d(n) * c(n) + v(n)\} \tag{7.13}$$

Therefore

$$\begin{aligned} \min\{\varepsilon\} &= \min \mathrm{E}\,|d(n)(1 - c(n) * \hat{f}(n))| - \\ &\qquad \min \mathrm{E}\,|v(n) * \hat{f}(n)| \end{aligned} \tag{7.14}$$

where $\hat{f}(n)$ is the impulse response of the estimated adaptive filter, $f(n)$. It is clear from the equation (7.14) that the net effect of minimising the MME is to

minimise the first term of right-hand-side of the equation and thus results

$$\min \varepsilon \approx \min \mathrm{E}\,|v(n) * \hat{f}(n)| \tag{7.15}$$

The genetic parameters such as crossover probability α_c, mutation probability α_m, and the standard deviation σ are set equal to 0.85, 0.05 and 0.005 respectively. The probability of immigration, α_i, is set 0.02. The step size μ normalised RPE is set equal to 0.001. The hybrid algorithm is allowed to run over 8000 generation unless otherwise termination criterion is met. The simulation is repeated for 15 independent EA runs using the same training data but in different system environments and with distinct initial conditions. The final results are taken by averaging the results obtained from 15 independent runs.

Figure 7.5 shows the power spectrum of input, output and the original signals of equalisation of the channel (7.6) of parameters (7.8). This channel with these particular parameters has poles, which are closer to the unit circle. The spectral analyses show how the signal power varies against its frequencies. The power spectrum of the channel output clearly shows the effect of the channel's response on the transmitted signal. Corrupted signal has several spectral spikes, which appears throughout the spectrum. These spikes represent the distortion introduced by the channel. The noise interference on the other hand produces white spectrum, which appears throughout the frequency band. Even though the IIR equaliser perfectly models the inverse of the channel, the noise interference is still appeared at the output of the equaliser. This clearly proves that the noise interference, which is added to the signals during transmission, cannot be removed from linear inverse filtering approach.

Figure 7.6 shows the power spectrum of the signals obtained from FIR equalisers with various filter lengths. Finite impulse response filters with 32, 64 and 256-taps are employed along with LMS algorithm to equalise the channel (7.6) of parameters (7.8). In appendix E, we have shown how the FIR-LMS algorithm are developed for this experiment. Even a 256-tap FIR equaliser still provides less performance than a second-order IIR equaliser. This proves that

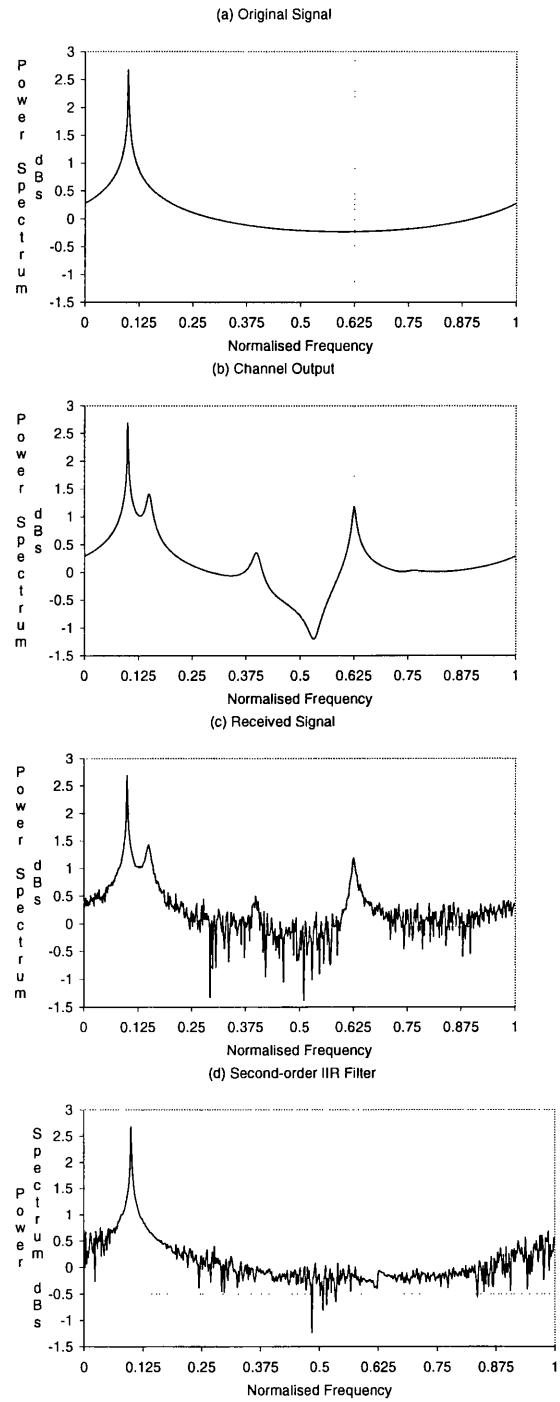

Figure 7.5: Power spectrum of input, corrupted and output of equalisation of the channel (7.6) of parameters (7.8). (a) Original signal. (b) Channel output. (c) Received signal (channel output contaminated with white noise). (d) Recovered signal by a second-order IIR filter optimised through the hybrid algorithm

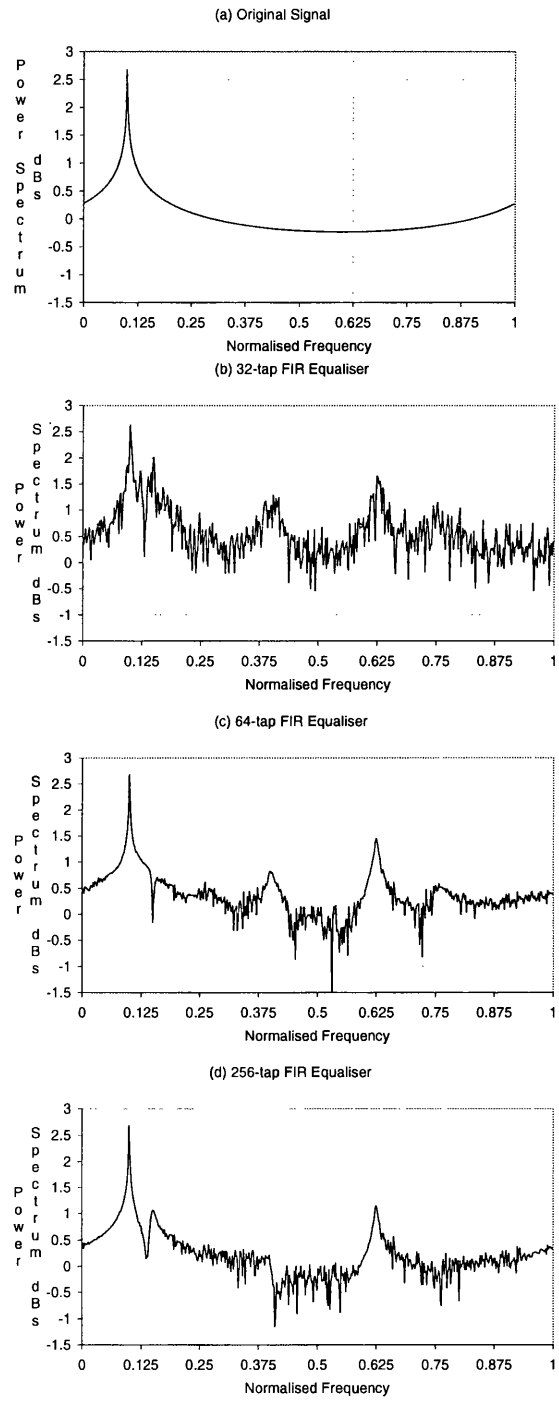

Figure 7.6: Power spectrum of signals. Original and recovered signals from FIR equalisers of various filter lengths when equalising the (7.6) of parameters (7.8). (a) Original signal. (b) Recovered signal by 32-tap FIR filter. (c) Recovered signal by 64-tap FIR filter. (d) Recovered signal by 256-tap FIR filter.

the frequency response of an IIR system is significantly better than equivalent FIR filters.

Figures 7.7 - 7.8 provide the results of channel equalisation of the channel (7.6) of parameters (7.9). The power spectrum of the input, output and the original signals are illustrated in Figure 7.7. This channel has poles, which are well inside the unit circle. The effect of this channel is clearly seen from the Figure 7.7(b). This channel introduces a spectral spike in the frequency band, but its effect is very less when compared to the previous channel shown in Figure 7.5(b). However, FIR equaliser still requires much more coefficients to compensate for the distortions. The recovered signal from a second-order IIR equaliser is shown in Figure 7.7(d), while Figures 7.8(b)-(d) illustrate the results obtained from 32, 64 and 256-tap FIR equalisers. In this case the FIR equaliser of order 256 does better equalisation. However, its performance is still worse when compared to those of IIR equaliser. An IIR equaliser of order 2 perfectly equalises this channel.

Figure 7.9 shows the learning curves which are obtained while optimising the IIR filters for channel equalisation of parameters (7.8) and (7.9) respectively. In both cases the hybrid algorithm finds the solutions within 4000 generations.

We also present an experiment with very high SNR to show the ability of IIR equalisers in less noise environments. For this experiment, the channel parameters are assumed to be

$$\begin{cases} a_1 = 0.98,\ a_2 = 0.95,\ a_3 = 0.94,\ a_4 = 0.92, b_1 = 0.92,\ b_2 = 0.85, \\ \phi_1 = 0.7\pi,\ \phi_2 = 1.5\pi,\ \phi_3 = 1.75\pi,\ \phi_4 = 0.5\pi, \\ \theta_1 = -1.3\pi,\ \theta_2 = 1.2\pi \end{cases} \tag{7.16}$$

This channel has poles, which are very much closer to the unit circle and hence introduces severe distortions in the frequency band. In this experiment, the channel output is added to a white noise sequence so as to produce SNR of 60 dB. The power spectrum of the input/ output and the recovered signals are shown in Figures 7.10 and 7.11 respectively. Equalising performance of an IIR filter is clearly seen from Figure 7.10(d). Figure 7.12 illustrates the learning

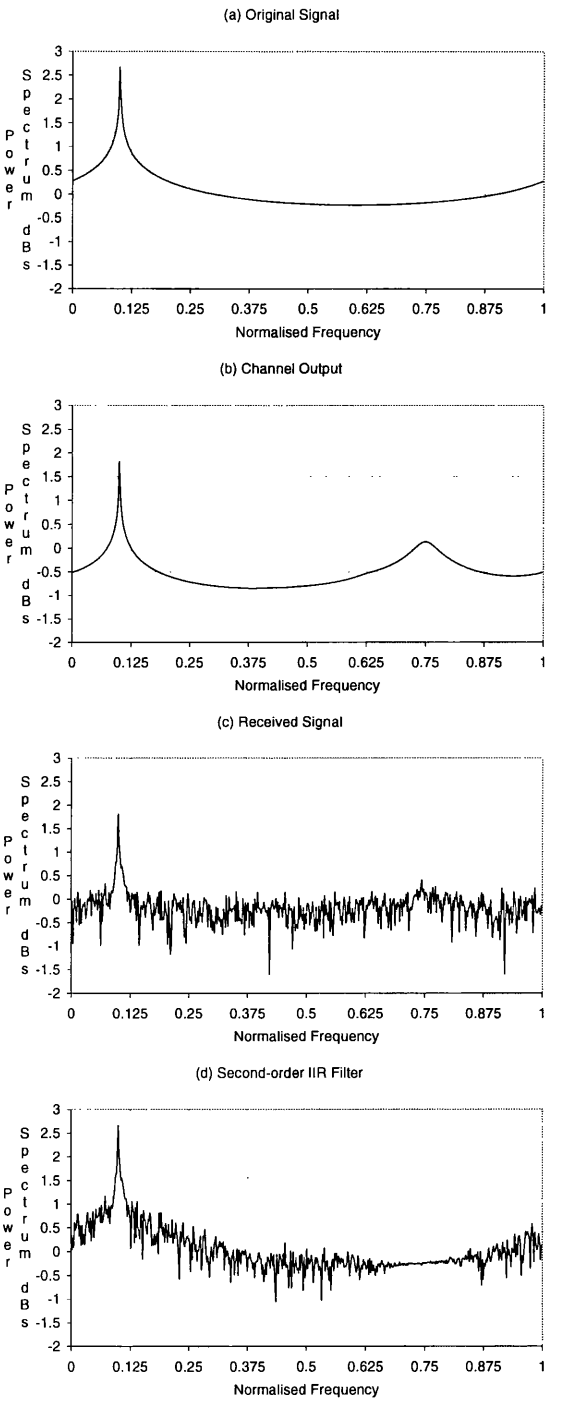

Figure 7.7: Power spectrum of input, corrupted and output when equalising the channel (7.6) of parameters (7.9). (a) Original signal. (b) Channel output. (c) Received signal contaminated in white noise. (d) Recovered signal by second-order IIR filter optimised through the hybrid algorithm.

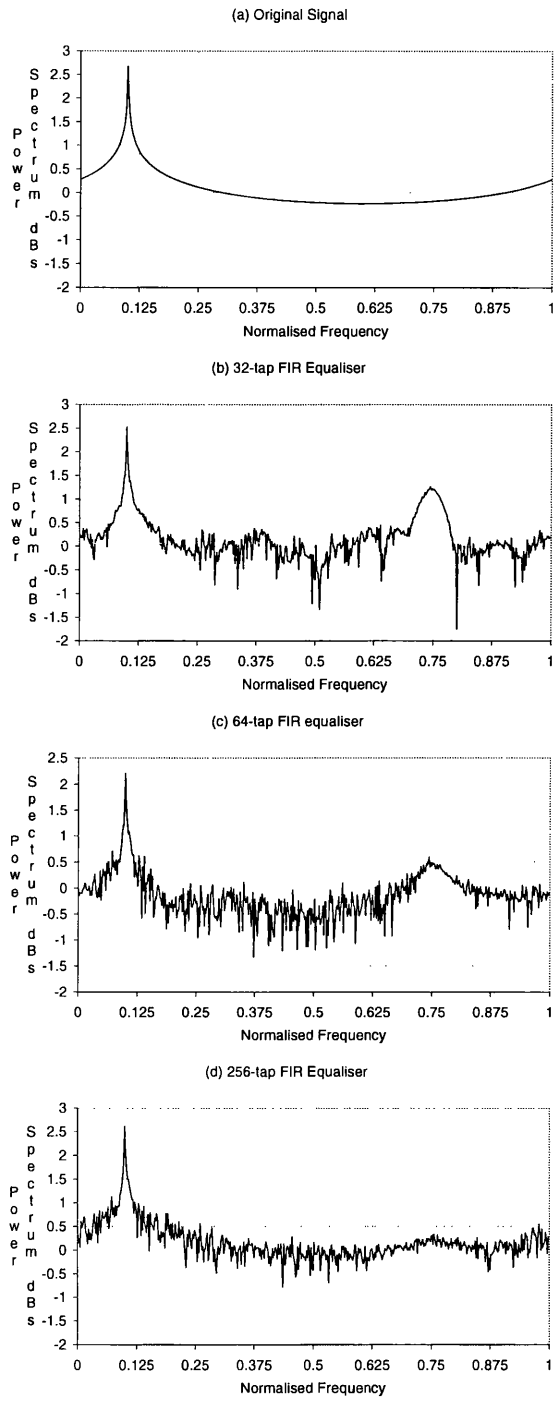

Figure 7.8: Power spectrum of signals. Original and recovered signals from FIR equalisers of various filter lengths when equalising the channel (7.6) of parameters (7.9). (a) Original signal. (b) Recovered signal by 32-tap FIR filter. (c) Recovered signal by 64-tap FIR filter. (d) Recovered signal by 256-tap FIR filter.

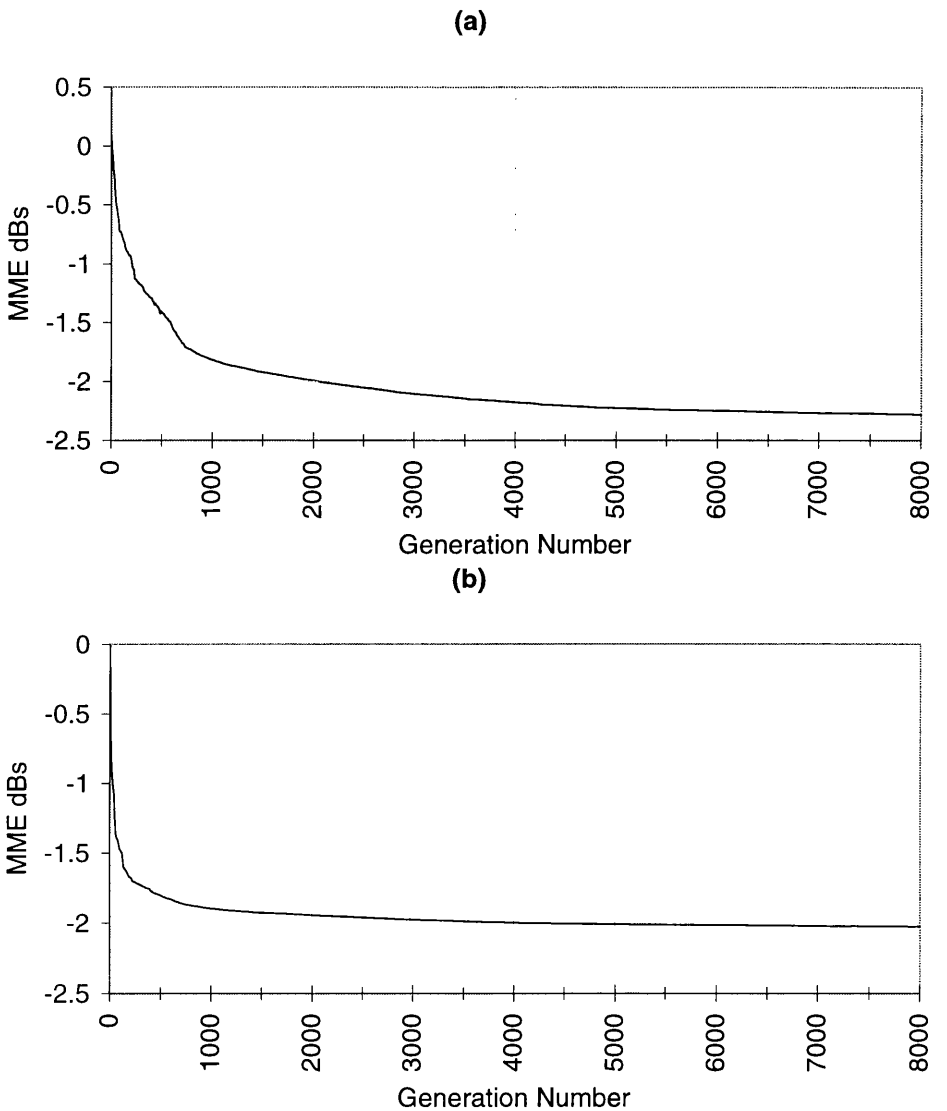

Figure 7.9: Learning curves showing the MME values plotted against generation number. (a) Learning curve of channel equalisation while equalising the channel (7.6) of parameters (7.8). (b) Learning curve of channel equalisation while equalising the channel (7.6) of parameters (7.9).

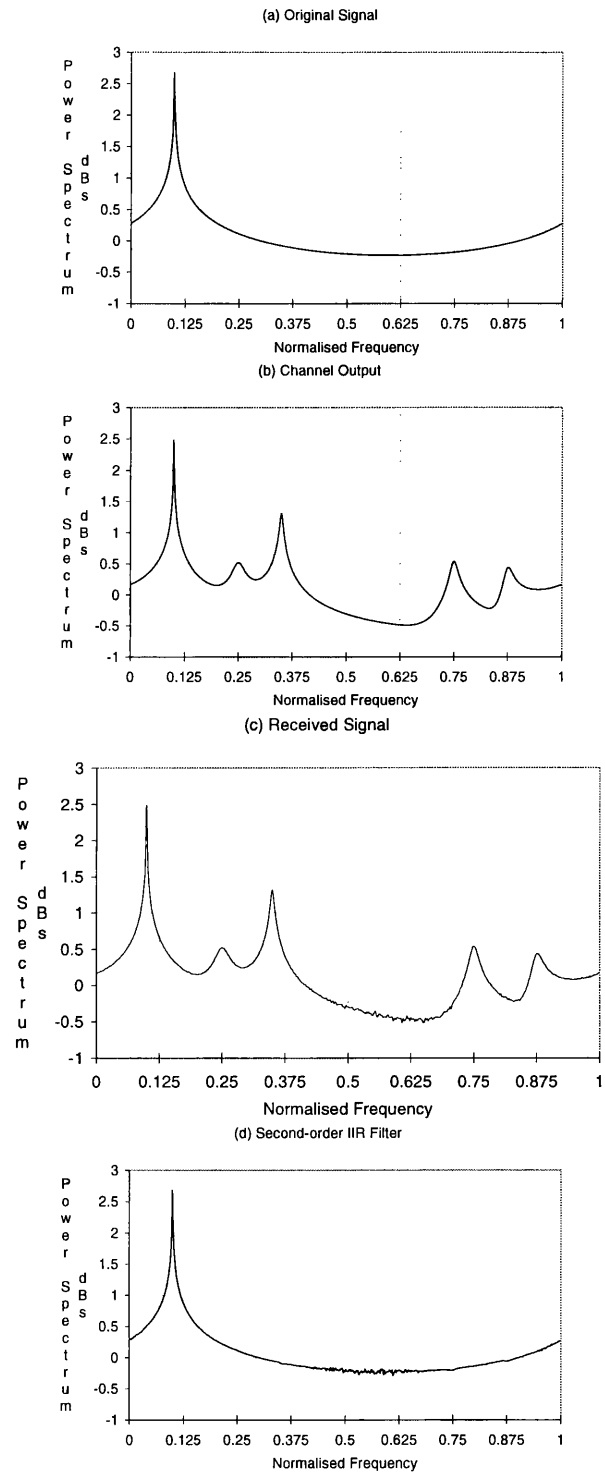

Figure 7.10: Power spectrum of input, corrupted and output of channel equalisation of parameters (7.16). (a) Orignal signal. (b) Channel output. (c) Received signal contaminated in white noise. (d) Recovered signal by a second-order IIR filter optimised through the hybrid approach.

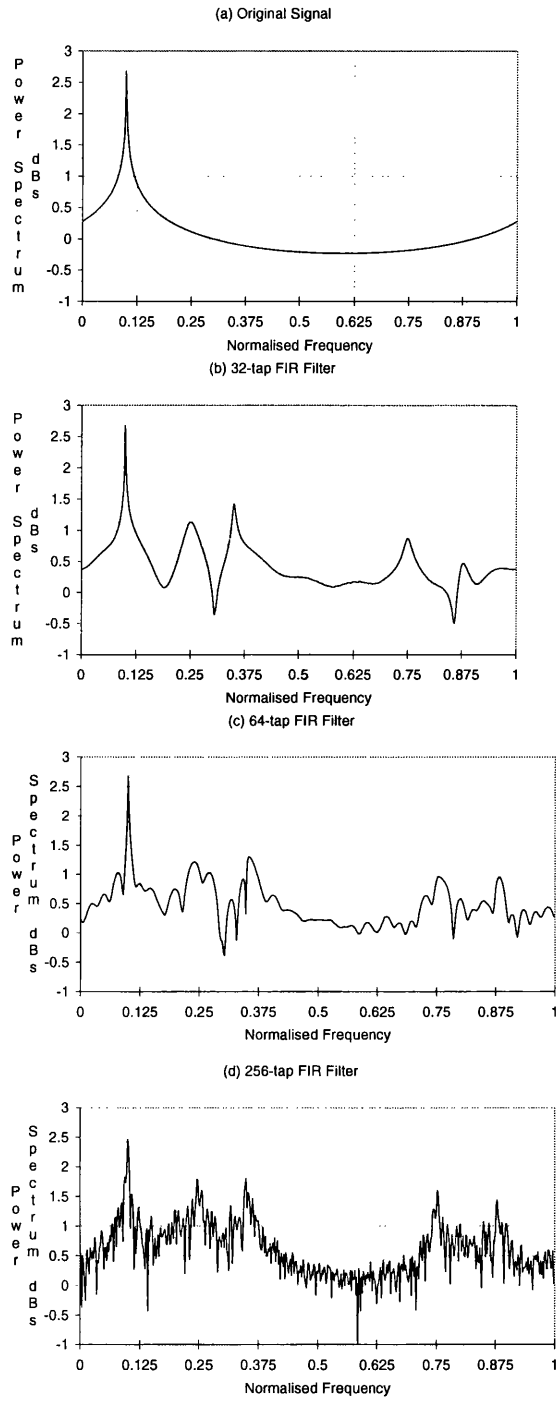

Figure 7.11: Power spectrum of signals. Original and recovered signals from FIR equalisers of various filter lengths when equalising the channel's parameters (7.16). (a) Original signal. (b) Recovered signal by 32-tap FIR filter. (c) Recovered signal by 64-tap FIR filter. (d) Recovered signal by 256-tap FIR filter.

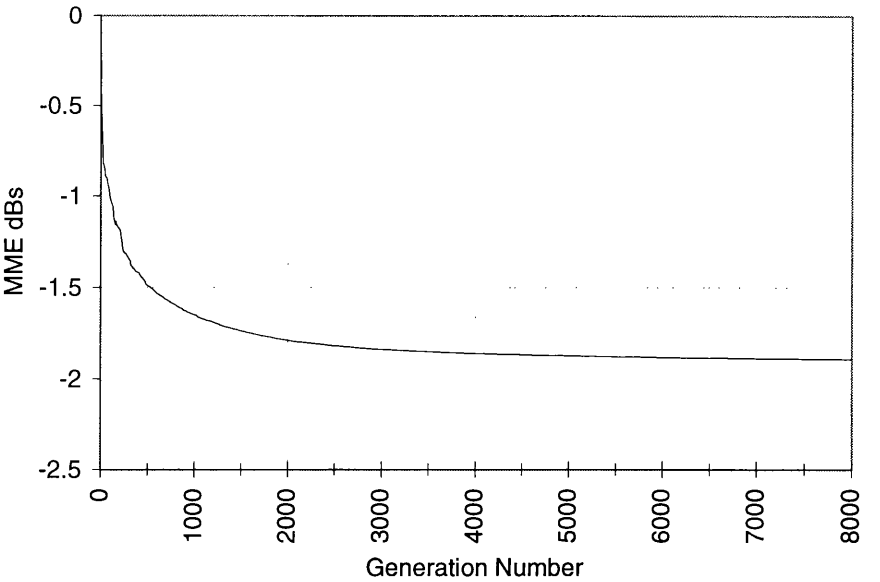

Figure 7.12: Learning curves showing the MME values plotted against generation number while optimising the IIR filters for equalising the channel (7.6) of parameters (7.16)

curve, which shows the MME of the best filter plotted against generation number while optimising the adaptive filters. Again, hybrid algorithm identifies the final solution within 4000 generations.

Along with the above results, we also compare the BER performance between IIR and FIR equalisers when transmission of binary signals through two simple channels:

$$C_1(z) = 1 + 0.7z^{-1} \tag{7.17}$$

$$C_2(z) = 1 + 0.95z^{-1} \tag{7.18}$$

For this experiment, a pseudo random binary sequence (PRBS) is applied as an original signal and BERs is obtained to measure the performance of the filters. Appendix D gives full details of calculating BER from the input and output signals. The binary sequence considered in this chapter is assumed to belong to the alphabets PRBS $\in \{-1, +1\}$. Last 100 samples of inputs and the corrupted signals of the above channels are shown in Figure 7.13. It is clearly seen from the figure that the error introduced by the channel, $1 + 0.95z^{-1}$, is greater than those of $1+0.7z^{-1}$. This is because, when the zeros are closer to the unit circle, it

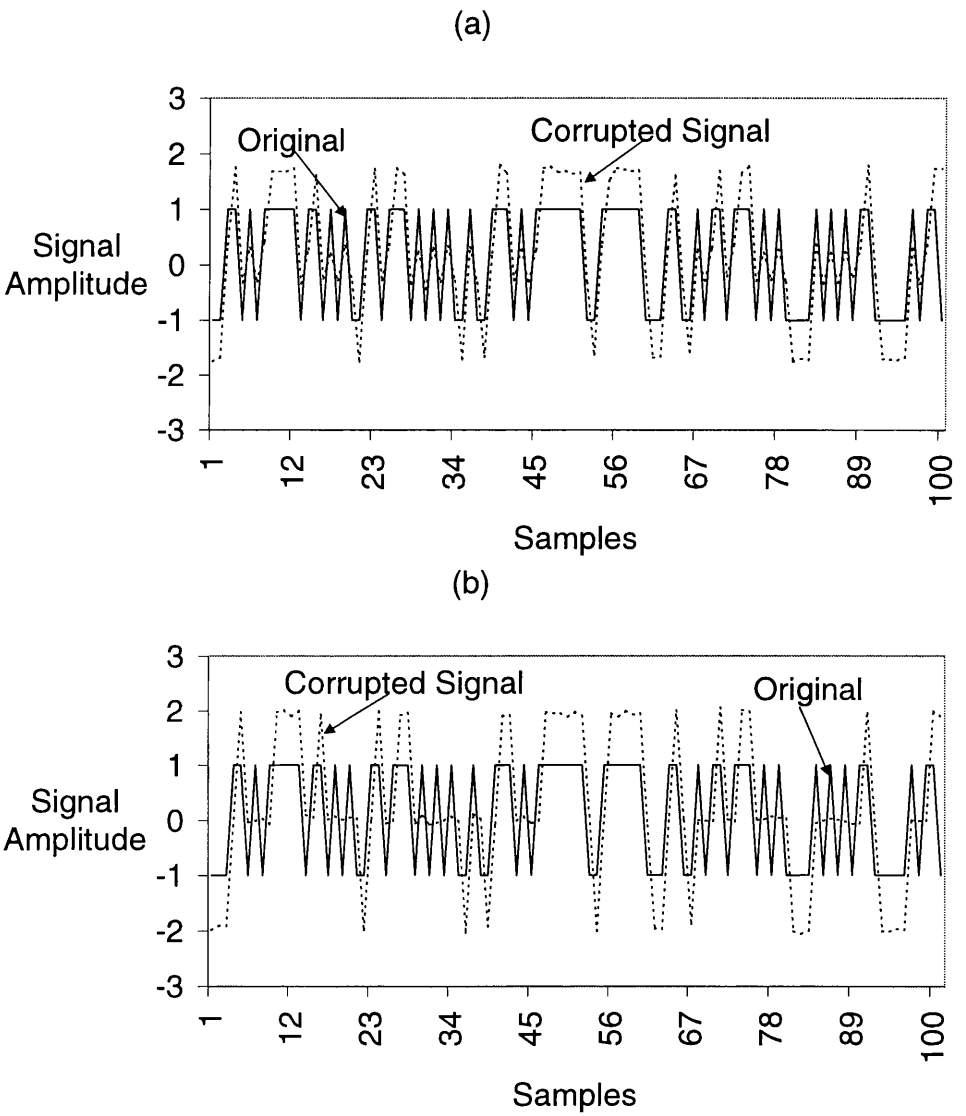

Figure 7.13: Input and corrupted signals when equalising the channels, (a) $c_1(z) = 1 + 0.7z^{-1}$ and (b) $c_2(z) = 1 + 0.95z^{-1}$, with SNR = 25 dB

introduces spectral nulls in the passband, therefore it is difficult to equalise these distortions by conventional FIR filters [28]. The power spectrums of the signals which are obtained from these two channels are shown in Figure 7.14. The channel $1+0.95z^{-1}$ introduces negative spikes in the passband, but the channel $1+0.7z^{-1}$ doesn't produce such negative spikes. A finite impulse response filter with 14-taps is used along with the LMS algorithm to model the inverse of the channels (7.17) and (7.18) respectively. A population of adaptive IIR filters of second-order is used with the floating-point EA. Figure 7.15 presents the BER comparisons of the optimum filters for various SNRs. This comparative results show that the BER achieved by both equalisers for the channel $1+0.7z^{-1}$ has very little difference. The IIR filter gives slightly better performance than a FIR equaliser does. However, the results achieved for $1+0.95z^{-1}$, which has a zero very close to the unit circle, gives confidence in the IIR approach. Even a 14-tap FIR equaliser is not able to perform as well as equalisation as a second-order IIR equaliser.

7.2.3 Blind Channel Equalisation

A major problem inherent in line of sight communication is that of changes of channel characteristics due to variations of atmospheric condition. The channel characteristics can be changed with time due to variations in the refractive index of the atmosphere. The signals transmitted through these channels are nonstationary and will necessitate the use of online adaptation, which equalise the channels as new data being received. Moreover, the receiver has no exact knowledge of the transmitted signal. Therefore trained adaptation cannot be used as in the case of MODEM channels. In such a case blind equalisation is widely used [76].

Blind equalisation or blind deconvolution is an adaptive inverse filtering technique where the adaptive algorithm has no access to training or desired signal. It is a self-learning technique, which can restore a signal corrupted by channels back to original condition. In blind equalisers, a cost function is

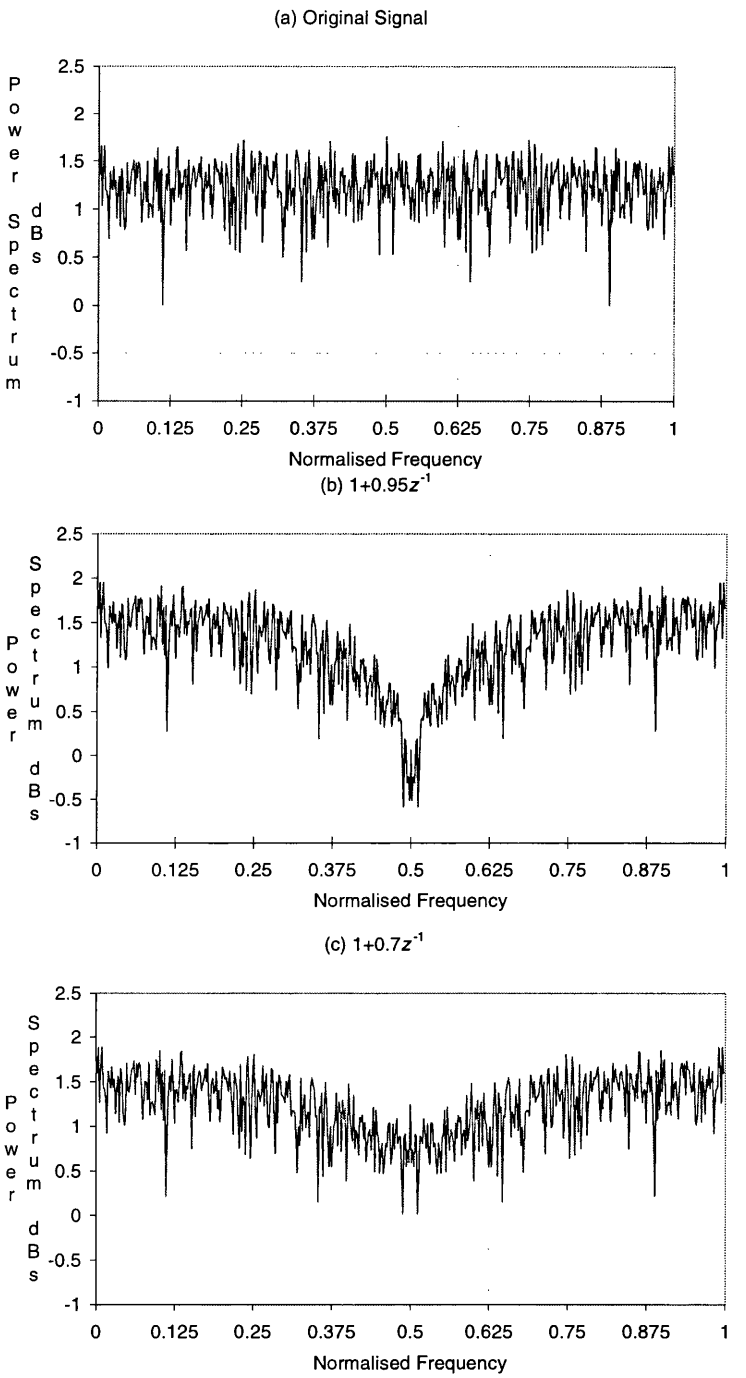

Figure 7.14: Power spectrum of the input and corrupted signals: (a) input signal (b) corrupted signal by $c_1(z) = 1 + 0.95z^{-1}$ (b) corrupted signal by $c_2(z) = 1 + 0.7z^{-1}$, with a SNR, SNR = 25 dB

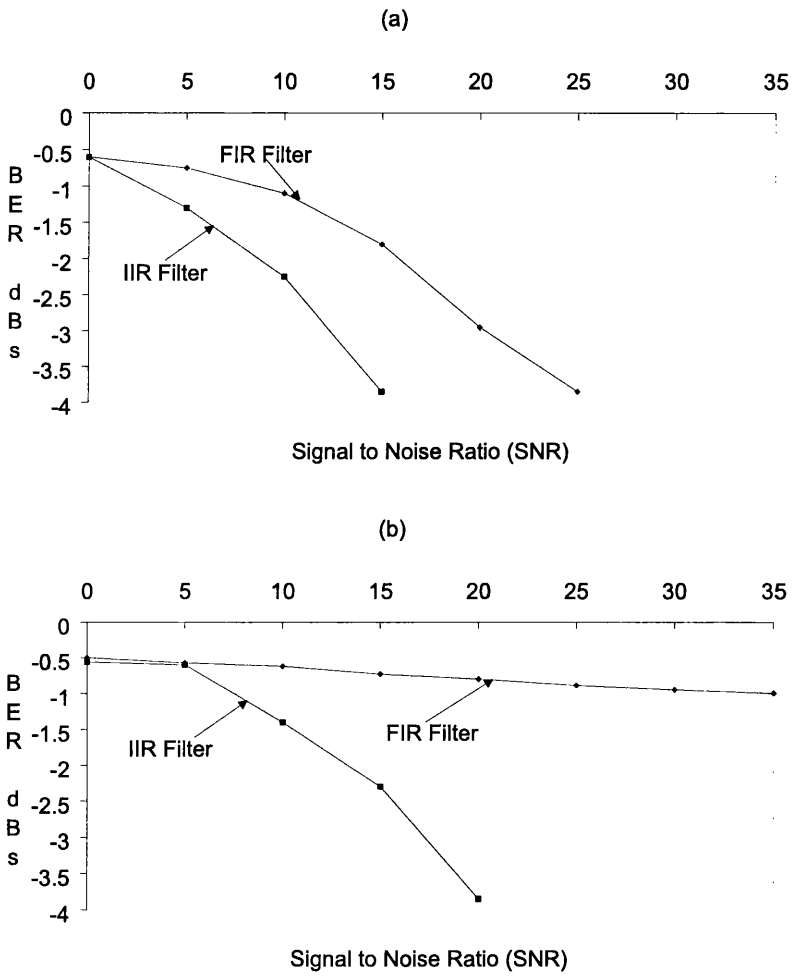

Figure 7.15: Bit error rate comparison of adaptive filters. (a) BER comparison between a 14-tap FIR equaliser and a second-order IIR equaliser when equalising the communication channel $C_1(z) = 1 + 0.7z^{-1}$. (b) BER comparison between a 14-tap FIR equaliser and a second-order IIR equaliser when equalising the communication channel $c_2(z) = 1 + 0.95z^{-1}$.

minimised in much the same way that least mean square and recursive least square adaptation minimise the MSE. One of the most popular algorithms is constant modulus algorithm (CMA).

Constant modulus algorithm is designed to penalise deviations of the blind equaliser output from a constant modulus. The cost function that is minimised via CMA is

$$p = \mathrm{E}[(|y(n)|^2 - R^2)^2] \tag{7.19}$$

where $y(n)$ is output while R represents a constant modulus. The constant R is chosen in such a way that the gradient of the cost function p is zero when perfect equalisation is attained [28]:

$$R = \frac{\mathrm{E}[|\hat{s}(n)|^2]}{\mathrm{E}[|\hat{s}(n)|]} \tag{7.20}$$

where $\hat{s}(n)$ is an estimate of the input signal $s(n)$.

Thus CMA can be applied to those areas where the signal of interest contains a constant envelope property. Examples of such signal types are frequency modulation (FM), phase modulation (PM), double side band amplitude modulation (DSBAM), etc. The gradient descent version of CMA was successfully applied to adaptive FIR filters and is well defined in [76]. A simple multipath channel, which is more likely to be encountered in practical communication systems, is given by

$$c_1(n) = \delta(n) + 0.5\delta(n-1) + 0.1\delta(n-2) + v(n) \tag{7.21}$$

where $c_1(n)$ represents channel impulse response, $\delta(n)$ represents the input and $v(n)$ denotes the additive Gaussian noise with zero mean and σ as the standard deviation. In our simulations standard deviation is arranged so as to meet signal to noise ratio of 25 dB. The transmitted signals used in the following simulations is assumed to be pseudo random binary taking the values $\{-1, 1\}$. For PRBS, it can be easily shown that $\mathrm{E}\{s^2(n)\} = 1$ and $\mathrm{E}\{s^4(n)\} = 1$ which in turn gives $R = 1$. We further assume that the channel, (7.21) is nonstationary and their parameters are varied in such a manner

$$c_{n,1}(n) \quad = \quad \delta(n) + \{0.5 - 0.5(1 - \exp(0.0001n))\}\delta(n-1)$$

$$+\{0.1 + 0.5(1 - \exp(0.0001n))\}\delta(n-2) + v(n) \qquad (7.22)$$

Figure 7.16 and 7.17 shows the trajectories of $a_1(n)$ and $a_2(n)$ while equalising the above channel, (7.22), by a second-order adaptive filter of the following structure:

$$A_1(z) = \frac{b_0}{1 - a_1 z^{-1} - a_2 z^{-2}} \qquad (7.23)$$

using normalised RPE algorithm. Before the normalised RPE is switched on, the best member of a population for a particular training set is found using hybrid methodology described above. From the coefficient trajectories shown,

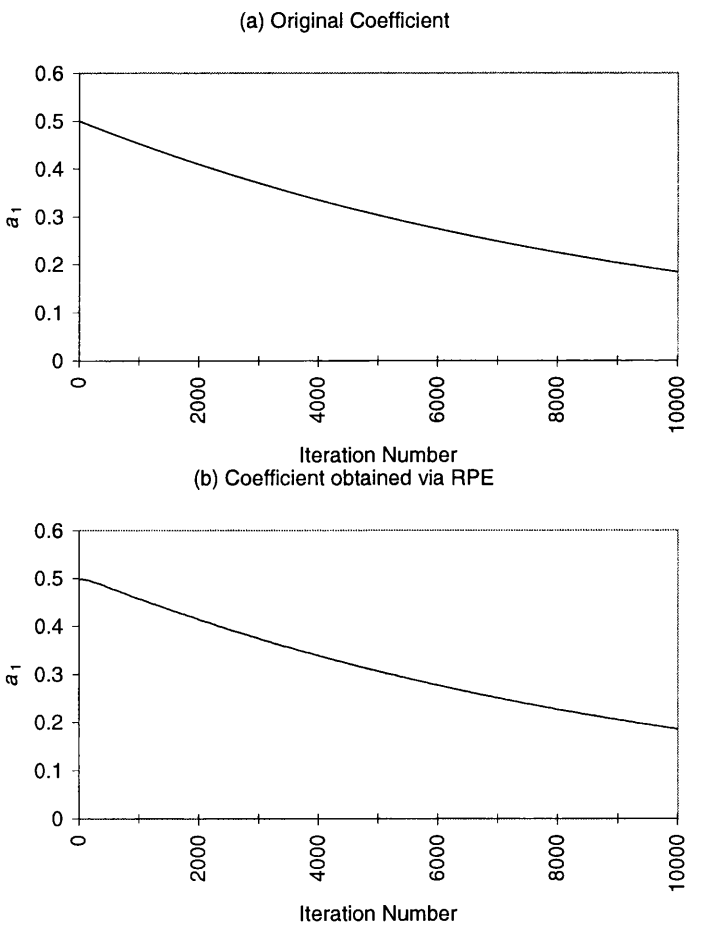

Figure 7.16: Trajectory of a_1 while equalising the channel (7.22) with an adaptive filter of structure (7.23)

it is clear that equalisation of channels in nonstationary environments can be easily achieved with the recently proposed approach. Figure 7.18, 7.19 and 7.20

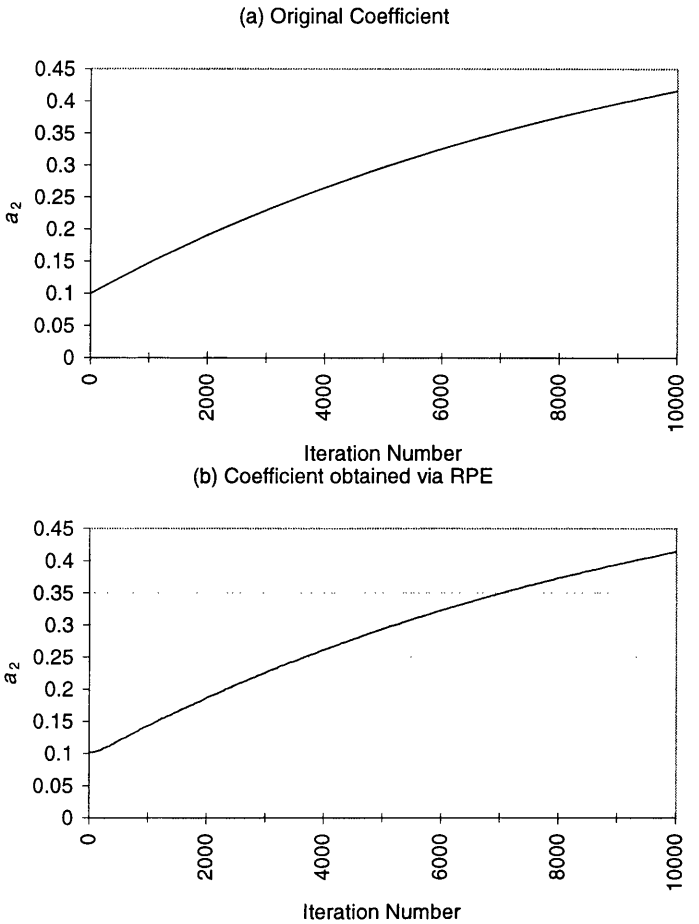

Figure 7.17: Trajectory of a_2 while equalising the channel (7.22) with an adaptive filter of structure (7.23)

shows the coefficient trajectories while equalising the following channel

$$\begin{aligned} c_{n,2}(n) &= \delta(n) + \{1.2 - 0.15(1 - \exp(0.0001n))\}\delta(n-1) \\ &\quad +\{-0.15 + 0.15(1 - \exp(0.0001n))\}\delta(n-2) \\ &\quad -0.586 + \{-0.1682 + 0.15(1 - \exp(0.0001n))\}\delta(n-4) \end{aligned} \quad (7.24)$$

with the following adaptive filter

$$A_2(z) = \frac{b_0}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}} \quad (7.25)$$

by using normalised RPE algorithm.

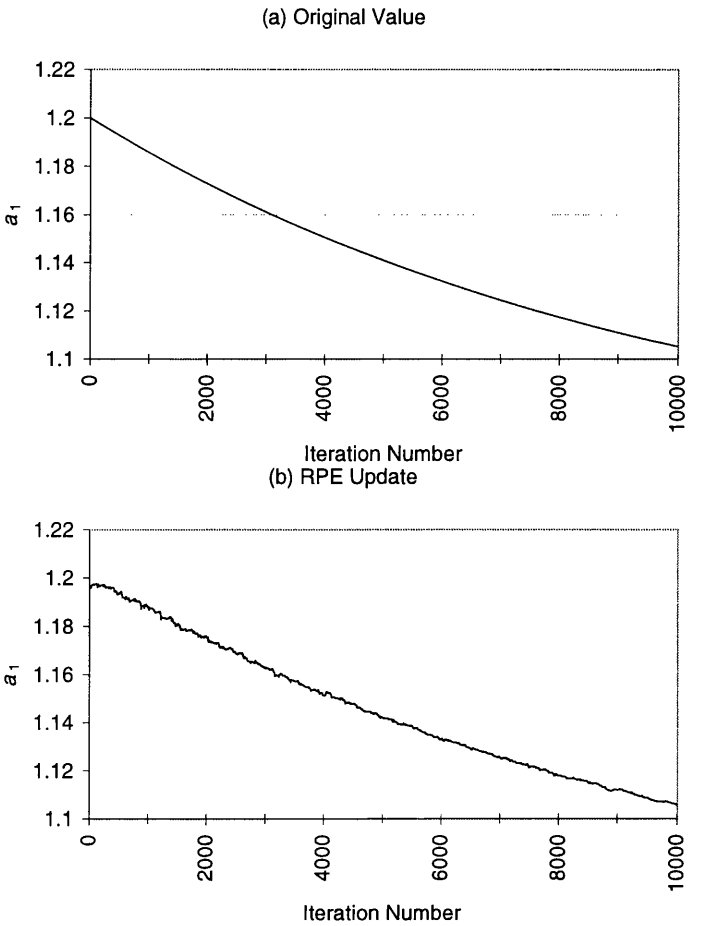

Figure 7.18: Trajectory of a_1 while equalising the channel (7.24) with an adaptive filter of structure (7.25)

7.3 Noise Cancelling

The main objective in noise cancelling is to produce an optimum estimate of the noise in the corrupted signals and hence an optimum estimate of the desired signal. A more general diagram for an adaptive noise cancelling system is shown in Figure 7.21. The signal $s(n)$ is the corrupted signal containing both the desired signal, $o(n)$, and the noise, $v(n)$, assumed to be uncorrelated with each other. The signal, $x(n)$ is a measure of the noisy signal which is correlated in some way with $v(n)$. The signal $x(n)$ is processed by the digital filter to produce an estimate, $\hat{v}(n)$, of $v(n)$. An estimate of the desired signal is then obtained by subtracting the digital filter output, $\hat{v}(n)$, from the corrupted signal, $s(n)$:

$$\hat{o}(n) = s(n) - \hat{v}(n)$$

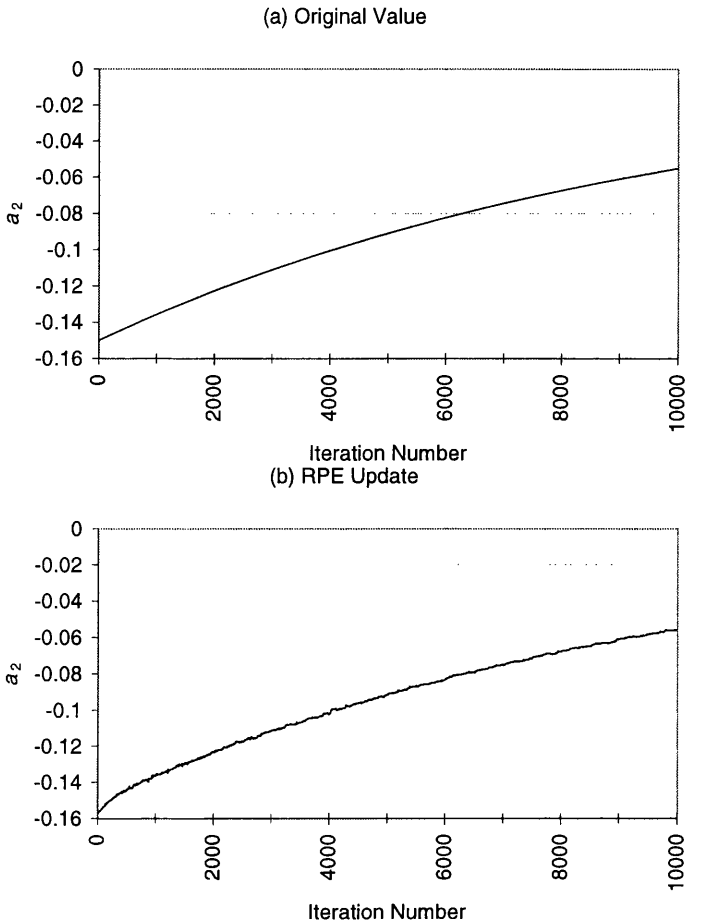

Figure 7.19: Trajectory of a_2 while equalising the channel (7.24) with an adaptive filter of structure (7.25)

$$= \quad o(n) + v(n) - \hat{v}(n) \tag{7.26}$$

As shown in Figure 7.21 noise cancelling is achieved by using $\hat{o}(n)$ in the feedback arrangement to adjust the digital filter coefficients, via a suitable algorithm, to minimise the noise in $\hat{o}(n)$. The output signal, $\hat{o}(n)$, serves two purposes:

1. As an estimate of the desired signal and

2. As an error signal which is used to adjust the filter coefficients.

It can be easily shown that the total power at the output of the cancellor maximise the output signal to noise ratio [82, 81]. For example, consider the estimate of the desired signal $\hat{o}(n)$ which is given in equation (7.26). Squaring

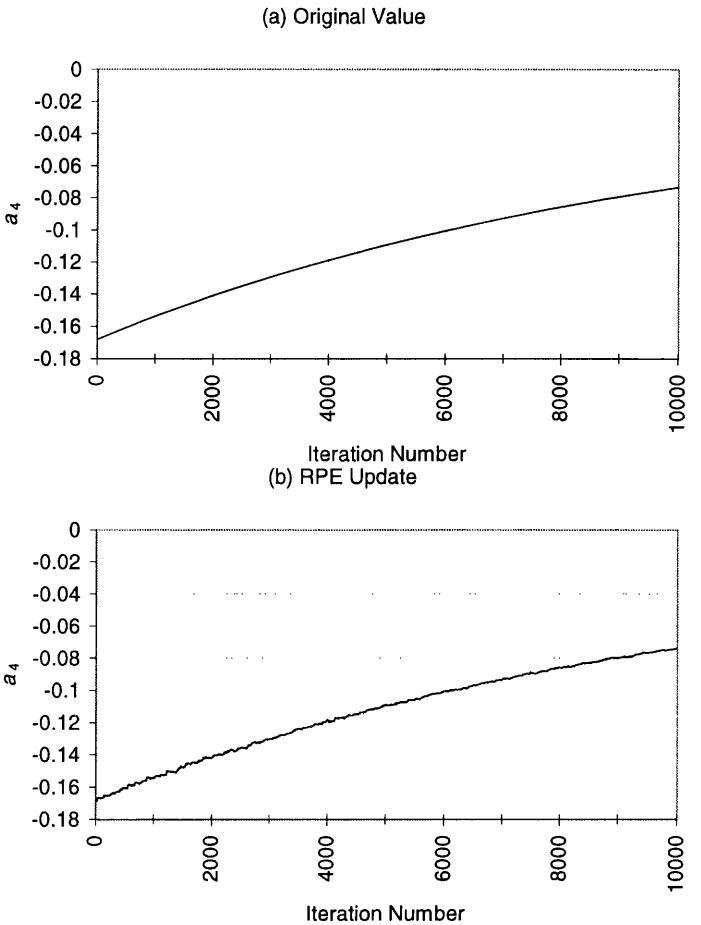

Figure 7.20: Trajectory of a_4 while equalising the channel (7.24) with an adaptive filter of structure (7.25)

equation (7.26) we have

$$\begin{aligned} \hat{o}^2(n) &= o^2(n) + \{v(n) - \hat{v}(n)\}^2 \\ &\quad +2o(n)\{v(n) - \hat{v}(n)\} \end{aligned} \tag{7.27}$$

Taking the expectation of both sides of equation (7.27) we have

$$\begin{aligned} \mathrm{E}\left[\hat{o}^2(n)\right] &= \mathrm{E}\left[o^2(n)\right] + \mathrm{E}\left[\{v(n) - \hat{v}(n)\}^2\right] \\ &\quad +2\mathrm{E}\left[o(n)\{v(n) - \hat{v}(n)\}\right] \end{aligned} \tag{7.28}$$

Since the desired signal, $o(n)$, is uncorrelated with $v(n)$ or with $\hat{v}(n)$ the last term in equation (7.28) is zero and we have

$$\mathrm{E}\left[\hat{o}^2(n)\right] = \mathrm{E}\left[o^2(n)\right] + \mathrm{E}\left[(v(n) - \hat{v}(n))^2\right] \tag{7.29}$$

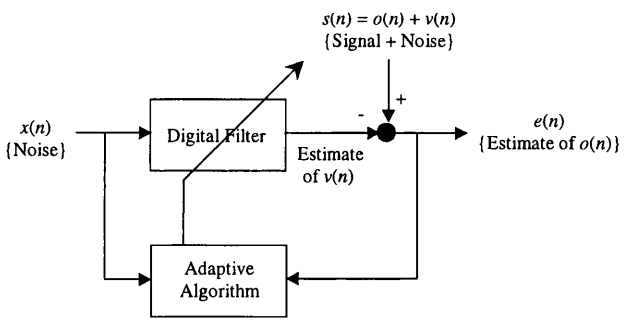

Figure 7.21: A more general diagram for adaptive noise cancelling.

where E $[o^2(n)]$ represents the total signal power, while $E[\hat{o}^2(n)]$ represents the estimate of the signal power. It is evident in the above equation that if the estimate $\hat{v}(n)$ is the exact replica of $v(n)$, the output power will contain only the signal power:

$$\min \mathrm{E}\left[\hat{o}^2(n)\right] = \mathrm{E}\left[o^2(n)\right] + \min \mathrm{E}\left[(v(n) - \hat{v}(n))^2\right] \tag{7.30}$$

Therefore it is clear that the net effect of minimising the total output power is to maximise the output SNR.

Results and Validation

Let us consider the following noise cancelling diagram shown in Figure 7.22. Here $B(z)$ and $\{1-A(z)\}$ are the noise coloration filters which may be described by

$$A(z) = \{1 - a_1 z^{-1} - a_2 z^{-2}, \cdots, -a_L z^{-L}\} \tag{7.31}$$

$$B(z) = \{b_0 - b_1 z^{-1}, \cdots, +b_{M-1} z^{-(M-1)}\} \tag{7.32}$$

We assume that two measurements are available,

1. Signal to be extracted plus noise

2. an auxiliary measurement, u, which is correlated with the noise $w(n)$.

We also assume that the additive noise contaminating the signal is a filtered version (filtered through $B(z)$) of a white noise $\{w(n)\}$ and that the measured

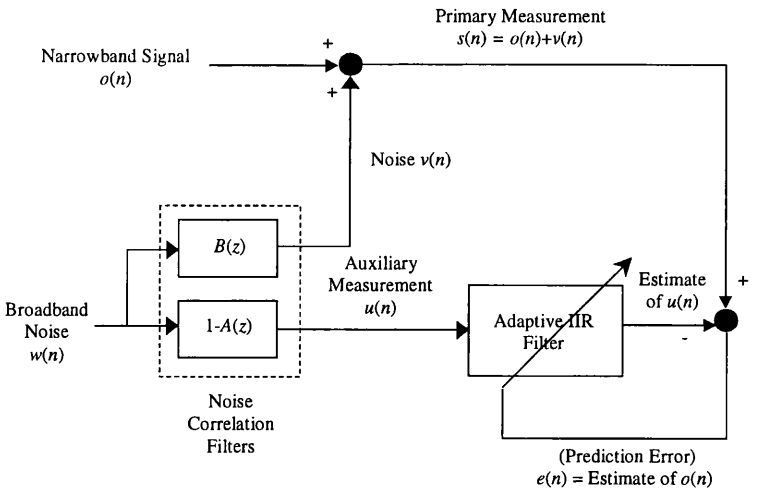

Figure 7.22: An experimental arrangement of noise cancellation.

correlated noise is the same $\{w(n)\}$ passed through different coloration filter, $\{1 - A(z)\}$. Let us consider the impulse responses of the coloration filters, $B(z)$ and $\{1 - A(z)\}$ are $b(n)$ and $a(n)$ respectively. For the sake of simplicity, the noise coloration filters used in this experiment are assumed to be FIR.

Our aim is to minimise MME:

$$\min\{\varepsilon\} = \min \mathrm{E}\{e(n)\} = \min \mathrm{E}\{o(n) + v(n) - f(n)\} \tag{7.33}$$

using recently proposed hybrid techniques, where $f(n)$ denotes the estimate of the filtered noise $v(n)$. The equation given in (7.33) can be expanded in such a way that:

$$\begin{aligned} \min\{\varepsilon\} &= \min \mathrm{E}\{w(n) * b(n) + o(n) - w(n) * a(n) * f(n)\} \\ &= \min \mathrm{E}\{o(n)\} + \min \mathrm{E}\{w(n) * [b(n) - a(n) * f(n)]\} \end{aligned} \tag{7.34}$$

Thus minimising the MME in the equation (7.34) will actually minimise the last term of the equation. Therefore, it is evident that the error can be minimised to zero if

$$b(n) - a(n) * f(n) = 0 \tag{7.35}$$

If the transfer function of the estimated adaptive filter, $f(n)$ is $F(z)$, the equation (7.35) approaches zero when

$$F(z) = \frac{B(z)}{1 - A(z)} \tag{7.36}$$

The equation (7.36) clearly illustrates the necessity of adaptive IIR filters in the application noise cancellation.

A PRBS signal

$$\text{PRBS} \in \{-1, +1\} \tag{7.37}$$

was used as an original data sequence and the noise filters are assumed to be of the form

$$B(z) = 0.5 + 0.6z^{-1} - 0.4z^{-2} \tag{7.38}$$

and

$$A(z) = 0.4z^{-1} - 0.5z^{-2} \tag{7.39}$$

The reference noise is a set of Gaussian random variables with zero mean and some standard deviation so as to meet the signal to noise ratio, SNR = 10 dB. Samples of 10000 data, $\{o(n), w(n)\}_{n=0}^{9999}$ are generated. Among 10000 samples only 100 samples are taken to train the model using adaptive algorithm as shown in Figure 7.22. It is evident from Figure 7.23(b) that all of the signal information of the original binary sequence cannot be determined from the corrupted signal. The floating-point EA with following set of parameters:

$$\{\alpha_c = 0.9, \alpha_m = 0.02, \alpha_i = 0.05, \alpha_n = 10\} \tag{7.40}$$

is used to evolve a population of IIR filters, each having 3 zeros and 2 poles. The algorithm is allowed to run over 4000 generations. The results of noise cancelling are illustrated in Figure 7.23. From the results it can be seen that the original binary symbols which buried in noisy environment is perfectly identified. i.e. noise has been significantly reduced.

7.4 Summary and Discussion

In this chapter we have shown two major applications of adaptive IIR filtering. These include, Channel Equalisation and Noise Cancellation.

In channel equalisation, the concepts of blind and trained adaptation techniques have been revealed. Firstly, equalisation of complex channel responses has been shown, in which the recently proposed hybrid technique was employed to obtain the best-estimated equaliser using trained adaptation. The power spectrum of the equalised signals obtained through IIR filters were compared with those of FIR filters adapted through standard LMS algorithm. Moreover, by applying the PRBS as an input, BER performances of FIR and IIR filters were compared when equalising the channels, $1 + 0.7z^{-1}$ and $1 + 0.95z^{-1}$ with various SNRs. From these results it can be concluded that a FIR equaliser needs much more coefficients than an IIR equaliser to achieve a performance as good as an IIR equaliser does. However, it is also evident from these results that equalisation of channels whose zeros are very close to the unit circle is very difficult to equalise by a FIR equaliser even with a large number of coefficients. These results clearly show the necessity of adaptive IIR filtering techniques when equalising communication channels.

A blind algorithm was employed in our simulation as a constant modulus algorithm, which perform adjustments of the adaptive filter coefficients without the need for a desired response. We have shown that direct form adaptive IIR filters can be successfully used to identify nonstationary channels with the use of the proposed hybrid methodology.

We have shown that equalisation of communication channels in noisy environments can be achieved through linear adaptive filtering. The technique used by linear adaptive filtering approach is inverse filtering method that cannot remove the noise interference. i.e. linear adaptive filters can only be used to identify the channel interference, but not to remove the noise interference. The removal of noise interference requires the use nonlinear filtering techniques

and may be achieved by adding a nonlinear function at the output of the filter. Another observation is that IIR filters give much better frequency response than equivalent FIR filters. The results confirming this statement are shown in Figures 7.5 - 7.10 which were obtained when equalising various complex IIR channels using trained adaptation. Furthermore, IIR equalisers gave better BER performance than FIR equalisers did. Bit error rates obtained against various signal to noise ratios when equalising received binary signals of the channels $1+0.7z^{-1}$ and $1+0.95z^{-1}$ through IIR and FIR equalisers are shown in Figures 7.15. From these results we also concluded that BER performance obtained for the channel whose zeros close to the unit circle (e.g. $1+0.95z^{-1}$) is worse than those of zeros well inside the unit circle (e.g. $1+0.7z^{-1}$). However, the IIR filters able to equalise these close poles with less number of coefficients than FIR filters.

Finally, concept of adaptive noise cancelling has been described. We have shown the necessity of adaptive IIR filtering. The critical characteristic is that the relationship between the auxiliary measurement, u, and the primary measurement, s, of Figure 7.22 are described by an IIR transfer function shown by the equation (7.36). A population of adaptive IIR filters was used with a floating-point evolutionary algorithm to obtain an estimate of the original signal, which is corrupted by noise interference. The recovered signal has been compared with the original sequence and this result clearly shows the ability of an adaptive IIR filter to cancel the noise interference.

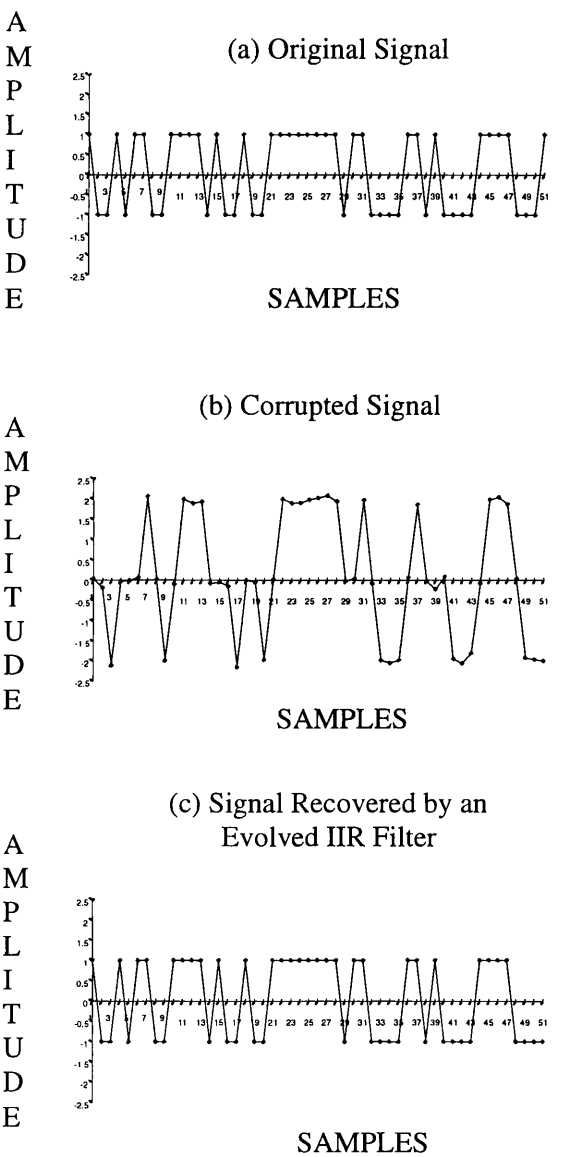

Figure 7.23: The results of noise cancellation.

Chapter 8

Conclusion and Further Work

8.1 Conclusion

A critical review related to the importance of this research has been outlined in Chapter 1. In particular, limitations of current design methods of IIR adaptive filters have been investigated. Chapters 2 and 3 have provided a theoretical background of digital filter design with various conventional and evolutionary algorithms. These have led to the development of a pole and coefficient design approach to IIR, adaptive and complex digital filters and the development of a globally optimal, fine-tuned and efficient evolutionary design methodology.

In Chapter 4, the pole based design technique has been developed. This has avoided the stability-monitoring problem by constraining the search space of poles within a known stable region. This method has also overcome the major drawback encountered by conventional design methods arising from multiple optima in the design space. Evolutionary algorithms have been developed to represent complex poles in the chromosome structures in the same manner as they appear in the filter structures and hence the coding/ decoding process has been avoided. Split-point crossover which combines uniquely as indivisible floating-point genes have been developed for combining complex genes and demonstrated as being a better method when compared to standard crossover operations. This crossover also increased the degree of population diversity by

introducing more new members.

Learning curves have been generated to illustrate the convergence behaviour of Gaussian and Uniform distribution of mutations. Gaussian distribution has been shown as being a better mutation distribution for perturbing floating-point genes. Convergence curves illustrating this performance have been obtained while modelling various IIR systems. Furthermore, advantages of tournament among other EA selection schemes have been revealed more clearly. In this work, the cost functions such as mean square error and mean modulus error have been successfully employed for modelling direct and inverse systems. The final parameter values as obtained from these cost functions have been compared with the original values. From these results and from the corresponding learning curves it has been proven that the improved EA can work with any cost functions with ease. This brings a major turning point to hardware designer where implementation of MME in hardware is cost-effectiveness when compared to mean square error.

In Chapter 5, the direct coefficient design method has been developed. The major difficulty of ensuring filter stability during evolution has been overcome. Alternative realisations such as parallel, cascade, lattice structures have also been formulated to simplify stability monitoring. Learning curves while modelling various IIR systems have been generated for each realisation and have been compared with those obtained from direct form filters. These comparative results have shown that rate of convergence of parallel, cascade and lattice form filters were much slower than that of direct form filters. However, the convergence rate of lattice filters was better than parallel and cascade forms. The reason for these variations in the rate of convergence is that, for example, parallel and cascade forms provide multiple optima with the same fitness values, which arises when rearranging the subsections among the filter structures.

A novel method has been introduced to simplify stability monitoring in direct form evolution of IIR filters. This method uses a termination factor that decides the filter stability by comparing this factor with the output calculated

during evaluation. A new correction mechanism has been shown in which unstable filters have been replaced by a healthy parental filter of the previous generation. The learning curves as obtained from this correction mechanism have been compared with standard correction methods and the results illustrated the advantages. This method has increased the rate of convergence and provided final MSE values that are much lower than that of standard approaches.

A new approach has been presented to avoid the degree of premature convergence while evolving the filters using floating-point EAs. This is achieved by introducing a new operator with an aim to increase the population diversity. The total number of chromosomes introduced into the population at each generation is controlled by a factor called probability of immigration α_i. Too small or too large values of α_i will degrade the convergence performance, where the choice of this parameter was more crucial.

Two major drawbacks of evolutionary techniques are (1) poor fine-tuning performance and (2) unsuitability for applications requiring online adaptation. Therefore, in Chapter 6, a hybrid methodology has been developed to improve the local tuning and also to track time varying parameters of nonstationary systems. It is shown how an evolutionary algorithm can be combined with a fast classical algorithm to achieve both goals, where the evolutionary algorithm is used to globalise the parameters while LMS to achieve fine-tuning. This approach is validated by a variety of IIR linear models with distinct pole locations. Various nonstationary systems have been identified with this hybrid approach where the evolutionary algorithm is first employed to ensure a global optimality and the LMS based algorithm to track time varying changes of the nonstationary system. Results show the dependency of adaptive coefficients as fast as parameter variations.

In Chapter 7, two major applications, which are predominantly in the area of adaptive signal processing, have been shown. They include channel equalisation and noise cancellation. Two types of channel equalisers, namely trained equalisers and blind equalisers, are developed for online applications. Trained

equalisation first represents removal of channel interference and additive noise, which appears in MODEM channels that, carries QAM signals. The concept of complex filtering technique has been shown to simplify the process trivial where QAM signals have been treated as complex time-waveform. The Hybrid methodology has been successfully applied to equalise those channels and produce the equalised signals for comparison with FIR filters with various filter lengths. Secondly, trained equalisation of binary signals has been shown to compare BER performance of IIR against FIR filters. Two different channels have been examined, one with zero closer to the unit circle and the other with zero lie well inside the circle. A major observation, was that equalising the channels whose zero closer to the unit circle was only possible with IIR filters. This is because the channels that have zeros closer to the unit circle introduce spectral nulls in the passband and are difficult to equalise by conventional FIR filters.

Application to blind channel equalisation has also been presented, which shows the ability of the proposed technique in nonstationary environments. The blind algorithm employed in our simulation was a constant modulus algorithm, which performed adjustments of the adaptive filter coefficients without the need for a desired response. This work has shown that direct form adaptive IIR filters can be successfully used to identify nonstationary channels with the use of the proposed hybrid methodology.

Finally, the concept of noise cancelling has been shown and the necessity of adaptive IIR filters was revealed clearly. Time waveforms illustrating the capability of noise cancelling have been generated using the proposed adaptive approach.

In summary, all research goals laid out on pages 7 - 8 have been achieved, with other achievements listed on pages 10 - 11.

8.2 Suggestions to Further Research

There are several ways in which this research can go further. A comparative study on binary and floating-point EAs and also a comparison with other coding schemes under the same operating conditions would be useful. Coding a complex parameter into binary is prohibitively long representation and hence requires high computation. However, the rate of convergence may be better than floating-point EAs when the parameters being optimised are complex. Learning curves illustrating the convergence performance would be therefore useful. Also, a study of the Genetic Programming (GP) technique for finding model structures would expand the research discussed in this thesis. A description of this method can be found in Appendix G.

The experiments carried out in this thesis have been developed from C++ code. It is worth to test the algorithms using the ADI real time simulator available in the department. Also, it would be better to investigate the implementation of these algorithms using currently available DSP processors or ASIC design techniques and hence find the issues, which limits those implementations.

The results of channel equalisation illustrated that linear adaptive filtering approach cannot be used in environments where signal to noise ratios are to be very small. In such case, a linear adaptive filter may follow a nonlinear function in order to avoid the noise interference. Also, transmission of binary signals may employ nonlinear classifiers rather than inverse filtering techniques to remove channel interference and noise.

Bibliography

[1] A. Benallal and A. Gillorie. A new method to stabilize fast rls algorithms based on a first-order model of the propagation of numerical errors. In *Proceedings of the International Conference on Acoustics, Speech & Signal Processing, ICASSP-92*, pages 1373–1376, New York, April 1988.

[2] Y. J. Cao and Q. H. Wu. Convergence analysis of adaptive genetic algorithms. In *Proceedings of the Second International Conference on GA in Engineering Systems: Innovations and Application, GALESIA-97*, pages 85–89, University of Strathclyde, Glasgow, September 1997.

[3] Kumar Chellapilla, David B. Fogel, and S S. Rao. Gaining insight into evolutionary programming through landscape visualization: An investigation into IIR filtering. In *Proceedings of the sixth Annual Conference on Evolutionary Programming, EP-97*, pages 364–368, Berlin, Germany, September 1997.

[4] T. Clark and J. S. Mason. Adaptive uniform crossover in genetic algorithms. In *Proceedings of the second IEE International Workshop on Natural Algorithms in Signal Processing*, volume 1, pages 2/1–2/5, Chelmsford, Essex, November 1993.

[5] H. Clergeot. Filter-Order selection in adaptive maximum likelihood estimation. *IEEE Trans. on Information Theory*, IT–30(2):199–210, March 1984.

[6] C. F. N. Cowan and P. M. Grant. *Adaptive Filters.* Prentice Hall, New Jersey, 1985.

[7] P. S. R. Diniz, M. L. R. de Campos, and A. Antoniou. Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor. *IEEE Trans. on Signal Processing*, 43(3):617–627, March 1995.

[8] D. F. Elliot. *Hand Book of Signal Processing.* Rockwell International Corporation, Calfornia, 1987.

[9] L. J. Eshelman and J. D. Schffer. Preventing premature convergence in genetic algorithms by preventing incest. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 115–121, University of California, San Diego, July 1991.

[10] D. M. Etter, M. J. Hicks, and K. H. Cho. Recursive adaptive filter design using an adaptive genetic algorithm. In *Proceedings of the IEEE International Conference on Acoustics Speech & Signal Processing, ICASSP '82*, volume 2, pages 635–638, May 1982.

[11] H. Fan and W. K. Jenkins. A new adaptive IIR filter. *IEEE Trans. on Circuits & Systems*, CAS–33:938–949, 1986.

[12] D. B. Fogel. *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling.* Ginn Press, Needham Heights, MA 02194, 1991.

[13] D. B. Fogel, L. J. Fogel, and W. J. Atmar. Meta-evolutionary programming. In *Proceedings of the 25th Asilomar Conference on Signals, Systems and Computers*, pages 540–545, Pacific Grove, California, 1991.

[14] G. J. Gibson, S. Siu, and C. F. N. Cowan. The application of nonlinear structures to the reconstruction of binary signals. *IEEE Trans. on Signal Processing*, 39(8):1877–1884, August 1991.

[15] I. A. Glover and P. M. Grant. *Digital Communications.* Prentice–Hall, Hertfordshire, 1998.

[16] D. E. Goldberg. *Genetic Algorithms-in Search, Optimization and Machine Learning.* Addison–Wesley, New York, 1989.

[17] D. E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and block. Research Report 90001, Department of General Engineering, University of Illinois, 1990.

[18] G. J. Gray, D. J. Murray-Smith, Y. Li, and K. C. Sharman. Nonlinear model structure identification using genetic programming. In *Proceedings of the Second International Conference on GA in Engineering Systems: Innovations and Application, GALESIA-97*, pages 308–313, University of Strathclyde, Glasgow, September 1997.

[19] S. Haykin. *Communication Systems.* John Wiley & Sons, Canada, second edition, 1983.

[20] S. Haykin. Neural networks expand sp's horizons. *IEEE Signal Processing Magazine*, pages 24–49, March 1996.

[21] Simon Haykin. *Adaptive Filter Theory.* Prentice–Hall, Engle-wood Cliffs, New Jersey, second edition, 1997.

[22] Simon Haykin. Adaptive filters. *IEEE Signal Processing Magazine*, 16(1):20–22, January 1999.

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

[24] E. C. Ifeachor and B. W. Jervis. *Digital Signal Processing.* Addison–Wesley Longman Ltd., England, sixth edition, 1998.

[25] C. Z. Janikow and Z. Michaniewicz. An experimental comparison of binary and floating point representation in genetic algorithms. In *Proceedings of*

the Fourth International Conference on Genetic Algorithms, pages 31–36, University of California, San Diego, July 1991.

[26] C. R. Johnson. Adaptive IIR filtering: Current results and open issues. *IEEE Trans. on Information Theory*, IT–30(2):237–249, March 1984.

[27] E. I. Jury. *Theory and Applications of the Z-Transform Method.* Wiley, New York, 1964.

[28] G. Kechriotis, E. Zervas, and E. S. Manolakis. Using recurrent neural networks for adaptive communications channel equalisation. *IEEE Trans. on Neural Networks*, 5(2):267–278, March 1994.

[29] J. B. Kenney and C. E. Rohrs. The composite regressor algorithm for iir adaptive systems. *IEEE Trans. on Signal Processing*, 41(2):617–627, Febrauary 1993.

[30] J. B. Kenney and C. E. Rohrs. The composite regressor algorithm for iir adaptive systems. *IEEE Trans. on Signal Processing*, 43(3):720–729, March 1995.

[31] K. Kurosawa and S. Tsujii. An IIR parallel-type adaptive algorithm using the fast least squares method. *IEEE Trans. on Acoustics, Speech & Signal Processing*, 37(8):1226–1230, August 1989.

[32] M. Lang and B. C. Frenzel. Polynomial root finding. Research Report EDICS: SPL 4.1, Electrical and Computer Engineering - MS 366, Rice University, 1994.

[33] M. G. Larimore, J. R. Treichler, C. R. Johnson, and JR. Sharf: An algorithm for adapting IIR digital filters. *IEEE Trans. on Acoustics, Speech & Signal Processing*, ASSP–28(4):428–440, August 1980.

[34] Y. Leung, Y. Gao, and Zong-Ben Xu. Degree of population diversity-a perspective on premature convergence in genetic algorithms and its markov

chain analysis. *IEEE Trans. on Neural Networks*, 8(5):1165–1176, September 1997.

[35] Y. Li, K. C. Tan, and M. Gong. Global structure evolution and local parameter learning for control system model reductions. *Evolutionary Algorithms in Engineering Applications*, pages 345–360, January 1997.

[36] L. Ljung, M. Morf, and D. Falconer. Fast calculation of gain matrices for recursive estimation schemes. *International Journal of Control*, 27:1–19, January 1978.

[37] Q. Ma and C. F. N Cowan. Genetic algorithms applied to the adaptation of IIR filters. *IEEE Trans. on Signal Processing*, 48:155–163, 1996.

[38] J. R. McDonnell. Evolving recurrent perceptrons for Time-Series modelling. *IEEE Trans. on Neural Networks*, 5(1):24–38, January 1994.

[39] E. W. McGookin, D. J. Murray-Smith, and Y. Li. A population minimisation process for genetic algorithms and its applications to controller optimisation. In *Proceedings of the Second International Conference on GA in Engineering Systems: Innovations and Application, GALESIA-97*, pages 79–84, University of Strathclyde, Glasgow, September 1997.

[40] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effect of noise. IlliGAL Report 95006, Department of General Engineering, University of Illinois, 1995.

[41] M. Miyoshi and Y. Kaneda. Inverse filtering of room acoustics. *IEEE Trans. on Acoustics, Speech & Signal Processing*, 36(2):145–152, February 1988.

[42] B. Mulgrew. Applying radial basis functions. *IEEE Signal Processing Magazine*, pages 50–65, March 1996.

[43] R. N. Mutagi. Pseudo noise sequences for engineers. *IEE Journal on Electronics & Communications*, 8(2):79–87, April 1996.

[44] R. Nambiar and P. Mars. Genetic algorithms for adaptive digital filtering. In *Proceedings of the IEE Colloquium on Genetic Algorithms for Control Engineering*, pages 160–168, Savoy Place, London, 1992.

[45] R. Nambiar and P. Mars. Genetic and annealing approaches to adaptive digital filtering. In *Proceedings of the 26th Asilomar Conference on Signals, Systems and Computers*, pages 871–890, August 1992.

[46] R. Nambiar and P. Mars. Adaptive IIR filtering using natural algorithms. In *Proceedings of the second IEE International Workshop on Natural Algorithms in Signal Processing*, volume 2, pages 20/1–20/9, Chelmsford, Essex, November 1993.

[47] R. Nambiar, C. K. K. Tang, and P. Mars. Genetic and learning automata algorithms for adaptive digital filters. In *Proceedings of the International Conference on Acoustics, Speech & Signal Processing, ICASSP-92*, volume IV, pages 41–44, San Fransico, California, September 1992.

[48] M. Nayeri and W. K. Jenkins. Analysis of alternate realizations of adaptive IIR filters. In *Proceedings of the IEEE International Symposium on Circuit and Systems*, pages 2157–2160, Espoo, Finland, June 1988.

[49] S. C. Ng, C. Y. Chung, S. H. Leung, and A. Luk. An evolutionary search algorithm for adaptive IIR equalizer. In *Proceedings of the IEEE International Symposium on Circuit and Systems*, volume 2, pages 53–56, London, May 1994.

[50] S. C. Ng, C. Y. Chung, S. H. Leung, and Andrew Luk. Fast convergent genetic search for adaptive IIR filtering. In *Proceedings of the IEEE International Conference on Acoustics Speech & Signal Processing*, volume III, pages 105–108, Adelaide, South Australia, April 1994.

[51] S. C. Ng, S. H. Leung, C. Y. Chung, A. Luk, and W. H. Lau. The genetic search approach. *IEEE Signal Processing Magazine*, pages 38–46, November 1996.

[52] L. J. Nicolson and B. M. G. Cheetham. Simulated annealing applied to the design of IIR digital filters by multiple criterion optimisation. In *Proceedings of the second IEE International Workshop on Natural Algorithms in Signal Processing*, volume 1, pages 6/1–6/7, Chelmsford, Essex, November 1993.

[53] T. Petillon, A. Gilloire, and S. Theodoridis. The fast newton transversal filter: An efficient scheme for acoustic echo cancellation in mobile radio. *IEEE Trans. on Signal Processing*, 42(3):509–517, March 1994.

[54] S. W. Piche. Steepest descent algorithms for neural network controllers and filters. *IEEE Trans. on Neural Networks*, 5(2):198–212, March 1994.

[55] J. G. Prokis and D. G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications.* Macmillan, New York, second edition, 1992.

[56] S. U. H. Qureshi. Adaptive equalization. In *Proceedings of the IEEE*, volume 73, pages 1349–1387, September 1985.

[57] Y. Rabinovich and A. Wigderson. An analysis of a simple genetic algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 215–221, University of California, San Diego, July 1991.

[58] D. W. Redmill and D. R. Bull. Design of low complexity fir filters using genetic algorithms and directed graphs. In *Proceedings of the Second International Conference on GA in Engineering Systems: Innovations and Application, GALESIA-97*, pages 168–173, University of Strathclyde, Glasgow, September 1997.

[59] H. P. Schwefel. *Numerical Optimisation of Computer Models.* John Wiley, Chichester, 1981.

[60] K. C. Sharman, A. I. Esparcia-Alcázar, and Y. Li. Evolving digital signal processing algorithms by genetic programming. In *Proceedings of the First IEE/IEEE International Conference on GA in Engineering Systems: Innovations and Application, GALESIA-95*, pages 473–480, Sheffield, U.K., September 1995.

[61] J. J. Shynk. A complex adaptive algorithm for IIR filtering. *IEEE Trans. on Acoustics, Speech & Signal Processing*, ASSP–34(5):1342–1344, October 1986.

[62] J. J. Shynk. Performance of alternative adaptive IIR filter realizations. In *Proceedings of the 21st Asilomar Conference on Signals, Systems and Computers*, pages 144–150, Pacific Grove, California, November 1987.

[63] J. J. Shynk. Adaptive IIR filtering. *IEEE Trans. on Acoustics, Speech & Signal Processing*, 37(4):4–21, April 1989.

[64] J. J. Shynk. IIR filtering using Parallel-Form realizations. *IEEE Trans. on Acoustics, Speech & Signal Processing*, ASSP-37(4):519–533, April 1989.

[65] D. T. M. Slock and T. Kailath. Numerically stable fast recursive least squares transversal filters. In *Proceedings of the International Conference on Acoustics, Speech & Signal Processing, ICASSP-92*, pages 1365–1368, New York, April 1988.

[66] R. D. Strum and D. E. Kirk. *First Principles of Discrete Systems and Digital Signal Processing.* Addison–Wesley, New York, 1989.

[67] S. Sundaralingam and K. C. Sharman. Genetic evolution of adaptive filters. In *Proceedings of DSP UK, London*, pages 47–53, London, December 1997.

[68] S. Sundaralingam and K. C. Sharman. Genetic evolution of adaptive filters. CSC Research Report CSC-97009, Centre for Systems and Control, University of Glasgow, 1997.

[69] S. Sundaralingam and K.C. Sharman. Evolving complex adaptive IIR structures. In *Proceedings of the IXth European Signal Processing Conference, EUSIPCO-98*, volume II, pages 753–756, Greece, Septembr 1998.

[70] S. Sundaralingam and K.C. Sharman. Evolving filters in multipath environments. In *Proceedings of the Seventh Annual Conference on Evolutionary Programming, EP-98*, pages 397–407, San Diego, March 1998.

[71] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, University of California, San Diego, July 1989.

[72] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, SAn Mateo, CA, 1989.

[73] K. C. Tan, Y. Li, D. J. Murray-Smith, and K. C. Sharman. System identification and linearisation using genetic algorithms with simulated annealing. In *Proceedings of the First IEE/IEEE International Conference on GA in Engineering Systems: Innovations and Application, GALESIA-95*, pages 164–169, Sheffield, U.K., September 1995.

[74] K. S. Tang, K. F. Man, S. Kwong, and Q. He. Genetic algorithms and their applications. *IEEE Signal Processing Magazine*, pages 38–46, November 1996.

[75] S. Theodoridis, C. F. N. Cowan, C. P. Callender, and C. M. S. See. Schemes for equalisation of communication channels with nonlinear impairments. In *IEE Proceedings on Communications*, volume 142, pages 165–171, June 1995.

[76] J. R. Treichler, C. R. Johnson, Jr., and M. G. Larimore. *Theory and Design of Adaptive Filters.* A Wiley-Interscience Publication, New Jersey, 1987.

[77] P. E. Wellstead and M. B. Aarrop. *Self-Tuning Systems: Control and Signal Processing.* John Wiley & Sons Ltd., West Sussex, England, 1991.

[78] M. S. White and S. J. Flockton. A comparative study of natural algorithms for adaptive IIR filtering. In *Proceedings of the second IEE International Workshop on Natural Algorithms in Signal Processing*, volume 1, pages 22/1–22/7, Chelmsford, Essex, November 1993.

[79] M. S. White and S. J. Flockton. Adaptive recursive filtering using evolutionary algorithms. *Evolutionary Algorithms in Engineering Applications*, pages 361–376, January 1997.

[80] D. Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, SAn Mateo, CA, 1989.

[81] B. Widrow, J. R. Glover, Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong, and R. C. Goodwin. Adaptive noise cancelling: principles and applications. In *Proceedings of the IEEE*, volume 63, pages 1692–1716, December 1975.

[82] B. Widrow and S. D. Stearns. *Adaptive Signal Processing.* Prentice Hall, New Jersey, 1985.

Appendix A

Error Surface and Multimodality

In this dissertation, MSE performance surfaces of IIR filters were plotted in several occasions to show the multiple local optima. These 3-D plots are called mesh-plots, which show the variations of objective function values (MSE) against adaptive filters' parameters such as coefficients and poles. The MSE surfaces were obtained for various combinations of filter parameters, while modelling a variety of IIR systems as EA evolves the parameters.

To obtain a MSE, an excitation input $\{s_n\}_{n=0}^{w-1}$, where w represents the number of samples, are applied to both the system and adaptive filter to obtain prediction error, $e(n)$, as shown in Figure A.1. Gaussian noise signal with mean 0 and unity variance is often used as an excitation input in many system modelling problems [51, 64, 73, 62, 12]. The excitation inputs used in chapters 4 - 6 were sets of Gaussian variables with mean 0 and standard deviation, $d = 1$.

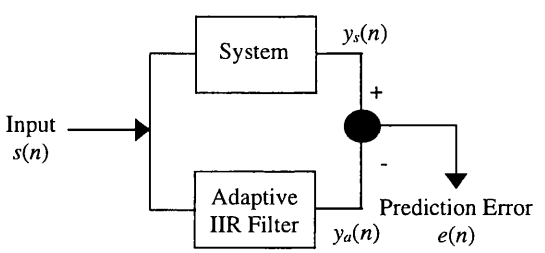

Figure A.1: A block diagram illustrating system identification.

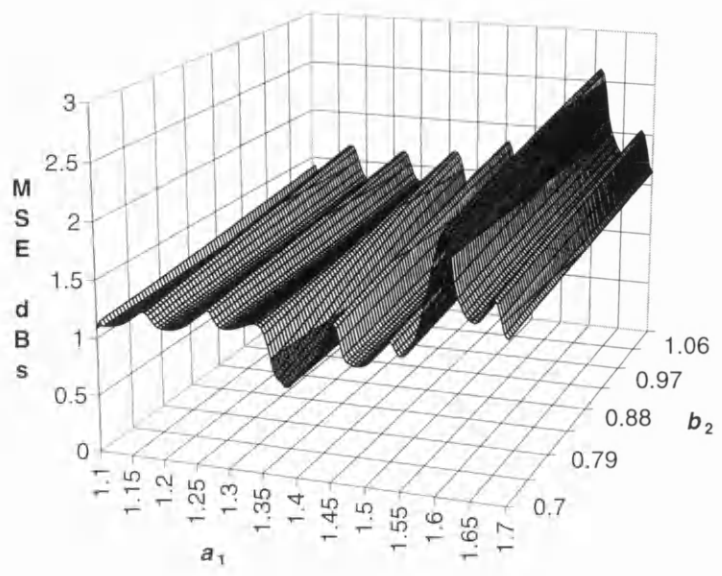

Figure A.2: An example of local MSE surface.

The MSE performance, for example, while modelling a second-order system,

$$H(z) = \frac{0.5 - 0.4z^{-1} + 0.89z^{-2}}{1 - 1.4z^{-1} + 0.98z^{-2}} \tag{A.1}$$

by using an adaptive filter,

$$A(z) = \frac{b_o + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{A.2}$$

can be easily obtained from summed-squared error, which is averaged over a suitable window length w:

$$\text{MSE} = \varepsilon = \frac{1}{w} \sum_{n=0}^{w-1} |y_s(n) - y_a(n)|^2 \tag{A.3}$$

Thus, the MSE obtained for various choices of filter parameters are then plotted as 3-D mesh to show the local surfaces of the MSE. Figure A.2 shows an example of local MSE which is plotted against a_1 and b_2 for $w = 200$, while other parameters were set to the optimum values.

Appendix B

Conversion Between Pole Positions and Polynomial Coefficients

In Chapter 4 of this dissertation, a poles design method was discussed. When poles are designed, they must be converted into equivalent coefficients before evaluating the fitness of each individual.

The pole polynomials can be easily converted into coefficients by using simple iterative solutions. For example, consider an AR process which is given by z-domain representation as

$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}, \cdots, -a_N z^{-N}} \tag{B.1}$$

where the index N represents the number of feedback coefficients. The system $H(z)$ can also be represented by all pole form

$$H(z) = \frac{1}{(1 - p_1 z^{-1})(1 - p_2 z^{-2}) \cdots (1 - p_N z^{-N})} \tag{B.2}$$

Where $\{p_i\}_{i=1}^{N}$ represent the poles of the system.

A relationship can be made between poles and coefficients in such a way that

for $N = 1$

$$a_1 = p_1 \tag{B.3}$$

for $N = 2$

$$\begin{aligned} a_2 &= -p_1 p_2 \\ a_1 &= p_1 + p_2 \end{aligned} \tag{B.4}$$

for $N = 3$

$$\begin{aligned} a_3 &= p_1 p_2 p_3 \\ a_2 &= -\{p_1 p_2 + p_1 p_3 + p_2 p_3\} \\ a_1 &= \{p_1 + p_2 + p_3\} \end{aligned} \tag{B.5}$$

for $N = 4$

$$\begin{aligned} a_4 &= -p_1 p_2 p_3 p_4 \\ a_3 &= \{p_1 p_2 p_4 + p_1 p_3 p_4 + p_2 p_3 p_4 + p_1 p_2 p_3\} \\ a_2 &= -\{p_1 p_4 + p_2 p_4 + p_3 p_4 + p_1 p_2 + p_1 p_3 + p_2 p_3\} \\ a_1 &= p_1 + p_2 + p_3 + p_4 \end{aligned} \tag{B.6}$$

From equations (B.3) - (B.6) a relationship between poles and the coefficients can be made for a general order N,

$$a_N(N) = (-1)^{N-1} \prod_{i=1}^{N} p_i \tag{B.7}$$

$$a_l(N) = a_l(N-1) - a_{l-1}(N-1)p_N \quad \text{for } 1 < l < N \tag{B.8}$$

$$a_l(N) = \sum_{i=1}^{N} p_i \tag{B.9}$$

where $a_l(N)$ of (B.8) represents the l^{th} coefficient of the N^{th} order system, while $a_1(N-1)$ represents the l^{th} coefficient of the $(N-1)^{\text{th}}$ order system.

Appendix C

Test Signals

C.1 Classification of Signals

The signals considered in this dissertation can be classified as one of two types: *Deterministic Signals* and *Stochastic Signals*. A deterministic signal is one that has no random components and is therefore completely predictable. Examples of these signal types were used in the channel equalisation and noise cancelling applications in Chapter 6. These signals can be exactly described at any time using an analytical or mathematical description.

A stochastic (or random signal) is a signal that cannot be exactly described by a deterministic mathematical function. The value of a random signal cannot be determined until it has been measured or observed. Most signals that arise from naturally occurring process (e.g. speech, vision, etc.) have random characteristics and are thus stochastic.

C.2 Representation of Random Signals

Discrete time random signal considered in this work are represented as a sequence of random variables, $\{x_n\}$. Each sample x_n is a random variable with its own probability density function (pdf), variance (σ^2) and mean (μ).

pdf is a mathematical function that describes the probability that a random variable will, when observed, have a particular value. For example, the probability that a variable x will lie in the incremental range $y \cdots y + \delta y$, is given by $p_x(y)\delta y$,

$$\text{Probability}[y \leq x \leq y + \delta y] = p_x(y)\delta y \tag{C.1}$$

Mean:- the value of a random variable is defined as its expected (or average) value,

$$\mu(x) = \text{E}\,[x] \tag{C.2}$$

Variance of a random variable is usually denoted by σ^2. It represents the mean value of the square of a random variable after its mean has been subtracted,

$$\sigma^2 = \text{E}\,\left[(x - \mu)^2\right] \tag{C.3}$$

Power Spectral Density function is yet another way of describing the characteristics or structure of a random signal. It indicates how the energy in a signal is distributed over different frequencies. It is defined as the DFT of the covariance function,

$$P(\omega) = \text{DFT}[C_k] = \sum_{k=-\infty}^{\infty} C_k e^{-j\omega k} \tag{C.4}$$

C.3 Stochastic Processes

C.3.1 Stationary and Non-stationary Signals

A random signal is said to be stationary if each sample has exactly the same probability density function. A stationary random signal is one whose statistics do not change over time. Furthermore, a single pdf (and mean and variance) is sufficient to describe a stationary random signal.

In a nonstationary signal, each sample in the signal sequence can have a unique pdf. Non-stationary signals are therefore more difficult to analyse than stationary ones. Many important signals are nonstationary (e.g. speech). However, they can in some instances be considered to be stationary over a short time

segment. In the case of human speech, analysis has shown it is approximately stationary over periods less than about 30-millisecond duration. This implies that the analysis and processing can be simplified by breaking up the signal into short segments and applying stationary techniques to each segment.

C.3.2 Special Test Signals

White Noise

White noise is an important type of random signal. It is defined as a sequence of independent and identically distributed random variables, where each sample in the sequence has the same Gaussian probability density function. This random process occurs frequently in signal processing, as it is a suitable model for many naturally occurring processes, such as thermal noise or random measurement errors.

White noise has an energy density that is equal at all frequencies, just like white light. Since each sample in a white noise process is independent of every other sample, the covariance function is zero for all lags, except C_0 which represents the variance of the noise. i.e. the covariance of white noise can be represented as an impulse,

$$C_k = \sigma^2 \delta k \tag{C.5}$$

The power spectrum of the white noise sequence is therefore,

$$\hat{P}(\omega) = \sigma^2 \tag{C.6}$$

where σ^2 is the variance of white noise sequence. White noise can be generated from successive independent random numbers from a Gaussian random number generator, which takes mean μ and standard deviation σ as arguments.

$$\text{Gaussianvariable}\{v\} = \text{gauss}(\mu, d) \tag{C.7}$$

The standard deviation of the Gaussian random number generator can be related to the variance, σ^2 in such a way that

$$d = \sqrt{\text{variance}} = \sigma \tag{C.8}$$

Pseudo Random Binary Sequence

Pseudo random binary sequences (PRBSs), also known as pseudo noise (PN), linear feedback shift register (LFSR) sequences or maximal length binary sequences (m-sequences), are widely used in digital communications [43]. It is truly random sequence in which the bit pattern never repeats. The sequence serves as a reference pattern with known random characteristics for the analysis, optimisation and performance measurement of communication channels and systems.

A PRBS sequence, PRBS $\in \{-1, +1\}$ has an autocorrelation of 1 at zero phase (no time shift), and 0 at all other phases. Therefore, the power spectrum of PRBS sequence is similar to white noise except that it has unity variance,

$$\hat{P}(\omega) = 1 \tag{C.9}$$

Appendix D

Measure of Performance

D.1 Performance Functions

The functions such as Power Spectral Density (PSD), Signal to Noise Ratio (SNR) and Bit Error Rate (BER) are used to measure the performance of a system.

D.2 Power Spectral Density

D.2.1 Power Spectrum Estimation

Power spectral density function is used to describe the characteristics or structure of a random signal. For example, let us assume that

$$\{s_n, n = 0, \cdot\cdot, N-1\} \tag{D.1}$$

be a discrete time ergodic stationary random process. The term ergodic means that a time average will converge to the expectation, as the time interval becomes infinite.

The PSD function of the discrete time signal $\{s_n\}$ can be calculated from

$$P(\omega) = \sum_{k=-\infty}^{\infty} C_k e^{-j\omega k} \tag{D.2}$$

where C_k is the covariance sequence of $\{s_n\}$ described by

$$\begin{aligned} C_k &= E\{s_n s_{n-k}^*\} \\ &= \sum_{n=-\infty}^{\infty} s_n s_{n-k} \end{aligned} \tag{D.3}$$

For zero mean signals, the covariance can be estimated using the following time average,

$$\hat{C}_k = \frac{1}{N-k} \sum_{n=k}^{N-1} s_n s_{n-k} \quad k = 0 \cdots N-1 \tag{D.4}$$

It is important to note that maximum covariance lag that can be estimated from (D.4) is C_N. Furthermore, the limits of the power spectrum of equation (D.1) are infinite ($-\infty \leq k \leq \infty$). Therefore, it must be truncated using rectangular window, which results an estimate $\hat{P}(\omega)$,

$$\hat{P}(\omega) = \sum_{k=-(N-1)}^{N-1} \hat{C}_k e^{-j\omega k} \tag{D.5}$$

D.2.2 The Use of the DFT in Spectrum Estimation

The equation (D.2) is an indirect method of calculating the estimated power spectrum $P(\omega)$, because it requires two steps. First, the autocorrelation C_k is computed from the data sequence ($\{x_n\}$ and $\{s_n\}$ respectively) and then the Fourier transform of the autocorrelation is computed. However, it can be computed by use of the DFT, which in turn is efficiently computed by the FFT algorithm [55]. For example, if we have N data points, we compute as a minimum the N-point DFT for power spectrum estimation,

$$P\left(\frac{k}{N}\right) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x_n \exp\left(\frac{-j2\pi n}{N}\right) \right|^2, \; k = 0, 1, \cdots, N-1 \tag{D.6}$$

at the frequencies $f_k = \frac{k}{N}$. Discrete Fourier transform of a given sequence can be easily calculated from standard data analysis tools package.

D.3 Signal to Noise Ratio

Signal to noise ratio calculation was used to generate desired noise sequence for the experiments illustrated in Chapter 6. Signal to noise ratio is defined as that

the ratio of signal power to noise power:

$$\text{SNR} = \frac{P_s}{P_n} \tag{D.7}$$

where P_s and P_n are the average signal and noise power respectively. The aim is to generate suitable noise sequences so as to meet desired signal to noise ratios for given data signals.

The average power of a data sequence $\{x_n, n = 0, \cdots, N-1\}$ can be calculated from

$$P_s = \frac{1}{N} \sum_{n=0}^{N-1} |x_n|^2 \tag{D.8}$$

If the signal $\{x_n\}$ is a random process, the average power is simply the variance of that random process.

Now consider the SNR calculation given in equation (D.7). The equation can be rewritten as

$$\text{SNR} = \frac{P_s}{var\{v_n\}} = \frac{P_s}{\sigma_v^2} \tag{D.9}$$

where $\{v_n\}$ represents the noise signal, while the term var denotes the variance. In this dissertation, Gaussian noise was used to represent the possible noise interference. Therefore, the noise power is simply the standard deviation of the Gaussian random number generator. For a given signal to noise ratio, x dB, the standard deviation, σ, of the Gaussian random generator can be calculated from

$$d = \sigma_v = \sqrt{\frac{P_s}{10^{\frac{x}{10}}}} \tag{D.10}$$

D.4 Bit Error Rates

Bit error rate is defined as the ratio of misclassified to correct symbols at the receiver. It is an important tool to measure the performance of digital systems, which deal binary signals. The BER performance of a channel equaliser can be calculated from,

$$\text{BER} = \frac{\text{Number of Error symbols Received}}{\text{Correct Symbols}} \tag{D.11}$$

For example, the BER of 100 errors received over 10000 symbols in dBs is

$$\text{BER} = 20 \log \left\{ \frac{100}{10000 - 100} \right\} = -39.8 \text{ dB} \tag{D.12}$$

In this work, a PRBS was used as binary symbols which was taking a value from the set $\{-1, +1\}$. The output of the equaliser, definitely, do not have values from the original set $\{-1, +1\}$ due to noise interference or poor equalisation. Therefore, a decision device is employed at equaliser output y to form an estimate of original PRBS signal, s:

$$\begin{aligned} sgn(y) = \hat{x} \quad &= \quad +1 \; y \geq 0 \\ &\quad\; -1 \; y < 0 \end{aligned} \tag{D.13}$$

$\hat{x}$ is now a binary sequence which takes values from $\{-1, +1\}$. The bit errors are the counted if $\{x \neq \hat{x}\}$.

Appendix E

LMS Algorithm

In Chapter 6, we have compared the performance of IIR equalisers against FIR filters, which were adapted through LMS algorithm. Least mean square algorithm, used in this chapter was standard LMS developed by Widrow in 1975. The computational procedure for the LMS algorithm is summarised below.

Inputs:

$$\begin{aligned} \boldsymbol{X}(n) &= [x_1(n), x_2(n), \cdots, x_{N-1}(n)] \text{ input vector} \\ \boldsymbol{W}(n) &= [w_1(n), w_2(n), \cdots, w_{N-1}(n)] \text{ weight vector} \\ d(n) & \quad \text{desired signal} \end{aligned}$$

Outputs:

$$\begin{aligned} y(n) &= \boldsymbol{X}^t(n)\boldsymbol{W}(n) \text{ signal output} \\ e(n) &= d(n) - y(n) \text{ error signal} \end{aligned}$$

1. initialise the weights (choose initial weights arbitary), e.g. $\{w_k(0) = 0, k = 0, 1, \cdots, N-1\}$.

2. compute the filter output

   $\hat{y}(n) = \boldsymbol{X}^t(n)\hat{\boldsymbol{W}}(n)$ or

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

3. compute the error

   $e(n) = d(n) - \hat{y}(n)$

4. update the weights

   $\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + 2\mu e(n)\boldsymbol{X}(n)$ or

   $w_k(n+1) = w_k(n) + 2\mu e(n)x(n-k)$

   where μ represents the step size.

5. repeat steps 2 - 4.

When the input sequence $x(n)$, the output sequence $y(n)$, and the desired sequence $d(n)$ are all complex valued, then a complex version of the LMS algorithm must be used. The weight vector update of a complex LMS can be given by

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + 2\mu e(n)\boldsymbol{X}^*(n) \tag{E.1}$$

The asterisk $*$ denotes complex conjugation. The above algorithm with $\mu = 0.02$ was developed as C++ class for testing the inverse models illustrated in Chapter 6.

Appendix F

IIR Realisation Forms

To resolve stability problems, alternative realisations such as parallel, cascade or lattice-forms are considered with simple first or second-order filters [64, 48, 62, 63, 69, 70, 67, 44, 47, 3, 10]. These structures offer simple stability monitoring techniques and are less sensitive to finite-precision effects (coefficient round-off) [64, 63, 48, 62]. For example, the parallel or cascade forms comprising second-order adaptive IIR filters are trivial to factor. The lattice form on the other hand requires only that each reflection coefficient have a magnitude less than 1.

F.1 Parallel Form

The parallel form can be obtained from a partial fraction expansion of the pole-zero filter:

$$H(z) = \frac{b_0(n) + b_1(n)z^{-1} + \cdots + b_{M-1}(n)z^{L}}{1 - a_1(n)z^{-1} - a_2(n)z^{-2} - \cdots - a_M(n)z^{-L}} \tag{F.1}$$

where

$$\{a_i(n),\ i = 1 \cdots L;\ b_j(n),\ j = 0 \cdots L\}$$

represent the feedback and feedforward coefficients of the filter, and L represents the filter order. The equivalent parallel form representation of (F.1) is:

$$H(z) = \sum_{k=1}^{M} \frac{b_{0k}(n) + b_{1k}(n)z^{-1} + b_{2k}(n)z^{-2}}{1 - a_{1k}(n)z^{-1} - a_{2k}(n)z^{-2}} \tag{F.2}$$

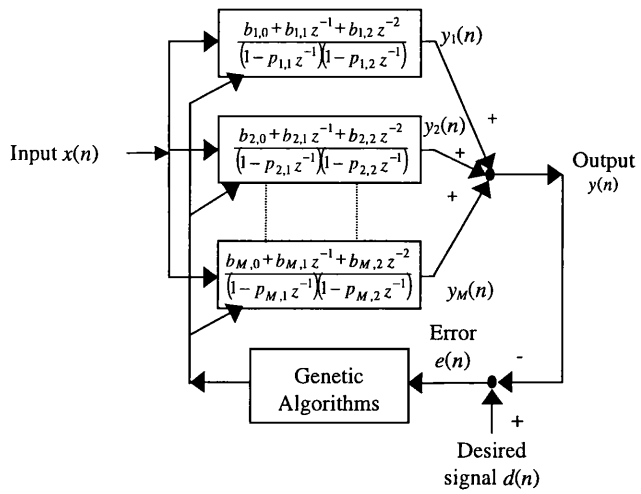

Figure F.1: A parallel form adaptive IIR filter.

where $M = \frac{(L+1)}{2}$ if L is odd, or $M = \frac{L}{2}$ if L is even.

Figure F.1 illustrates a parallel form adaptive IIR structure, which comprises with M second-order subsections, each having three feedforward coefficients $\{b_{kj},\ k = 1 \cdots M;\ j = 0 \cdots 2\}$ and two feedback coefficients $\{a_{ki},\ k = 1 \cdots M;\ i = 1 \cdots 2\}$. The overall output of this parallel structure is the sum of all the output from each sub-filter and is given in mathematical notation as:

$$y(n) = \sum_{k=1}^{M} y_k(n) \tag{F.3}$$

where $\{y_k(n)\}_{k=1}^{M}$ is the output of each parallel filter and M represents the total number of parallel sections.

F.2 Cascade Form

The cascade form is very similar to the parallel form in that it is generated by factoring the pole-zero filter (F.1) into the product of M ($M = \frac{(L-1)}{2}$ if L is odd, or $M = \frac{L}{2}$ if L is even) second-order sections. The equivalent cascade-form representation of $H(z)$ is:

$$H(z) = \prod_{k=1}^{M} \frac{b_{0k}(n) + b_{1k}(n)z^{-1} + b_{2k}(n)z^{-2}}{1 - a_{1k}(n)z^{-1} - a_{2k}(n)z^{-2}} \tag{F.4}$$

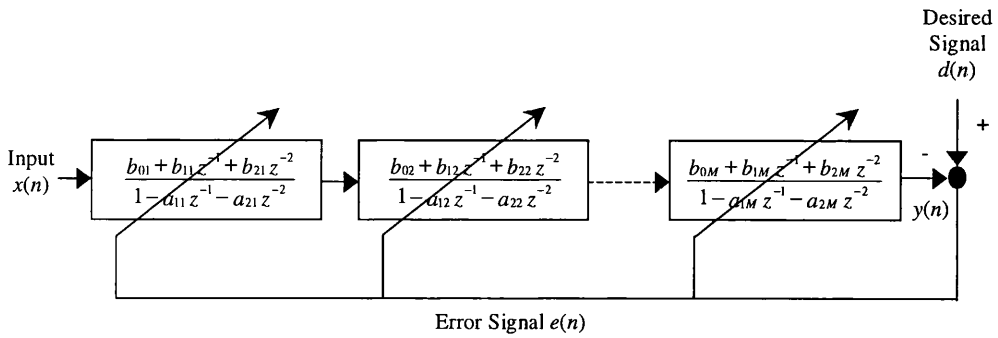

Figure F.2: A parallel form adaptive IIR filter.

when M is odd, the last stage in a parallel or cascade structure will be a single pole stage. An example of cascade form realisation is shown in Figure F.2. The same stability monitoring can be applied in cascade-forms, since similar second-order sections as in a parallel form structure form the overall structure. The overall output of a cascade realisation comprising M subsections is the convolution of all the output from each sub-filter:

$$y(n) = \prod_{k=1}^{M} y_k(n) \tag{F.5}$$

It can be seen from the above equation that the output signal of each section depends on the coefficients of that sections as well as all previous sections. It has been shown in a recent publication that the gradient based adaptation of cascade-form filters require a much longer time than that of other realisations [62].

A common disadvantage of the parallel and cascade forms is that they can have multiple optima with same fitness values that arise from rearranging the poles among the different sections. The number of possible rearrangements would be M factorials for M cascade or parallel sections. This property leads to a major drawback for gradient based algorithms when the subsections of these realisations are identically initialised (e.g. $\mathbf{a}(0) = \mathbf{0}$).

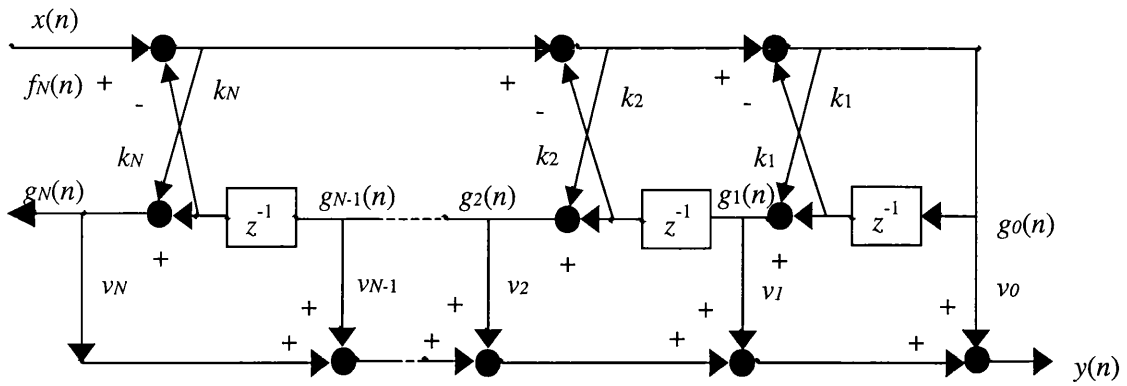

Figure F.3: A lattice structure.

F.3 Lattice Structure

The filter in (5.2) can be implemented in the form of a lattice with different weights $v_i(n)$ and $k_i(n)$, which is stable if the lattice coefficients $k_i(n)$ are all less than 1.

$$f_L(n) = x(n) \tag{F.6}$$

$$f_{m-1}(n) = f_m(n) - k_m g_{m-1}(n-1),\ m = L, L-1, \cdots, 1 \tag{F.7}$$

$$g_m(n) = k_m f_{m-1}(n) + g_{m-1}(n-1),\ m = L, L-1, \cdots, 1 \tag{F.8}$$

$$y(n) = \sum_{m=0}^{L} v_m g_m(n) \tag{F.9}$$

where $g_i(n)$ and $f_i(n)$ are backward and forward residuals of the i^{th} lattice stage at time n. A lattice structure of (5.2) is shown in Figure F.3.

Appendix G

Genetic Programming

Genetic programming is a branch of GAs. The main difference between GP and GAs is the representation of solution. Genetic programming creates computer programs in the *LISP* or *SCHEME* computer languages as the solution. That is, the objects that constitute the population are not fixed-length character strings that encode possible solutions to the problem at hand, they are programs that, when executed, are the candidate solutions to the problem. These programs are expressed in GP as expression trees, rather than lines of codes. Thus, for example, the simple program,

$$(\div(-\sqrt{(*2a)b})(*2a) \tag{G.1}$$

would be represented by an expression tree shown in Figure G.1. The expression trees in the population are composed of elements called nodes. According to the function they represent, these nodes can be classified into *Functions* and *Terminals* [60]. The functions and terminals set are the alphabet of the programs to be made.

Functions: are processing nodes that receive one or more input values and produce a single output value (e.g. $+$ and $*$ in the example shown in Figure G.1).

Terminals: are nodes, which represent external input and zero argument functions.

Genetic programming, uses four steps to solve problems:

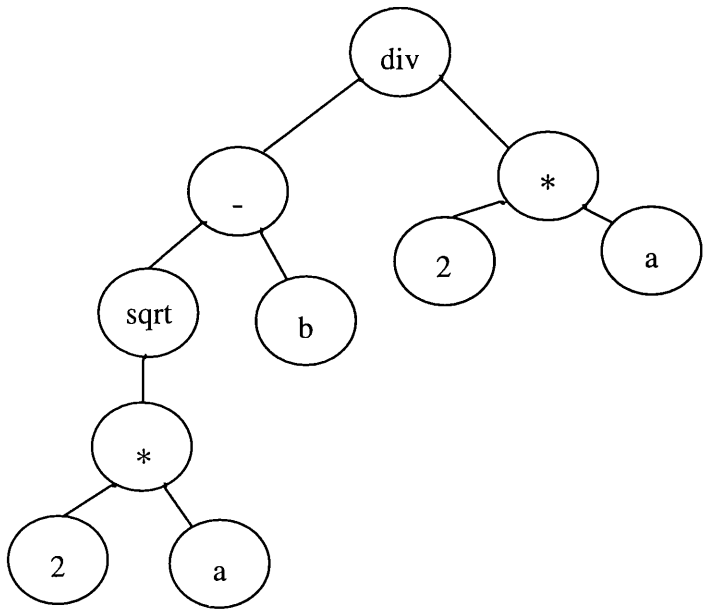

Figure G.1: An example of expression tree for a simple program, given in equation (G.1).

1. Generate an initial population of random compositions of the functions and terminals of the problem.

2. Execute each program in the population and assign it a fitness value according to how well it solves the problem. The fitness measures the performance of the function coded by the expression trees.

3. Create a new population of computer programs via:

   (a) **Selection**: selects couples of parent trees for reproduction based on their fitness.

   (b) **Crossover**: take randomly selected sub-trees in the individuals and exchanging them. For example, a crossover operation between two different parents is shown in Figure G.2.

   (c) **Mutation**: randomly introduce variations in the programs. There are two types of mutation are in use:

      i. A function can only replace a function, or a terminal can only replace a terminal.

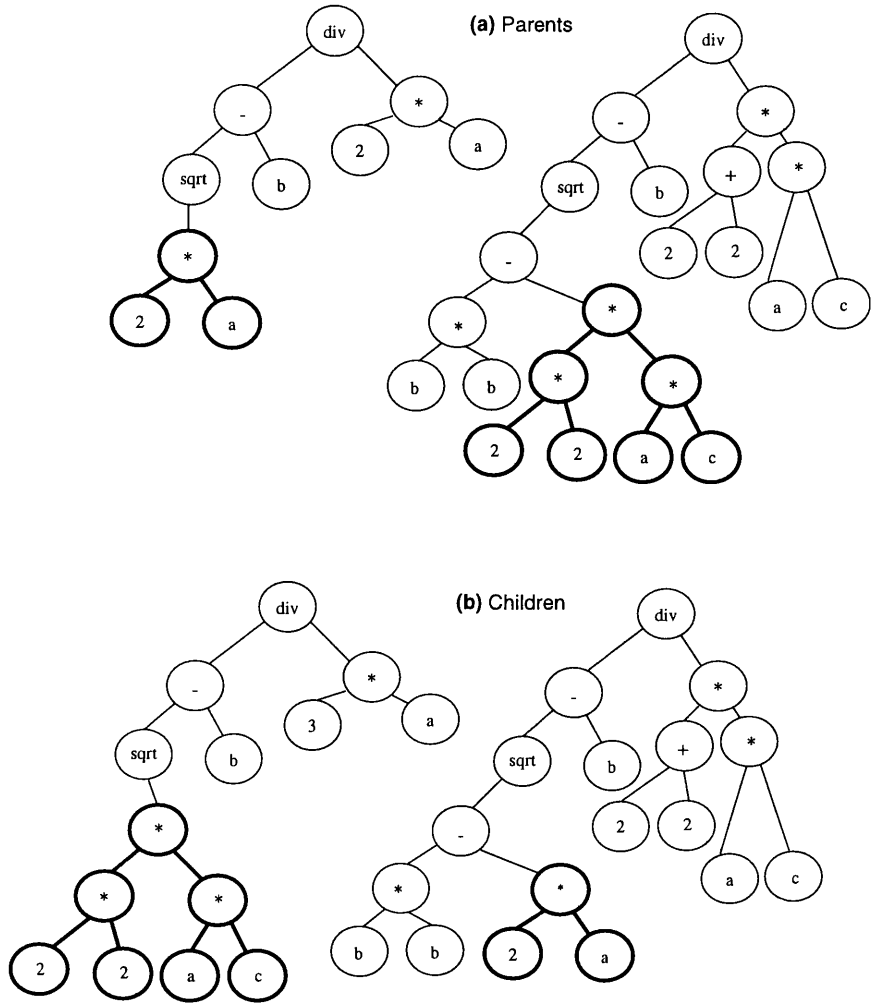

Figure G.2: Crossover in Genetic Programming: crossover operation with different parents.

ii. Entire sub trees can replace another sub tree.

Figure G.3 shows these types of mutation.

4. The best computer program that has appeared in any generation is designated as the result of genetic programming.

Special Feature Compared With a GA: One of the main advantages of GP over GAs is that identical parents can yield different offspring, whilst in GAs identical parents would yield identical offspring. Figure G.4 illustrates this difference, where bold sections indicate the subtrees to be swapped. Genetic programming can be used to evolve an algebraic expression as part of an equation

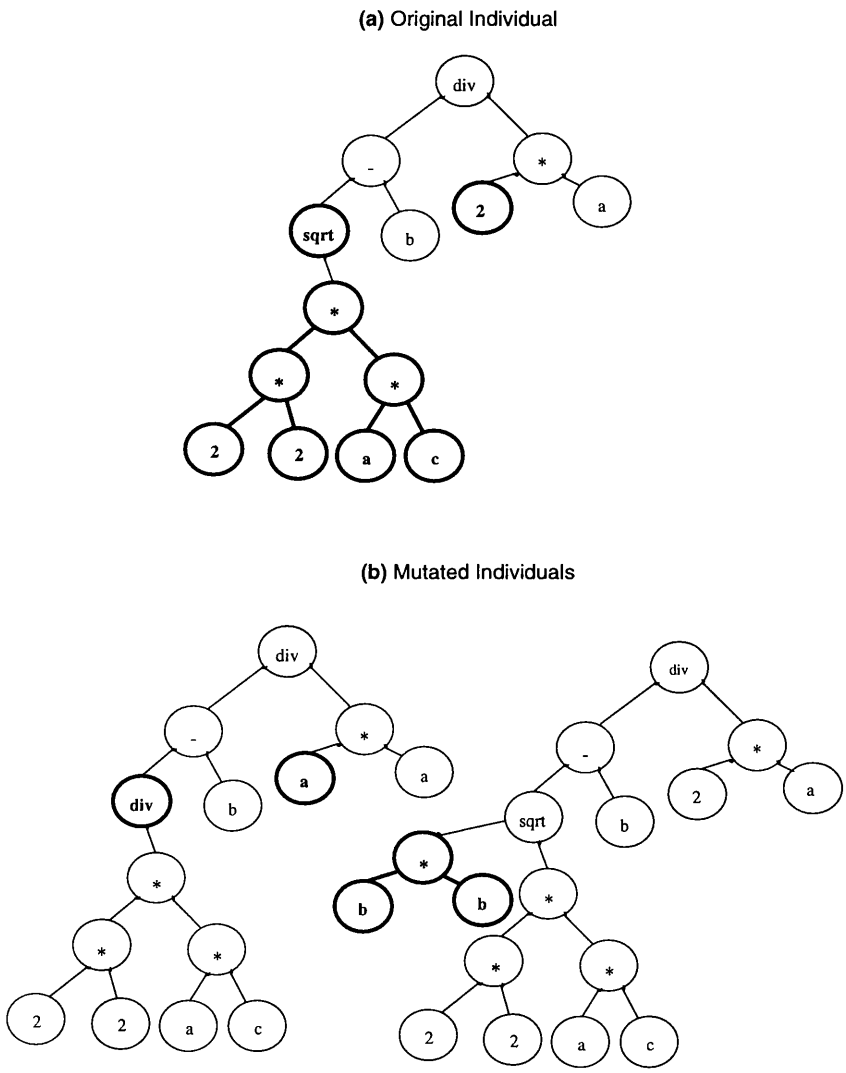

Figure G.3: Mutation operation in Genetic Programming.

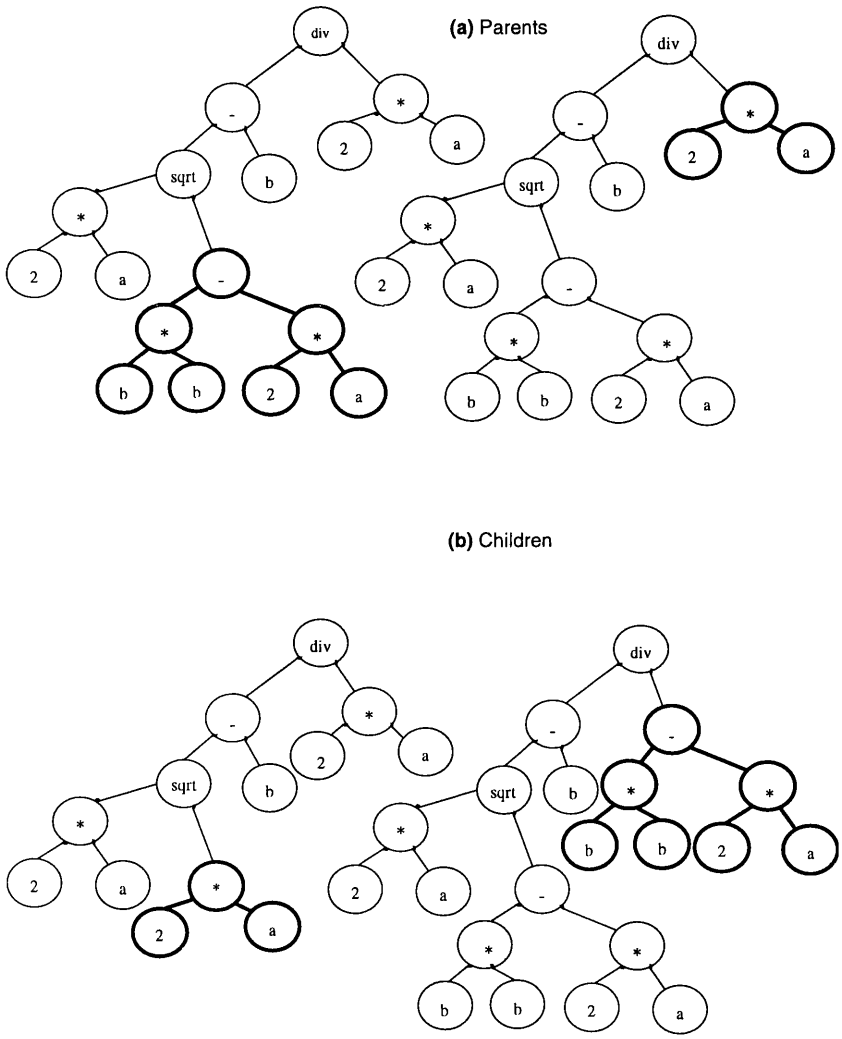

Figure G.4: Crossover in Genetic Programming: crossover operation with identical parents.

representing measured input-output response data. For example, used with a carefully selected library of functions, it can reveal information about the physical structure of a system and produce an accurate model describing the system [18].