The applicability of simple adaptive algorithms for the active control of random noise in ventilation ducts.

By

Nick Abbott

A thesis submitted to the University Of London for the degree of Master of Philosophy.

Royal Holloway and Bedford New College

University Of London

December 1988

Abstract

The applicability of simple adaptive algorithms for the active control of random noise in ventilation ducts.

The term *Active Noise Control* (ANC) describes the suppression of an unwanted sound field by the superposition of an antiphase field.

In general, an active control system consists of a sensing mechanism to detect the unwanted noise, a processing element to analyse the sound and produce the antiphase signal and a system of secondary sources to radiate the required *antisound*.

The ideal ANC system would have the ability to modify its own response to accommodate any changes in the environment in which it is placed. Such as system is known as an *adaptive* one.

This thesis is concerned with assessing the suitability of several different recursive algorithms for adapting digital controllers to control random noise within ventilation ducts. Algorithms for adapting both Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) digital filters are studied.

Computer simulations are presented which directly compare the performance of the algorithms when used in system identification and when used to control adaptive systems in ducts. Experiments were made in ducts of varying length and ranging from anechoic to highly reverberant in nature.

Conclusions drawn from the simulations indicate some very significant savings in terms of economy of filter length when implementing adaptive systems within the duct.

To my family

Acknowledgements

Firstly, I would like to thank my supervisor Stuart Flockton for all the help, encouragement and patience that he has shown during this research project.

Thanks also go to my colleagues Thomas Gurrie and Colin Bean for their useful guidance throughout the two years at Royal Holloway College.

Thanks to my CASE sponsors GEC and especially to John Edwards for his interest in my work.

Special thanks go to Thomas Crane, Brian Cowan and Max Van-Daalen for their valuable advice and to Peter Voke for his help with the SCI wordprocessing package.

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

List of figures

Chapter 1

Chapter 2

Abbreviations

Attn	- Attenuation
AFM	- Adaptive Filtering Mode
D	- Detector Microphone/s
FIR	- Finite Impulse Response
HARF	- Hyperstable Adaptive Recursive Filter
IIR	- Infinite Impulse Response
LMS	- Least Mean Squares
M	- Monitor Microphone/s
MSE	- Mean Square Error
MMSE	- Minimum Mean Square Error
PS	- Primary Source
RLMS	- Recursive Least Mean Squares
SHARF	- Simple Hyperstable Adaptive Recursive Filter
SS	- Secondary Source

List of abbreviations and symbols

Symbols

A - Transfer function of an all zero adaptive filter

B - Transfer functionof an all pole adaptive filter

C - Error compensation filter

a_i - Coefficients of an all zero digital filter

b_j - Coefficients of an all pole digital filter

c_i - Coefficients of error compensation filter C

G - Random White noise generator

H_{01} - Transfer function between input to PS and output of D

H^z_{01} - All zero part of H_{01}

H^p_{01} - All pole part of H_{01}

H_{03} - Transfer function between input to PS and output of Secondary Source SS.

H^z_{03} - All zero part of H_{01}

H^p_{03} - All pole part of H_{03}

H_{23} - Transfer function between input to SS and output of M

H^z_{23} - All zero part of H_{23}

H^p_{23} - All pole part of H_{23}

H_{21} - Acoustic feedback transfer function between input to SS and the output of D

H^z_{21} - All zero part of H_{21}

H^p_{21} - All pole part of H_{21}

H_{con} - Transfer function of general ANC controller

H_c - Transfer function of controller in the one dimensional duct

H_{pz} - Unknown pole/zero transfer function

H_e - Error plant transfer function

H'_e - Model of error plant

H^{-1}_e - Inverse of error plant H_e

List of abbreviations and symbols

H_t - Transfer function of error compensation filter for adapting all zero filter A

$H^z{}_t$ - All zero part of H_t

$H^p{}_t$ - All pole part of H_t

H_u - Transfer function of error compensation filter for adapting all pole filter B

$H^z{}_u$ - All zero part of H_u

$H^p{}_u$ - All pole aprt of H_u

H_{rt} - Round Trip transfer function in the duct

K - Constant used in griffiths formula

L_0 - Delay (in sample periods) between left hand termination of duct and PS

L_1 - Delay between PS and D

L_2 - Delay between SS and D

L_3 - Delay between SS and M

L_4 - Delay between M and right hand duct termination

M - Number of recursive taps in an IIR filter

N - Number of direct taps in an FIR/IIR filter

n - Discrete time variable

N_{ff} - Number of taps in pure FIR filter in simulation

P_1 , P_2 - Real parts of the coordinates of the poles of the adaptive controller on the complex z-plane

p_0 , p_1 - Coefficients of the direct element of the error compensation filter H_c

q_1 , q_2 - Coefficients of the recursive element of the error compensation filter H_c

Q - Number of taps in the model of the error plant

r_1 , r_2 - Reflection coefficients of the duct terminsations

x(n) - Input to digital filter

$X(z^{-1})$ - z transform of input to a digital filter x(n)

List of abbreviations and symbols

$x'(n)$ - Input to a digital filter after direct error compensation

$y(n)$ - Output of a digital filter

$Y(z^{-1})$ - z-transform of ouptut of a digital filter $y(n)$

$y'(n)$ - Output from a digital filter after direct error compensation

Z - Real part of the coordinates of zero of the adaptive controller on the complex z-plane

z^{-1} - One sample delay

$\alpha_i(n)$ - Direct updating variables used in Whites RLMS algorithm

$\alpha(n)$ - Updating variables used in simplified recursive update algorithms of Stearns and Fan.

$\beta_i(n)$ - Recursive updating variables used in Whites RLMS algorithm

$\beta(n)$ - Updating variables used in simplified recursive update algorithms of Stearns and Fan.

μ_a - Convergence parameter for adapting filter A

μ_b - Convergence parameter for adapting filter B

Δ - Pure delay

Σ - Series sum of...

1.1 Introduction to Active Noise Control

The term *Active Noise Control* (ANC) describes the suppression of an unwanted sound field by the superposition of an antiphase field. The idea is by no means new; the earliest published work in this field was conducted by an American called *Paul Lueg* [1], who proposed, developed and patented a system 55 years ago.

In general, an active control system consists of a sensing mechanism to detect the unwanted noise, a processing element to analyse the sound and produce the antiphase signal and a system of secondary sources to radiate the required *antisound*. Over the years, many implementations of this idea have been explored and several review texts [2,3] exist which describe the state of the art up to about 1982. This thesis does not attempt to review general early work in this field in any detail but is specifically concerned with up to date work in active control in ventilation ducts, an area that has received a lot of practical attention in the last ten years [4,5,6,7,8,9,10].

In the case of the ventilation duct, the unwanted noise may be caused by machinery such as a noisy fan and will propagate down the duct until it reaches a termination where it may cause great annoyance.

The work described in this thesis is concerned with the simple 1-dimensional case [6,8,11] in which the highest frequency propagating down the duct is below the cut-off frequency of the first cross mode within the duct. The absence of cross modes within the duct means that the propagating sound waves are plane waves and it therefore

follows that only one detector and secondary source are required to actively control sound at the termination. It should be noted however that any discussion given here is in principle also applicable to multi channel systems which have an array of sensors and secondary sources; only the 1-dimensional situation is considered in this thesis.

Several comparatively simple systems have been developed [4,5,6] that in previous years have been the only practicable solution. It has been the advent of fast digital processors such as the *TMS320* series [12] and others [13,14] that has meant that the implementation of far more sophisticated control systems has become a practicality, no longer is progress held back by the inability to process information both fast and cheaply enough. In the last five years there have been several attempts to produce working ANC systems using high speed digital devices: *Poole* et. al. [8] and more recently *Eriksson* [9] have produced commercially available systems to combat noise in duct systems.

1.2 The need for active control

It is important to understand the necessity for active control as opposed to traditional passive damping: The use of passive damping such as acoustic tiles and absorbent foam has been shown to be very effective at high frequencies [8]. However, for efficient passive damping, the physical dimensions of the absorbent need to be of the order of the wavelength of the sound to be absorbed. It should be clear that at low frequencies (below 500 Hz) we would need large and costly passive silencers and so are prompted to search for a better solution. Conversely, active control of high frequency noise is a far less practicable solution since the speed with which information must be processed is very great.

To complement the cost saving, the use of active control as a solution in the ventilation duct has the practical advantage that it need not impede the flow of air down the duct as much as a passive silencer with the same performance - there would not be much call for a ventilation system that was guaranteed absolutely silent but did not supply any air.

1.3 The duct problem

It is important to appreciate fully the requirements of a control system for the duct and to this end the physical situation will be described briefly. Without loss of generality we can represent the primary noise source within the duct as an electroacoustic transducer such as a loudspeaker and use the electrical input to this transducer in our calculations. Figure(1.1) shows this conceptual model of the duct with the primary source, control system and transducers in place.

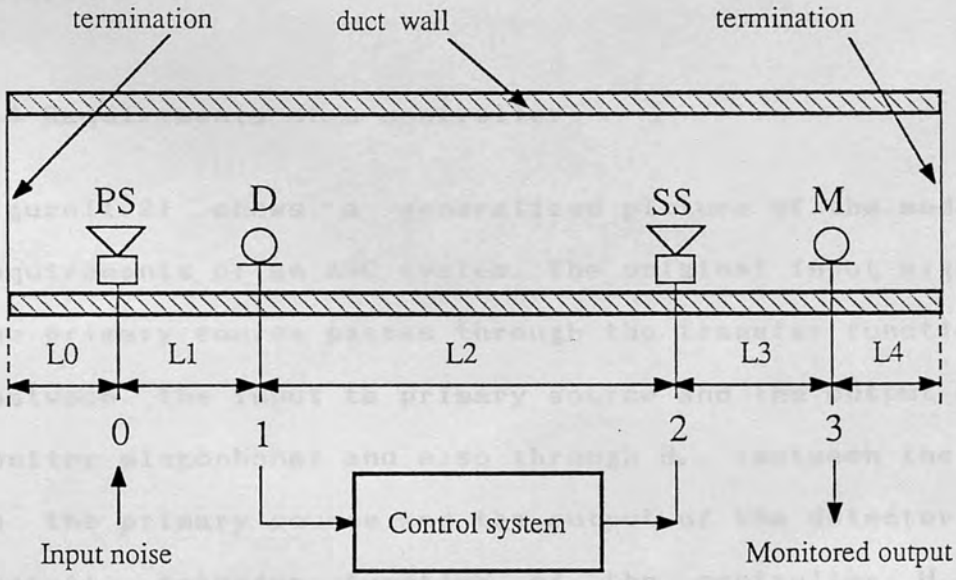


Figure 1.1 - Conceptual model of the 1-dimensional duct system

An active control system is required to detect the sound at the detector microphone (D) and produce a controlling signal from the secondary source (SS) such that (ideally) zero residual sound field is observed at the monitor position M.

Let us consider the propagation of acoustic energy down such a duct. The duct itself may be such that the propagation of energy down it will be relatively loss free. Conversely, there could be quite significant propagation loss. The consequences of the differing amount of propagation loss in the duct will be dealt with in detail later in this thesis. The duct may be anechoic or reverberant in that it may have terminations at its ends that will give rise to reflections inside the duct. Again, the consequences of this are looked at in more detail later.

1.4 Requirements of a controller

Figure{1.2} shows a generalized picture of the modelling requirements of an ANC system. The original input signal to the primary source passes through the transfer function H_{03} (between the input to primary source and the output of the monitor microphone) and also through H_{01} (between the input to the primary source and the output of the detector). The overall transfer function of the controller H_{con} is specified by the requirement that it must be such as to cause the cancellation of any acoustic disturbance at the monitoring position by radiating a controlling signal through H_{23} (between the input to the secondary source and

the output of the monitor microphone).

It is clear that in order to obtain an identically zero residual sound field at the monitor position that the controller H_{con} should have the transfer function :

$$H_{con} = - \frac{H_{03}}{(H_{01})(H_{23})} \quad .....Eq.(1.1)$$

We can now move on to extend this simple idea to an ANC system within the duct.

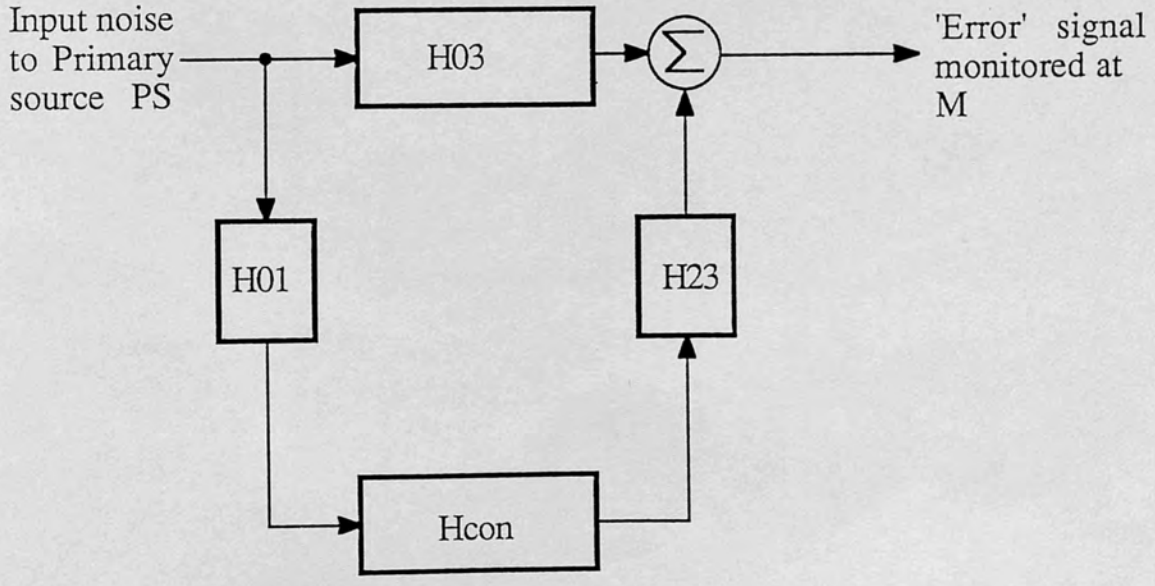


Figure 1.2 - Modelling requirements for a general ANC system

1.4.1 Acoustic feedback

When the ANC system in the duct is in operation, the controlling signal from the secondary source will propagate down the duct towards the monitor where it will destructively interfere with the primary noise as intended. It is inevitable however that at least part of this controlling signal will also propagate in the reverse direction and will be incident on the detector microphone. Thus, the system incorporates feedback in the form of the transfer function H_{21} (between the input to the secondary source and the output of the detector microphone). Figure{1.3} shows the more complete modelling requirements for the ANC system.

It has been shown [15] that, in the presence of feedback, the ideal controller has the transfer function H_c given by :

$$H_c = - \frac{H_{03}}{H_{01} H_{23} - H_{03} H_{21}} \quad \text{.....Eq.(1.2)}$$

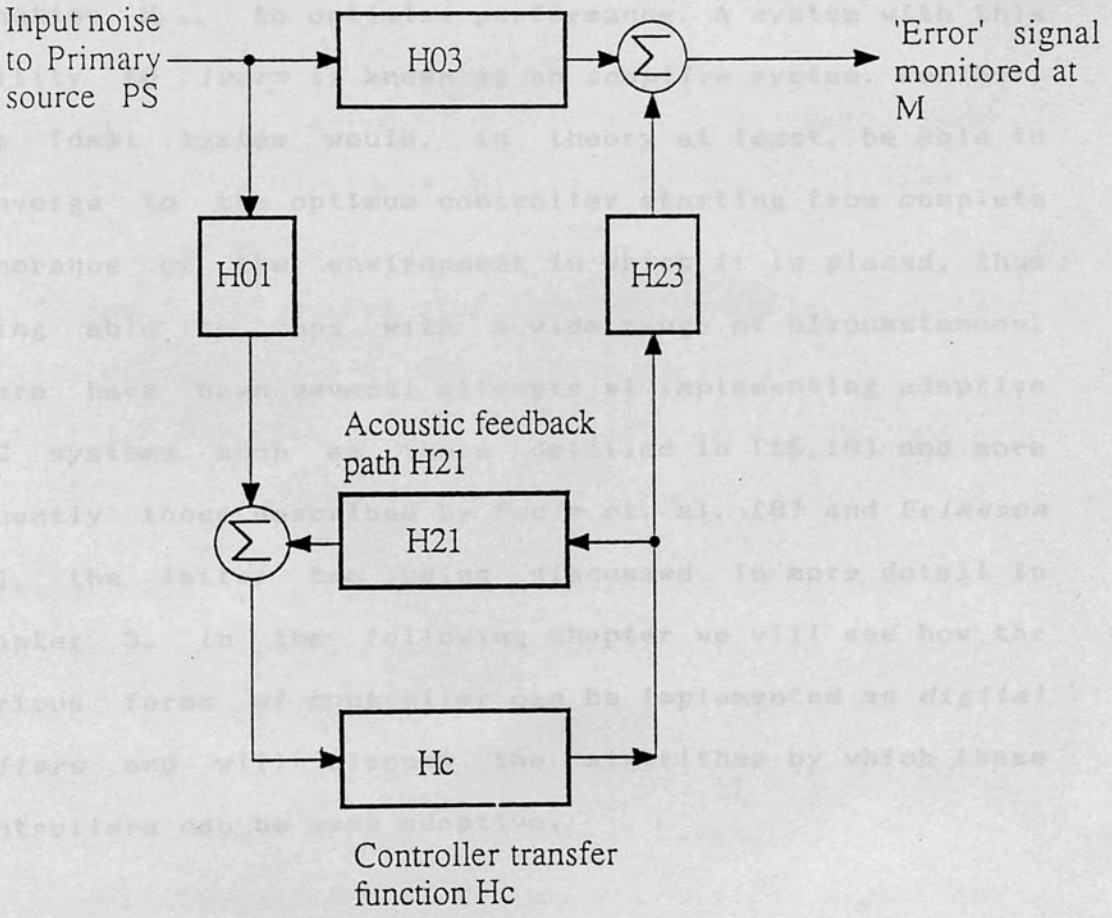


Figure 1.3 - Modelling requirements taking into account the acoustic feedback between secondary source (SS) and detector (D).

1.4.2 Adaptivity

Environmental changes in parameters such as temperature and air flow rate will inevitably cause changes in the transfer functions H_{01}, H_{03}, H_{21} and H_{23}. This in turn means that the performance of the ANC system will be adversely affected unless the controller is adjusted to suit the *new* duct. It is at this point that we can begin to introduce the concept of adaptivity of such a system. The ideal ANC system has the ability to compensate for such changes in the duct characteristics by continually adjusting the transfer function H_{con} to optimise performance. A system with this ability to *learn* is known as an *adaptive* system. Further, the ideal system would, in theory at least, be able to converge to the optimum controller starting from complete ignorance of the environment in which it is placed, thus being able to cope with a wide range of circumstances. There have been several attempts at implementing adaptive ANC systems such as those detailed in [15,16] and more recently those described by *Poole* et. al. [8] and *Eriksson* [9], the latter two being discussed in more detail in chapter 3. In the following chapter we will see how the various forms of controller can be implemented as *digital filters* and will discuss the algorithms by which these controllers can be made adaptive.

2.1 Introduction to digital filters

In recent years digital filters have been used to implement controllers in ANC systems [7,8,9,10,15,17,18,19]. Adaptive digital filtering has been a subject of research since the early seventies [20] and has advanced tremendously with the new processor technology, it is a widely publicized subject and many tutorial texts exist [21,22]. Unlike their analogue counterparts, digital filters are based on the simple operations of addition and multiplication and as such have several distinct advantages over conventional analogue filters :

(i) they can be implemented with software running on a general purpose computer and thus are relatively easy to *build* and test;

(ii) they do not suffer from drift as do analogue filters;

(iii) they are easier to modify than analogue filters;

(iv) they are far easier to understand than analogue filters;

Analogue and digital filters differ in the nature of the input and output signals. An analogue filter operates on analogue signals whereas a digital filter processes and generates digital data. The important difference between these two forms of data is that an analogue signal has a continuous independent variable and a digital signal has a discrete independent variable. In the analogue world we may describe a signal 's' as being 'a function of time' in which case we would usually refer to it symbolically as s(t). In the digital case we would refer to the signal as

s(k) where the symbol k is the discrete time independent variable and has integer values 0,1,2....etc. A general discussion of digital filtering is given by, for example, Williams [23].

2.2 Non recursive digital filters

Working in the time domain, the output generated by a non recursive digital transversal filter is simply the discrete time convolution of the input signal and the digital impulse response of the filter. Thus if we denote the discrete time filter input as

$$x(k) = x(0), x(1), x(2) \ldots\ldots\ldots etc$$

and the digital filter impulse response as;

$$a_k = a_0, a_1, a_2 \ldots\ldots\ldots a_{N-1}$$

for an (N tap) filter. This impulse response is referred to as the *Tap weights* of the filter.

The filter output is then denoted;

$$y(k) = y(0), y(1), y(2) \ldots\ldots\ldots etc$$

The filter output at discrete time interval n is expressed as follows :

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) \quad \ldots\ldots Eq.(2.1)$$

Thus the output is the weighted sum of the current input and the N-1 previous inputs.

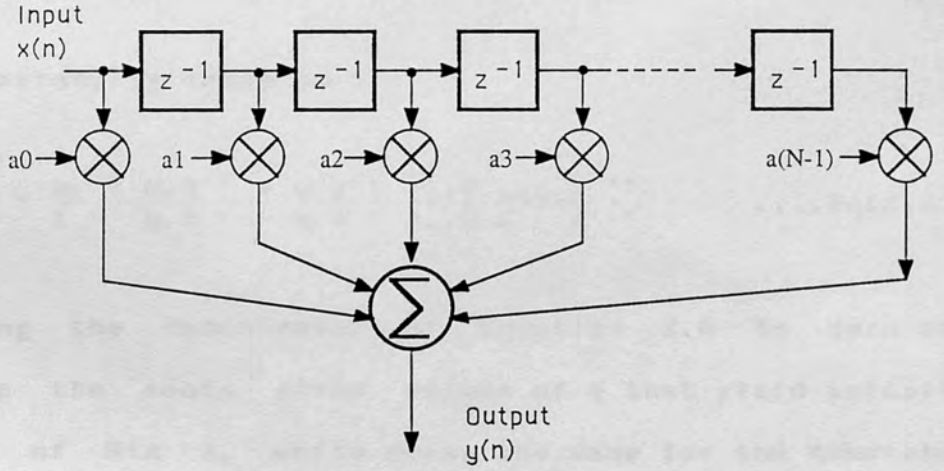
Such a filter has a Finite duration Impulse Response and is referred to as an FIR filter.

Using the standard nomenclature of the symbol z^{-1} representing a one sample delay, we can express the equation for the z-transform of an FIR filter as the following polynomial in z :

$$H(z^{-1}) = a_0 + a_1 z^{-1} + a_2 z^{-2} \ldots\ a_{N-1} z^{-(N-1)} \quad \ldots\ldots \text{Eq(2.2)}$$

for an N-tap filter.

The equation $H(z^{-1})=0$ has roots (of z) for which the transfer function is identically zero. Thus FIR filters are said to have *only zeroes* in their transfer functions and are often referred to as *all zero* filters. Figure{2.1} shows a diagrammatic representation of an all zero filter.



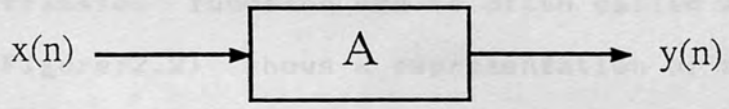


Figure 2.1 - A time domain all zero digital filter

2.3 Recursive digital filters

The output of an N-by-M recursive digital filter is the weighted sum of the current and the N-1 previous inputs

added to the weighted sum of the previous M-1 outputs. Thus in the same notation as before we can express the output of such a filter as :

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{j=1}^{M-1} b_i y(n-j) \quad \text{.....Eq.(2.3)}$$

A filter of this form has an Infinite duration Impulse Response and is referred as an IIR filter. Taking the z-transform of this equation and denoting,

$$\frac{Y(z^{-1})}{X(z^{-1})} = H(z^{-1})$$

and rearranging leads to :

$$H(z^{-1}) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots a_{(N-1)} z^{-(N-1)}}{1 - b_1 z^{-1} - b_2 z^{-2} - \ldots b_{(M-1)} z^{-(M-1)}} \quad \text{....Eq(2.4)}$$

Equating the denominator of Equation 2.4 to zero and finding the roots gives values of z that yield infinite values of $H(z^{-1})$, while doing the same for the numerator gives values of z for which $H(z^{-1})$ is identically zero. Thus an IIR filter is said to have *poles* as well as *zeroes* in its transfer function and is often called a *pole/zero* filter. Figure{2.2} shows a representation of a filter of this form.

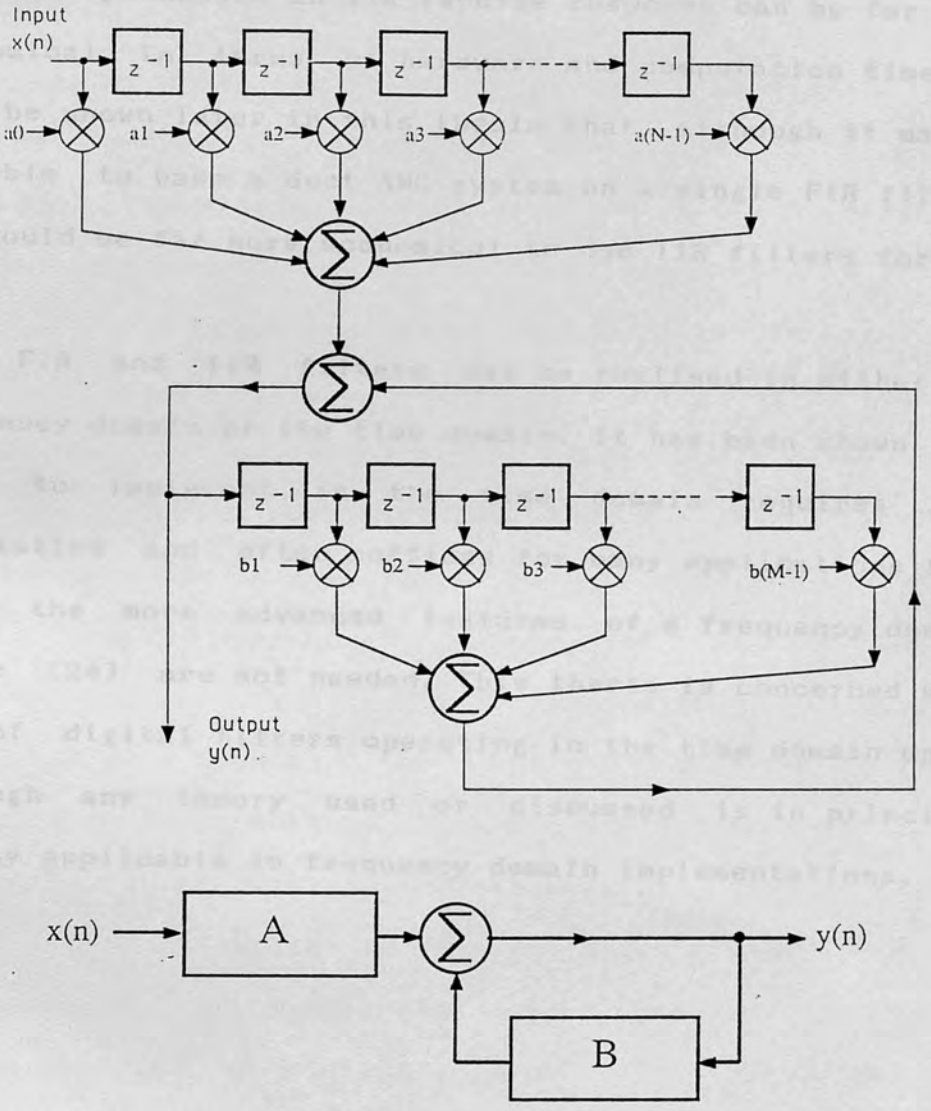


Figure 2.2 - A time domain pole/zero digital filter

2.4 Filter economy

FIR filters are inherently stable and perform very well in many signal processing applications [20,21]. However, there are instances where filter length required makes using an FIR filter an expensive and uneconomical method. IIR filters can be more economical in that by having just a few recursive taps we can obtain an overall impulse response which is much longer than the length of the filter itself.

Thus, to synthesize an IIR impulse response can be far more economical in terms of hardware and computation time. It will be shown later in this thesis that, although it may be possible to base a duct ANC system on a single FIR filter, it would be far more economical to use IIR filters for the job.

Both FIR and IIR filters can be realised in either the frequency domain or the time domain. It has been shown [24] that to implement in the time domain requires less computation and often suffices for many applications [25] where the more advanced features of a frequency domain filter [24] are not needed. This thesis is concerned with use of digital filters operating in the time domain only, although any theory used or discussed is in principle equally applicable to frequency domain implementations.

2.5 Adaptive digital filters

The design of a digital filter to perform a particular task requires *a priori* information about the data to be processed. The filter is optimum only when the statistical characteristics of the input data match the *a priori* information on which the design of the filter is based. An adaptive filter is one that is *self designing* in that it relies for its operation on a *recursive algorithm* [20], which makes it possible for the filter to perform satisfactorily in an environment where a complete knowledge of the signal characteristics is not available. Such a filter begins with a set of initial conditions and is able to *converge* towards the optimum solution in some statistical sense. Further, in a non-stationary environment such a filter offers tracking capabilities, whereby it can track time variations in the statistics of the input and (as we will see later) the output, provided that these variations are sufficiently slow. A wide variety of adaptive algorithms for both FIR and IIR filters have been developed in the last fifteen years [20,24,26,27,28,29], although the latter (the IIR adaptive filter) is far less well understood. The experimental work in this thesis compares the applicability of various algorithms [20,27,28,30] to adaptive ANC systems in ducts.

2.5.1 Adaptive filters in system identification

Let us suppose we have an unknown dynamic system and that we have a set of discrete time measurements, corresponding

to the output of the system for a given (known) input. We assume at this point that the unknown system is linear and does not vary with time. It is possible to identify the parameters of the unknown system by analysing the input and output data. Using very standard techniques such as least squares polynomial fitting [7], a model of the unknown system can be created. The task of an adaptive filter in such a situation, is to identify the unknown system's parameters by producing a model in the form of a filter as discussed in sections 2.2 and 2.3. The model filter must have *adjustable parameters*, these being progressively altered to converge the filter towards the ideal solution. This modelling situation is depicted in Figure(2.3) which shows an unknown system (H) being modelled by an adaptive filter. An *error signal* $\varepsilon(n)$ provides information concerning the discrepancy between the current transfer function of the adaptive filter and that of the unknown system, enabling the chosen algorithm to converge the adaptive filter by applying corrections to the individual filter taps. This error signal is derived from the difference between the filter output and the *desired* filter output d(n) (i.e the output from H_u).

$$\varepsilon(n) = d(n)-y(n) \quad \text{.....Eq.(2.5)}$$

It is the minimization of the mean square value of this error signal that is often used to define the optimum filter. A filter defined in such a way is termed a *Wiener* [20] filter in digital signal processing terminology. Once

the adaptive filter has converged to the Wiener solution, an important property is observed, namely that the cross correlation coefficient between the signals x(n) and ε(n) is zero.

This then is the basic idea for using an adaptive filter in system identification and we will now go on to look at various algorithms by which the adaptive filter taps can be adjusted to produce the desired effect. So far the discussion of adaptive algorithms has been kept very general, in the next section we will consider the the two types of filter introduced earlier i.e. FIR and IIR.

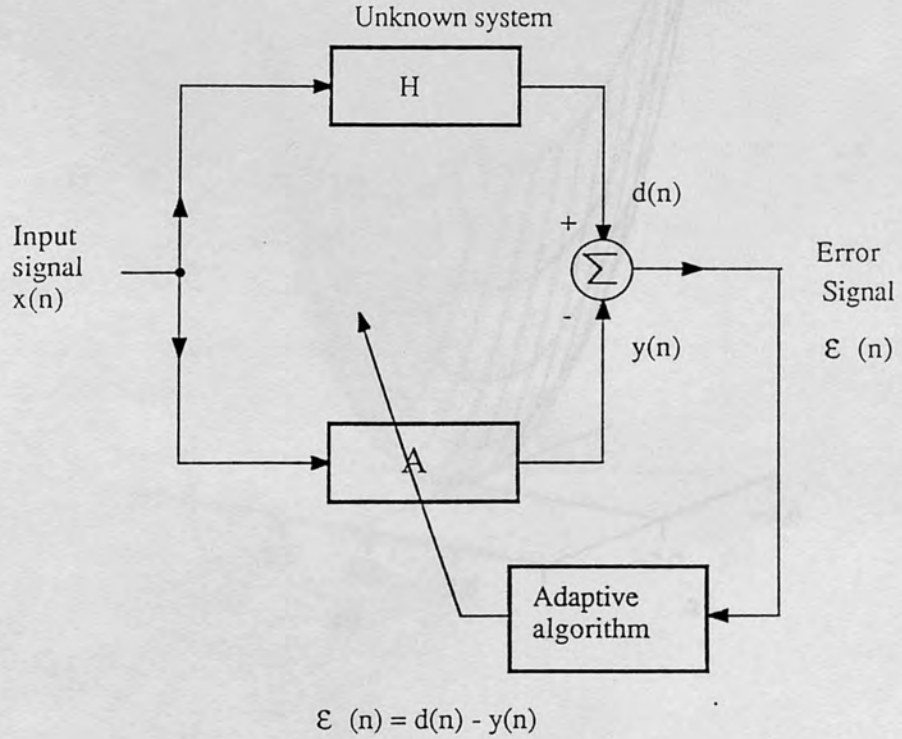


Figure 2.3 - An adaptive filter in system identification

2.6 FIR adaptive filters

It has been shown [20] that for an FIR filter, the Mean Square Error (MSE) is a quadratic function of the tap weight vector and thus has a unique minimum associated with it. For an adaptive filter, a plot of the Mean Square Error versus tap weights is known as the *error surface* or *performance surface* for the filter/algorithm. Clearly this can only be directly visualized in three dimensions for a two tap (coefficients a_0 & a_1) filter. Figure{2.4} shows the performance surface for such an FIR filter depicting the characteristic bowl shape.

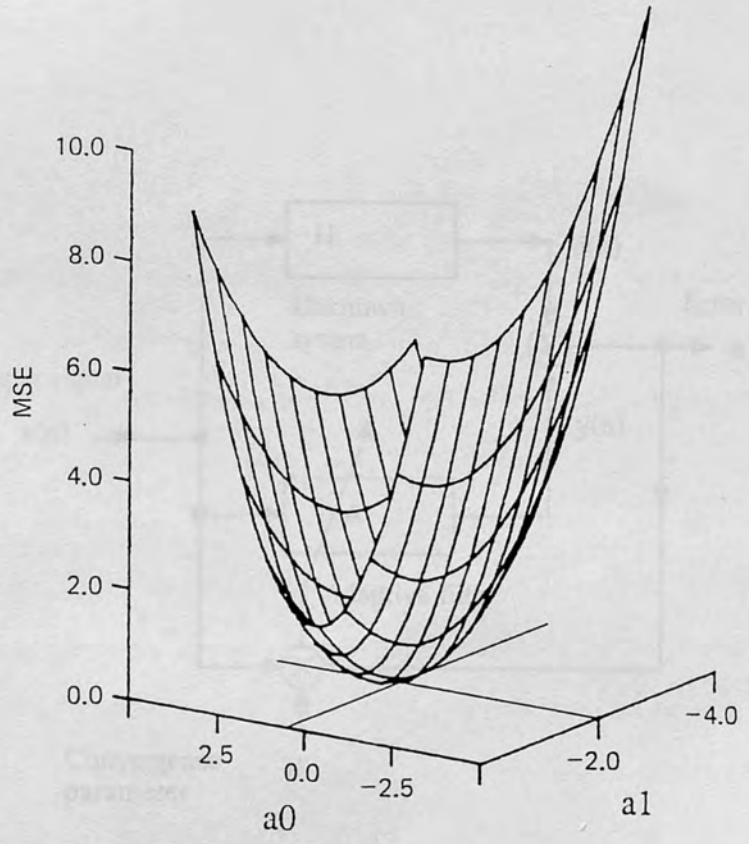
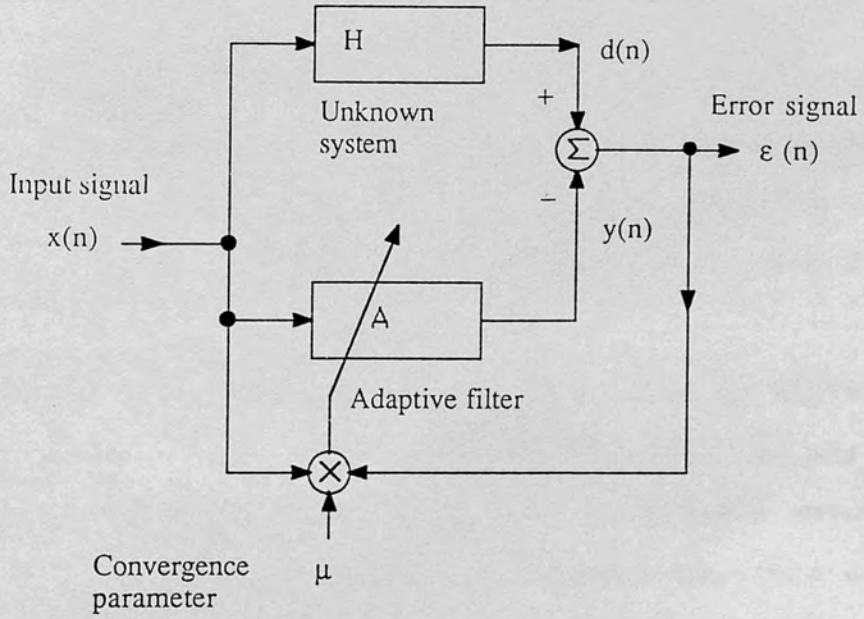


Figure 2.4 - The error surface for a two tap FIR LMS filter

Based on a simplified version of the method of steepest descent for solving equations [20], Widrow proposed his Least Mean Squares (LMS) algorithm for use in an adaptive FIR filter (see Figure 2.5). Such algorithms estimate the gradient of the performance surface and search for a turning point, thus finding the point of Minimum Mean Square Error (MMSE).



LMS Algorithm

Figure 2.5 - The Widrow LMS adaptive FIR filter

The algorithm is very well understood and used in many adaptive filtering situations. With reference to Figure{2.3}, the tap update equation is shown below :

$$a_i(n+1) = a_i(n) + \mu\varepsilon(n)x(n-i) \quad .....Eq.(2.6)$$

Where i=0 to N-1 for an N-tap filter.

and :

$a_0(n), a_1(n), a_2(n)....etc$

is the tap weight vector on completion of the nth iteration of the filtering/update cycle

$a_0(n+1), a_1(n+1), a_2(n+1)....etc$

is the tap weight vector after iteration n+1

and $\varepsilon(n)$ {=d(n)-y(n)} is the error signal described in section 2.5.1

μ is the convergence parameter used to scale the size of the individual updates to adjust the convergence rate.

Because of its simplicity, the Widrow LMS algorithm is very frequently used to implement FIR adaptive filters in real practical situations [8,31].

The rate of convergence of the LMS algorithm is dictated by the choice of the convergence parameter μ and for any given filtering problem this parameter will be given a value that will yield the most rapid convergence possible without leading to instability during adaptation or excessive random wandering (*misadjustment*) around the final value after convergence.

Griffiths [32] proposed a normalizing algorithm that adjusts the size of μ inversely proportional to the power in the input signal x(n).

$$\mu = \kappa \,/\, N\bar{x}^2(n) \quad \text{.....Eq.(2.7)}$$

where :

κ is a constant

N is the number of filter taps

$\bar{x}^2(n)$ is the power in the input signal x(n)

Normalization of some form or another would normally be used in any practical implementation of the LMS algorithm to ensure stability whilst maintaining an optimum convergence rate.

Computer simulations showing the performance of the LMS algorithm will be seen later in this thesis; evidence is presented that indicates that such a filter *can* be used to successfully form an ANC system for use in duct noise problems.

2.7 IIR adaptive filters

In section 2.4 some discussion was presented indicating the desirability of using IIR filters in certain situations. Clearly, it would be a further advantage to use an adaptive IIR filter in many such circumstances. However adaptive IIR filters exhibit some inconvenient characteristics not encountered in the FIR case : An IIR filter is unstable if it has a pole outside the unit circle on the complex z-plane. Thus, an adaptive IIR filter can become uncontrollably unstable if the movement of the poles during adaptation is not sufficiently governed. Further, the performance surface of an IIR filter is generally not quadratic and may have local minima [33]. Both these characteristics are serious disadvantages and consequently the use of adaptive IIR filters is very limited. Later, however, we will see some very convincing experimental evidence that in practice such adaptive filters can usefully be used for ANC in a duct.

Figure{2.6} shows the arrangement of an IIR adaptive filter (with direct element A and recursive element B) in an adaptive filtering environment, the filter is being used to match the output of an unknown (pole/zero) transfer function $H_{p,z}$. No attempt has been made here to indicate exactly how the error signal $\varepsilon(n)$ is manipulated in order to update the adaptive filter as this will be discussed with reference to specific algorithms in the next section. The reader should be aware of the fact that very little work has been published which *directly* compares the performance of IIR adaptive algorithms and it is hoped that

the experimental results shown in this thesis will lead to a greater appreciation of the similarities and differences between some of the simpler algorithms.

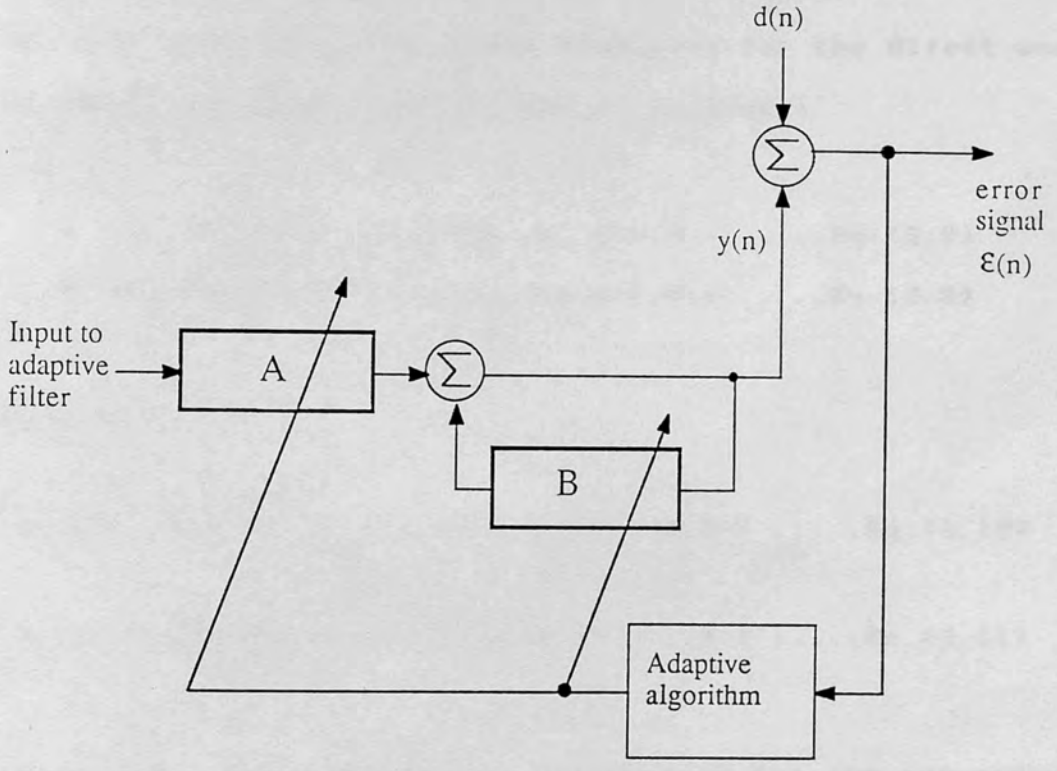


Figure 2.6 - An adaptive IIR filter in system identification

2.7.1 Simple IIR adaptive algorithms

We are concerned here with simple adaptive algorithms since only these can be realistically implemented in real time in the duct. We will begin by looking at the basic theory of an adaptive algorithm for IIR filters (a recursive LMS algorithm) as originally proposed by White [34]. Recall the

equation for the output of an IIR filter with N direct taps and M recursive taps (Equation 2.3):

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{j=1}^{M-1} b_i y(n-j) \quad \text{.....Eq.(2.3)}$$

White showed in his original publication [34] that, using the LMS approach, the update equations for the direct and recursive tap weight vectors are as follows :

$$a_i(n+1) = a_i(n) + \mu\varepsilon(n)\alpha_i(n) \quad i=0,N-1 \quad \text{.....Eq.(2.8)}$$

$$b_j(n+1) = b_j(n) + \mu\varepsilon(n)\beta_j(n) \quad j=1,M-1 \quad \text{.....Eq.(2.9)}$$

Where :

$$\alpha_i(n) = x(n-i) + \sum_{i=0}^{N-1} b_i(n)\ \alpha_i(n-1) \quad i=0,N-1 \quad \text{.....Eq.(2.10)}$$

$$\beta_j(n) = y(n-j) + \sum_{i=1}^{M-1} b_i(n)\ \beta_j(n-1) \quad j=1,M-1 \quad \text{.....Eq.(2.11)}$$

Figure(2.7) shows the filter arrangement for the recursive LMS algorithm in operation. With reference to Figure(2.7) it should be clear that to implement this form of the algorithm requires one N-point convolution and (N+M) M-point convolutions per iteration of the algorithm, which may well not be practical in many situations.

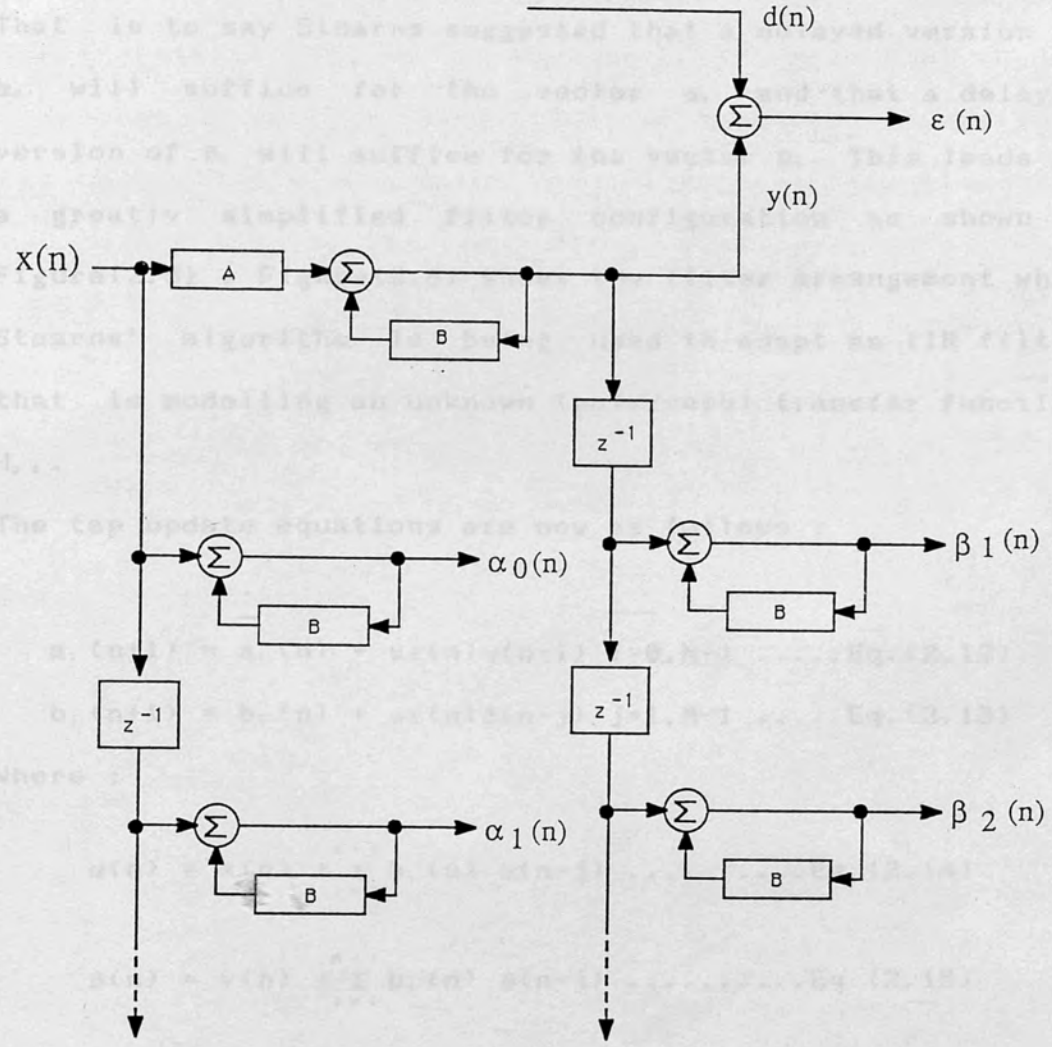


Figure 2.7 - The Recursive Least Mean Square IIR adaptive algorithm for adapting an IIR filter

2.7.1.1 Stearns' algorithm

A simplification of White's algorithm originally proposed by Stearns [30] is to make the assumption that :

$\alpha_i(n) \simeq \alpha_0 z^{-i}$

and

$\beta_j(n) \simeq \beta_i z^{-j}$

for *slow* convergence of the algorithm.

That is to say Stearns suggested that a delayed version of α_0 will suffice for the vector α_i and that a delayed version of β_1 will suffice for the vector β_j. This leads to a greatly simplified filter configuration as shown in Figure{2.8} . Figure{2.8} shows the filter arrangement when Stearns' algorithm is being used to adapt an IIR filter that is modelling an unknown (pole/zero) transfer function H_{pz}.

The tap update equations are now as follows :

$$a_i(n+1) = a_i(n) + \mu\varepsilon(n)\alpha(n-i) \quad i=0,N-1 \quad \text{.....Eq.(2.12)}$$

$$b_j(n+1) = b_j(n) + \mu\varepsilon(n)\beta(n-j) \quad j=1,M-1 \quad \text{.....Eq.(2.13)}$$

Where :

$$\alpha(n) = x(n) + \sum_{j=0}^{N-1} b_j(n)\,\alpha(n-j) \quad \text{.....Eq.(2.14)}$$

$$\beta(n) = y(n) + \sum_{j=1}^{M-1} b_j(n)\,\beta(n-j) \quad \text{.....Eq.(2.15)}$$

The input term used to both elements (direct and recursive) is *post filtered* through the recursive filter (1/1-B) prior to being used in the update equations. It has been demonstrated [28] that in many cases, this simplified algorithm behaves in an almost identical manner to that of White. In a more recent paper [33], Stearns provides evidence to show that if the adaptive IIR filter has at least as many poles and zeroes as does the plant that it is trying to model (i.e. it is a sufficient filter), that in many cases, the performance surface has only one minimum and is thus termed unimodal. Stearns' algorithm is directly

applicable to the duct ANC problem, all the necessary signals being available to the controller. Hsia [35] showed a further practical simplification to Stearns' algorithm that effectively eliminated the recursive convolution used when post filtering the input signal x(n).

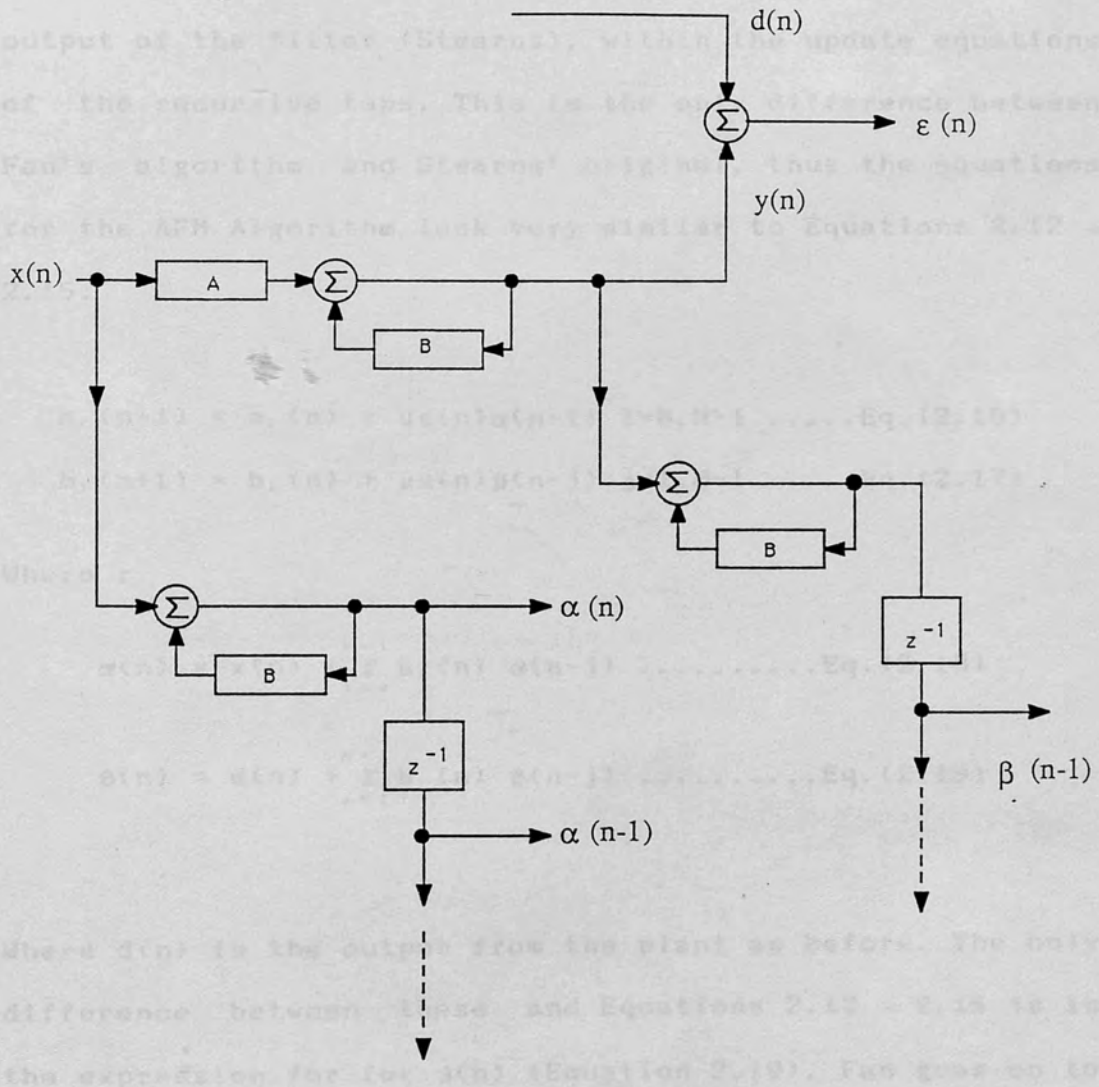


Figure 2.8 - Filter configuration for the use of Stearns' algorithm for adapting an IIR filter.

2.7.1.2 Fan's AFM algorithm

Stearns produced results that indicate that, when using an *insufficient* length filter, his simplified algorithm could not, in general, locate the global minimum on a multi-modal surface. In 1986 Fan [28] proposed his AFM (Adaptive Filtering Mode) IIR algorithm that he claims exhibits '*global convergence irrespective of local minima*' for cases where an *insufficient* length filter is used. In short Fan's new idea was to use the output of the plant, as opposed to the output of the filter (Stearns), within the update equations of the recursive taps. This is the only difference between Fan's algorithm and Stearns' original, thus the equations for the AFM Algorithm look very similar to Equations 2.12 → 2.15.

$$a_i(n+1) = a_i(n) + \mu\varepsilon(n)\alpha(n-i) \quad i=0,N-1 \quad \text{.....Eq.(2.16)}$$

$$b_j(n+1) = b_j(n) + \mu\varepsilon(n)\beta(n-j) \quad j=1,M-1 \quad \text{.....Eq.(2.17)}$$

Where :

$$\alpha(n) = x(n) + \sum_{j=0}^{N-1} b_j(n)\,\alpha(n-j) \quad \text{.....Eq.(2.18)}$$

$$\beta(n) = d(n) + \sum_{j=1}^{M-1} b_j(n)\,\beta(n-j) \quad \text{.....Eq.(2.19)}$$

Where d(n) is the output from the plant as before. The only difference between these and Equations 2.12 → 2.15 is in the expression for for $\beta(n)$ (Equation 2.19). Fan goes on to rigorously prove his claim for the case where his new idea is applied to Whites original algorithm.

Fan's algorithm cannot be implemented directly in the duct ANC system since the output of the plant is not directly accessible. This signal must be extracted from the error signal by subtracting away the filter output. Thus we have the signals $\varepsilon(n)$ and $y(n)$ which are both directly measurable and we would like to obtain the signal $d(n)$ given that $\varepsilon(n)=d(n)-y(n)$ as before.

The filter configuration for the use of this algorithm when modelling an unknown (pole/zero) transfer function $H_{p,z}$ is shown in Figure{2.9}.

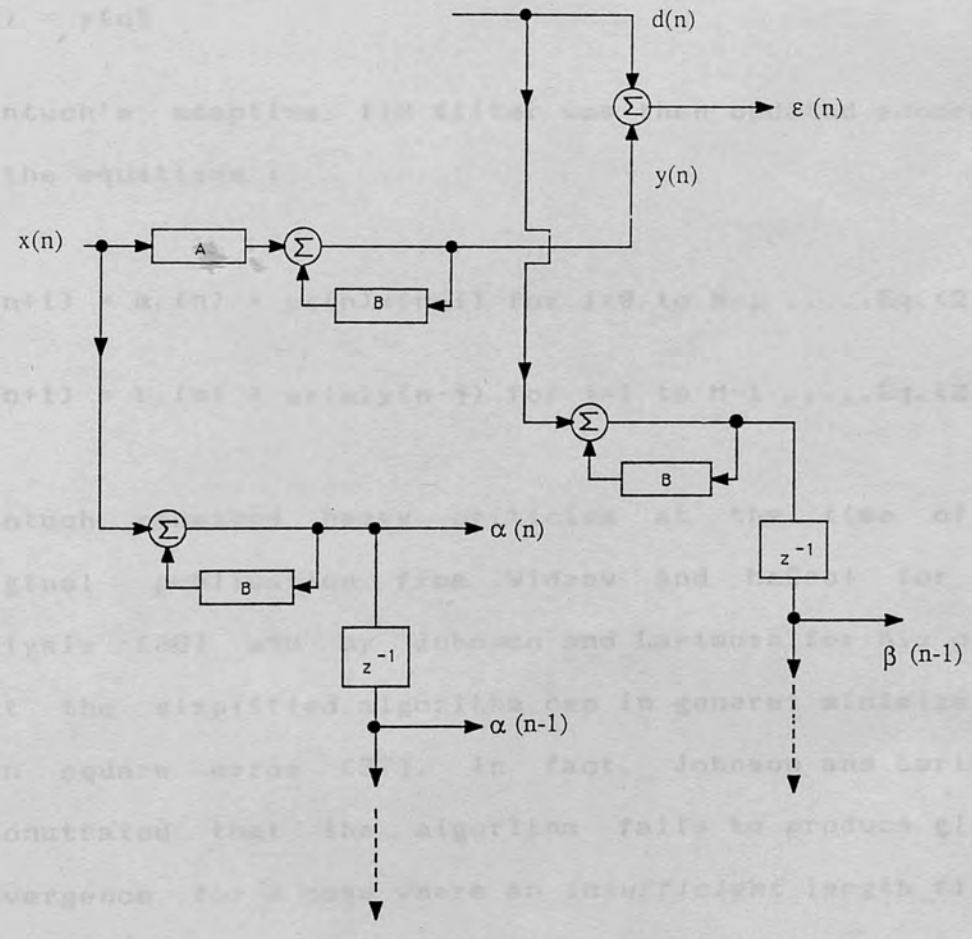


Figure 2.9 - Filter configuration for the use of Fan's AFM algorithm in used to adapt an IIR filter.

2.7.1.3 Feintuch's RLMS algorithm

The simplest of all IIR adaptive algorithms is the RLMS algorithm proposed by Feintuch [27] in 1976. Feintuch suggested that in practice there is no need for post filtering the signals x(n) and y(n) (or d(n)) and that to use the two signals y(n) and x(n) within the update equations will suffice. Thus he was suggesting that Equations 2.14 and 2.15 be replaced by :

$\alpha(n) = x(n)$

and

$\beta(n) = y(n)$

Feintuch's adaptive IIR filter was then updated according to the equations :

$$a_i(n+1) = a_i(n) + \mu\varepsilon(n)x(n-i) \text{ for } i=0 \text{ to } N-1 \quad \text{.....Eq.(2.20)}$$

$$b_j(n+1) = b_j(n) + \mu\varepsilon(n)y(n-j) \text{ for } j=1 \text{ to } M-1 \quad \text{.....Eq.(2.21)}$$

Feintuch received heavy criticism at the time of his original publication from Widrow and McCool for his analysis [36] and by Johnson and Larimore for his claim that the simplified algorithm can in general minimize the mean square error [37]. In fact, Johnson and Larimore demonstrated that the algorithm fails to produce global convergence for a case where an *insufficient* length filter was used, i.e. a case in which Feintuch never originally claimed success. Feintuch subsequently published experimental results [38] demonstrating the successful

performance of the algorithm for both *sufficient* and *over sufficient* filters.

Until Eriksson [9] in 1987, very few authors published any work involving the use of Feintuch's simple algorithm. However, computer simulations presented in this thesis do seem to show that the algorithm can not only be used usefully in IIR systems identification but can form the basis of an adaptive system for use in ANC in the duct. The use of Feintuch's simple algorithm when modelling an unknown (pole/zero) transfer function H_p is shown in Figure{2.10}.

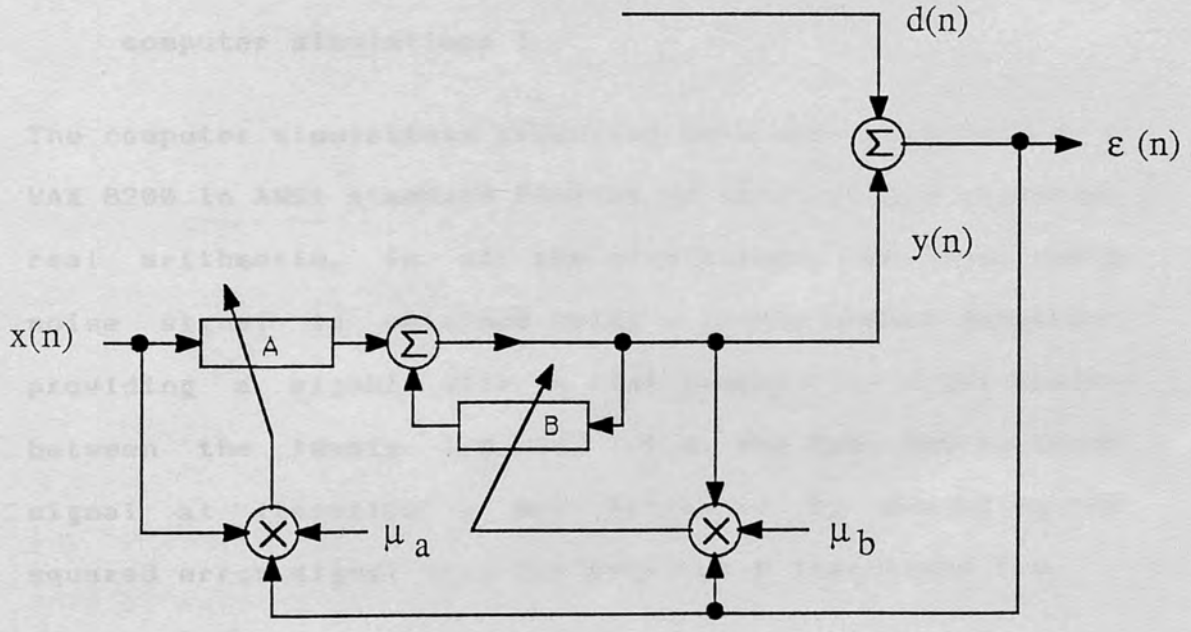


Figure 2.10 - Filter configuration for the use of Feintuch's RLMS algorithm for adapting an IIR filter

A development of Feintuch's simple algorithm is that known as the SHARF [26] algorithm in which the error signal $\varepsilon(n)$ is smoothed before being used to update the IIR filter according to Equations 2.20 and 2.21. However, appropriate choice of smoothing coefficients for different applications is impossible without *a priori* knowledge of the system being modelled so this type of algorithm is difficult to apply to a general ANC problem. One further recent piece of research by *Murali* and *Rao* [39] investigates the effects of modifying the gradient estimation of the adaptive filter but once again this approach is not applicable to ANC in the duct where we have no advanced knowledge of the plant transfer function and so cannot decide as to just *how* to modify the algorithm for optimum performance.

2.8 Algorithms' performance in system identification - computer simulations 1

The computer simulations presented here were performed on a VAX 8200 in ANSI standard FORTRAN 77 using single precision real arithmetic. In all the simulations, the input white noise signal is obtained using a random number generator providing a signal with a flat probability distribution between the levels 1.0 and -1.0. The Mean Square Error signal at iteration n was estimated by averaging the squared error signal over the previous P iterations i.e.

$$MSE(n) = \frac{1}{P} \sum_{j=0}^{P-1} \varepsilon^2(n-j)$$

Where P is typically 100. A plot of the Mean Square Error

(MSE) versus iteration number is referred to as the *learning curve* as it demonstrates the degree to which the adaptive filter has *learned* as a function of discrete time. A detailed description of the simulation programs used can be found in Appendices I & II of this thesis.

The following simulations directly compare the performance of the LMS algorithm for adapting an FIR filter and Stearns',Fan's and Feintuch's algorithms for adapting IIR filters.

The (system identification) performance of each of these algorithms was studied when adapting a filter to an IIR plant consisting of 2 direct taps and 2 recursive taps. The transfer function of the plant used in the simulations was that used by Johnson and Larimore [37] in their critiscism of Feintuch's algorithm in 1976.

This example plant was chosen in order that the results presented here can be compared with the previously published material [27,36,37,38].

The z-transform ($H(z^{-1})$) of the filter to be modelled was :

$$H(z^{-1}) = \frac{0.05 - 0.4z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}} \quad \text{.....Eq.(2.22)}$$

The plant has 1 purely real zero located at at 8.0 and 2 purely real poles located at 0.83030629 and 0.30109370 on the complex z-plane. The impulse response of this plant is shown in Figure{2.11} which shows the filter arrangement used to model the plant impulse response and the typical

decaying nature of such a overdamped Infinite Impulse Response.

In each experiment the adaptive filter began with a *zeroed* tap weight vector and is updated using a direct and recursive (where applicable) convergence coefficient of 0.01. Learning curves are given showing the Mean Square Error as a function of iteration number throughout each adaptation. The learning curves are plotted on a Decibel scale and the zero dB level corresponds to Mean Square Error level *before* adaptation is begun. The system is allowed to 'run' for 1000 iterations before adaptation is begun to allow time for transient effects to disappear and for the MSE to settle at its 0 dB level. In the case of the LMS algorithm, a plot of the impulse response of the adaptive filter after adaptation is also given with each learning curve in order that the truncation when using insufficient length filters can be seen.

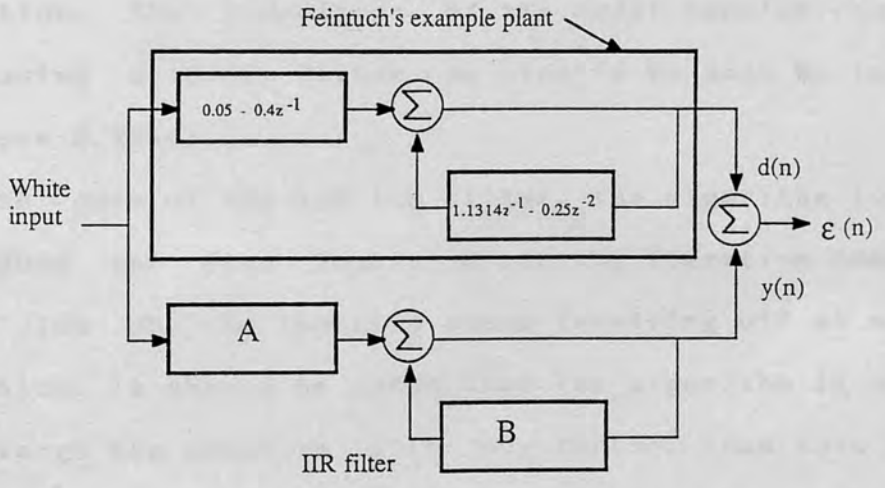


Figure 2.11 - Filter arrangement used to model the impulse response of the example IIR plant used by Johnson & Larimore in their criticism of Feintuch's RLMS algorithm.

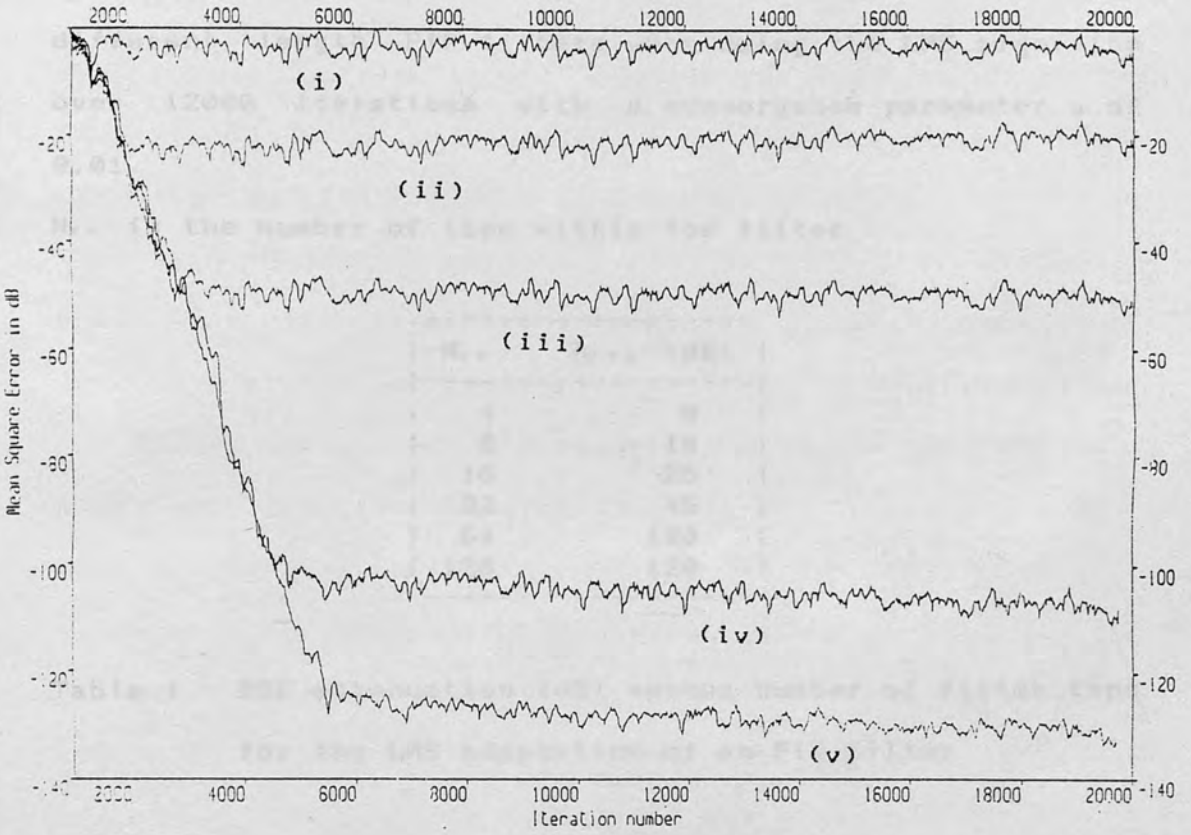
2.8.1 FIR adaptation - performance of the LMS algorithm

Figure (2.12) shows the learning curves and associated final adapted impulse responses for the LMS algorithm when used to adapt a 4,8,16,32,64 and 128 tap FIR filter (A only finite) to the above plant.

It can be seen from (2.12 (i)) that when an adaptive filter with 4 taps (i.e. the same total number of taps as the plant) is used to identify the plant, that the algorithm is unable to produce any significant attenuation in the MSE since the impulse response is truncated after 4 samples.

As the length of the filter is increased, the algorithm is able to lower the Minimum Mean Square Error as a greater duration of the required impulse response can be modelled. The comparison of the learning curves for the algorithm for the various length filters is summarized in Figure(2.12) which clearly shows the effects of impulse response truncation. The truncation of the model impulse response when using a 16 tap filter can clearly be seen by looking at Figure 2.12(a).

For the case of the 128 tap filter, the algorithm is able to reduce the Mean Square Error (by iteration 6000) by around 120 dB, the learning curve levelling off at around this value. It should be noted that the algorithm is unable to converge the adaptive filter any further than this since at this point the rounding error level of the machine machine on which the simulations were done has been reached.



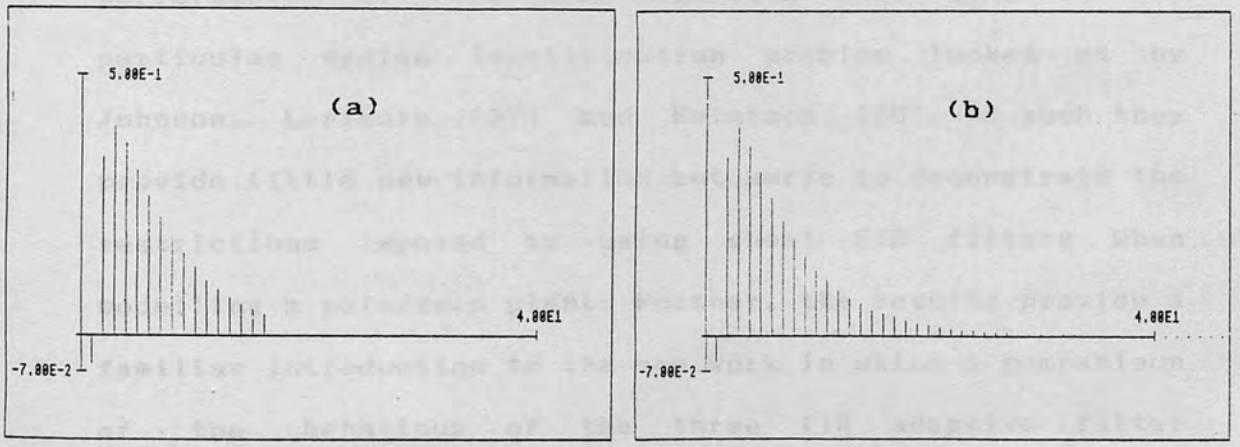


Figure 2.12 - Learning curves for the adaptation of a (i) 4, (ii) 8, (iii) 16, (iv) 32, (iv) 64 and (v) 128 tap FIR filters in system identification of Johnson and Larimores example plant using the Widrow LMS algorithm

Table I below shows a summary of the performance of the different length FIR filters when using the LMS algorithm over 12000 iterations with a convergence parameter μ of 0.01.

N_{FF} is the number of taps within the filter

N_{FF}	A_{TTN} (dB)
4	0
8	10
16	25
32	45
64	100
128	120

Table I - MSE attenuation (dB) versus number of filter taps for the LMS adaptation of an FIR filter

The results presented here demonstrate clearly the performance of the LMS algorithm when used in the particular system identification problem looked at by Johnson, Larimore [37] and Feintuch [38]. As such they provide little *new* information but serve to demonstrate the restrictions imposed by using short FIR filters when modelling a pole/zero plant. Further, the results provide a familiar introduction to the new work in which a comparison of the behaviour of the three IIR adaptive filter algorithms discussed in section 2.7.1. is made.

2.8.2 IIR filter adaptation

The following simulations compare the performance of the three IIR adaptive algorithms already discussed when adapting an IIR filter with 2 direct taps and 2 recursive taps (an exactly sufficient filter in this example). Results are also presented which compare the performance of the the algorithms when used to adapt an *over sufficient* IIR filter with direct and recursive elements that are more than eight times the required length.

Each adaptation was carried out for 100,000 iterations with a direct and recursive convergence parameter of 0.01. The filter tap weight vector is initially zeroed as with the LMS simulations. In the field of ANC the most important criterion for deciding whether or not a given algorithm is *suitable* is only that it converges the Mean Square Error. Thus, in the following experiments, the learning curves for the different algorithms are used to compare the performance of each. In order to compare the adaptation behaviour of the algorithms the trajectories of the coordinates (on the complex z-plane) of the zero and the poles of the adaptive filter are also given later.

2.8.2.1 Stearns' algorithm in system identification

Figure(2.13) shows the learning curve for the adaptation of a sufficient length IIR filter using Stearns' algorithm as discussed in section 2.7.1.1. The algorithm reduces the Mean Square Error by ≃ 120 dB by iteration 40,000 and to 130 dB by iteration 100,000 with the convergence parameters of 0.01 chosen.

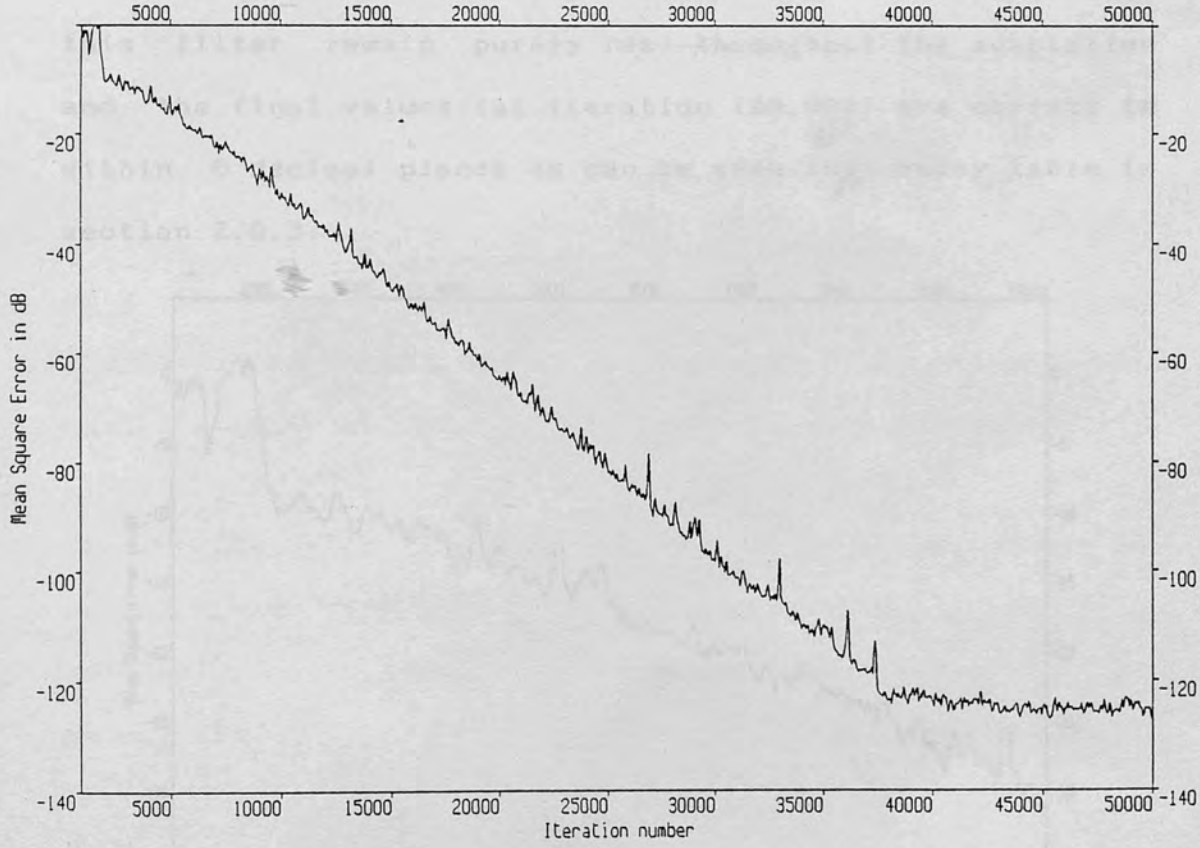


Figure 2.13 - Learning curve for the adaptation of an exactly sufficient length IIR filter using Stearns' adaptive algorithm. (system identification of Johnson & Larimore's example plant)

It is worth noting that since Stearns' algorithm uses the *filter* output to update the recursive element (Equations 2.13 & 2.15), the updates will initially be unreliable as the information used to update the filter is dominated by the output of the non-recursive element A. Consequently, the MSE may actually increase during the early stages of adaptation. Figure{2.14} shows the learning curve expanded during the first 10000 iterations of the algorithm, demonstrating that such an increase does indeed occur and that convergence only really began at about iteration 1800. The positions of the poles and the zero of this filter remain purely real throughout the adaptation and the final values (at iteration 100,000) are correct to within 6 decimal places as can be seen in the summary table in section 2.8.3.

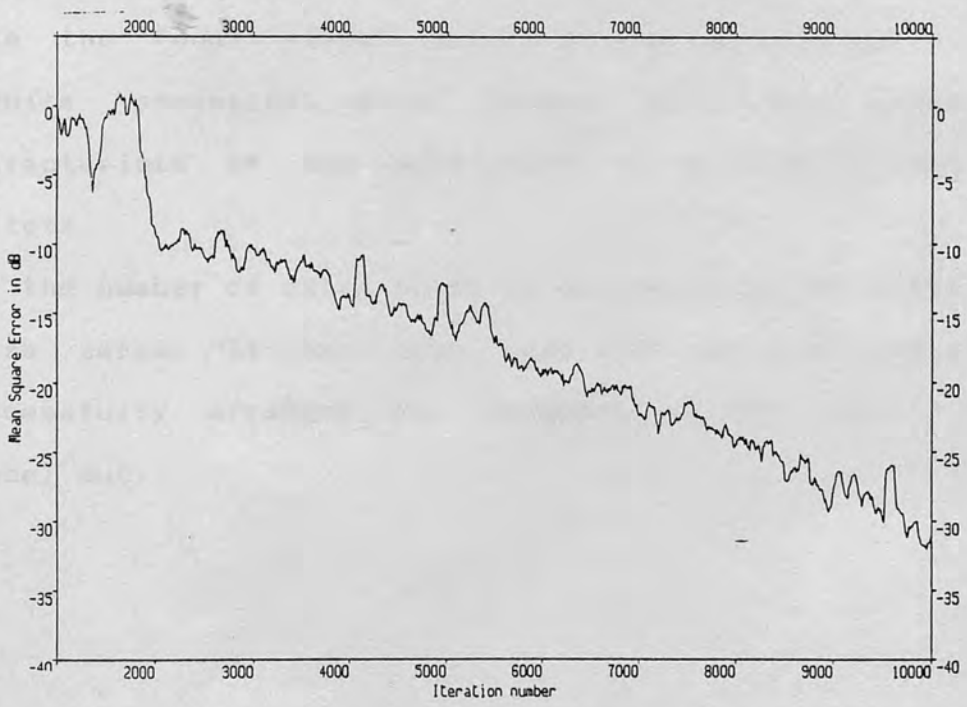


Figure 2.14 - Expanded view of the first 10000 iterations of the learning curve for a system identification example using Stearns' algorithm.

Stearns' algorithm using an over-sufficient filter

Figure (2.15a) demonstrates the increased MSE convergence rate obtained when using an over-sufficient filter (with the convergence parameter unchanged) to model the example plant and Figure(2.15b) shows the overall cancelling of the excess poles and zeroes on the complex z-plane after convergence. The increase in the convergence rate is to be expected since the algorithm has been given increased degrees of freedom with which to find a solution. The filter used here had 16 extra poles and 16 extra zeroes all of which are seen to cancel out exactly by forming pole/zero pairs when convergence to a minimum is achieved. The final overall transfer function is thus that of the 2-by-3 filter arrived at when using the exactly sufficient filter.

Note the 'dual slope' nature of the learning curve. The results presented here suggest that this shape is characteristc of the adaptation of over-sufficient IIR filters

If the number of extra poles is not equal to the number of extra zeros, it has been seen that the algorithm still successfully arranges the excesses so that overall they cancel out.

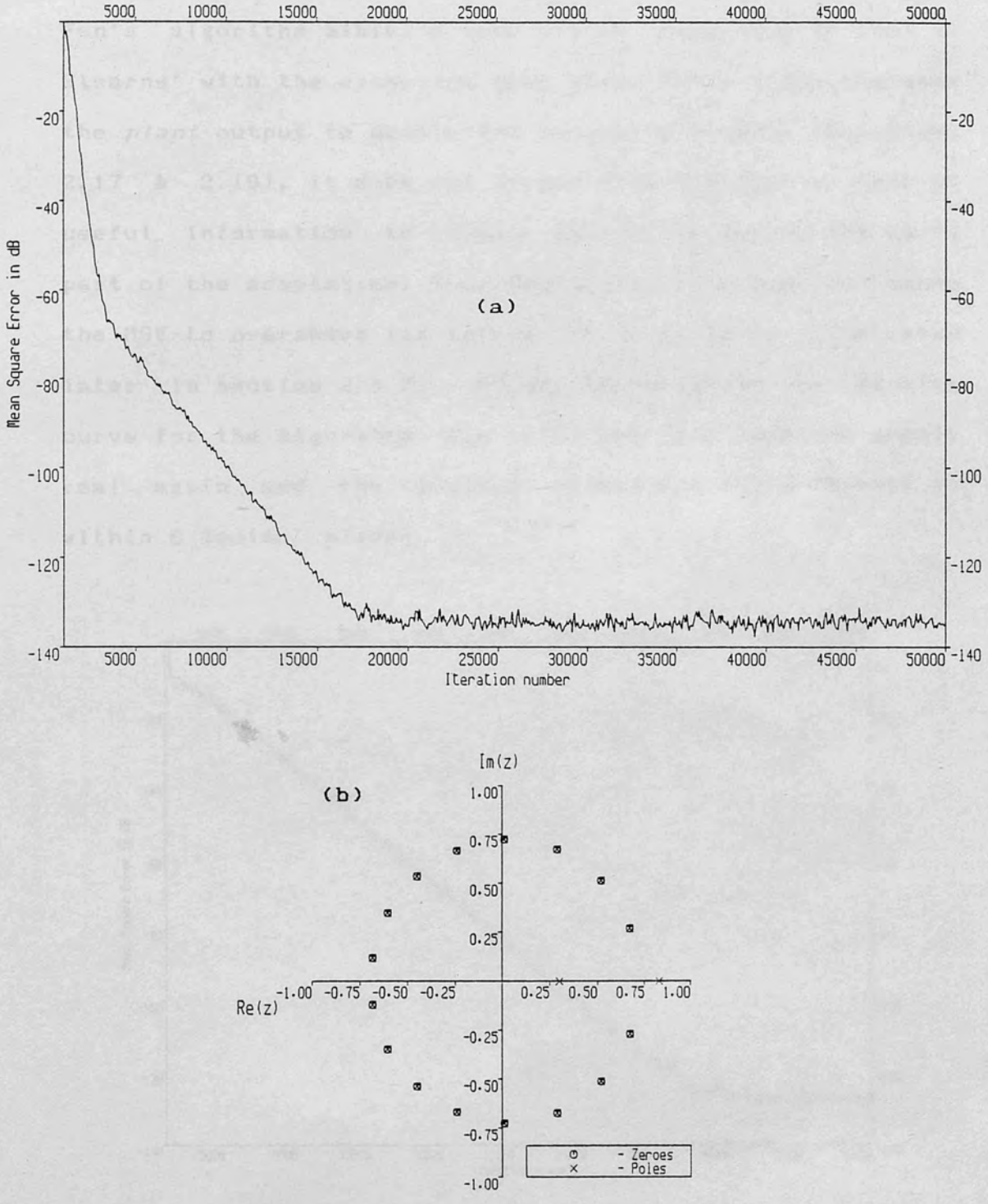


Figure 2.15 - Learning curve and pole/zero plot for a system identification example. Adaptation of an over-sufficient IIR filter with 16 extra poles and 16 extra zeroes, using Stearns algorithm.

2.8.2.2 Fan's algorithm in system identification

Fan's algorithm exhibits very similar behaviour to that of Stearns' with the exception that since Fan's algorithm uses the *plant* output to update the recursive element (Equations 2.17 & 2.19), it does not suffer from the initial lack of useful information to update the filter during the early part of the adaptation. Thus Fan's algorithm does not cause the MSE to overshoot its initial value (this is illustrated later in section 2.8.3) . Figure (2.16) shows the learning curve for the algorithm. The poles and zero remained purely real again and the eventual values are again correct to within 6 decimal places.

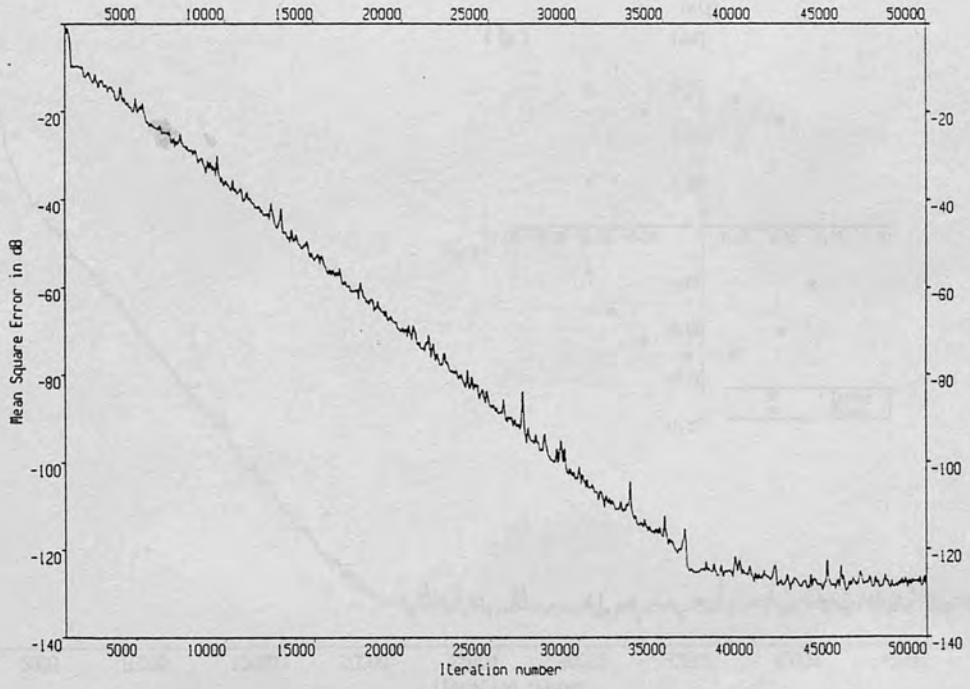


Figure 2.16 - Learning curve for the adaptation of an exactly sufficient length IIR filter in the system identification of Johnson & Larimore's example plant using Fan's AFM algorithm.

Fan's algorithm using an over-sufficient filter

As with Stearns' algorithm when given an over-sufficient model, Fan's AFM algorithm is able to increase the convergence rate and then arrange any excess poles and zeroes so that their combined effects cancel out. Figure(2.17) shows the increased convergence rate obtained (a) and (b) the overall cancelling (after convergence) of the excess poles and zeroes on the complex z-plane.

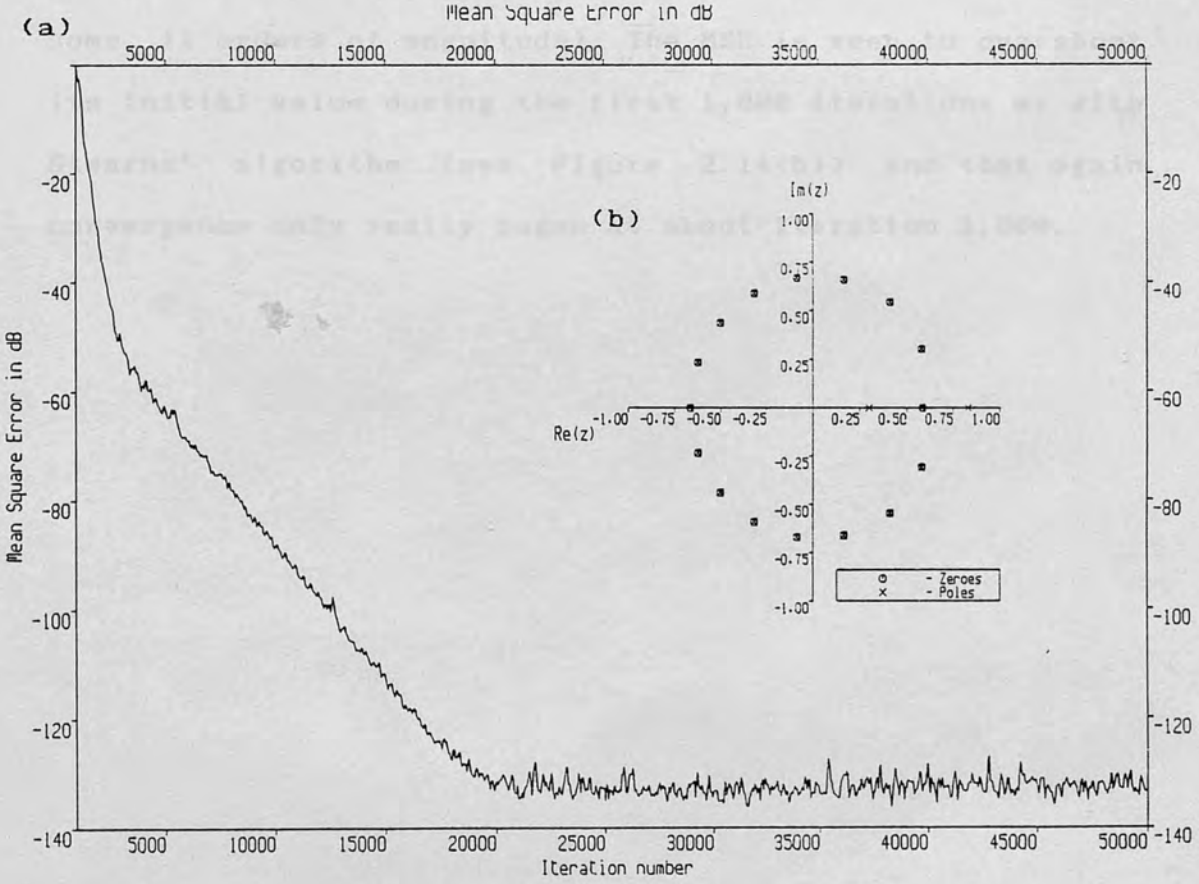


Figure 2.17 - Learning curve and pole/zero plot for the system identification adaptation of an over-sufficient IIR filter with 16 extra poles and 16 extra zeroes, using Fan's AFM algorithm.

2.8.2.3 Feintuch's RLMS algorithm in system identification

Figures (2.18a & 2.18b) show the learning curve for the case where Feintuch's RLMS algorithm is used to update the IIR filter using Equations 2.20 & 2.21. Comparing Figures 2.13, 2.16 and 2.18(a) it can be seen that Feintuch's algorithm is unable to converge the MSE to quite the same level as do the previous two algorithms. However, the algorithm produces ≈ 110 dB MSE reduction and therefore performs perfectly well (the 20 dB difference is insignificant since the MSE has already been reduced by some 11 orders of magnitude). The MSE is seen to overshoot its initial value during the first 1,800 iterations as with Stearns' algorithm (see Figure 2.14(b)) and that again convergence only really began at about iteration 1,800.

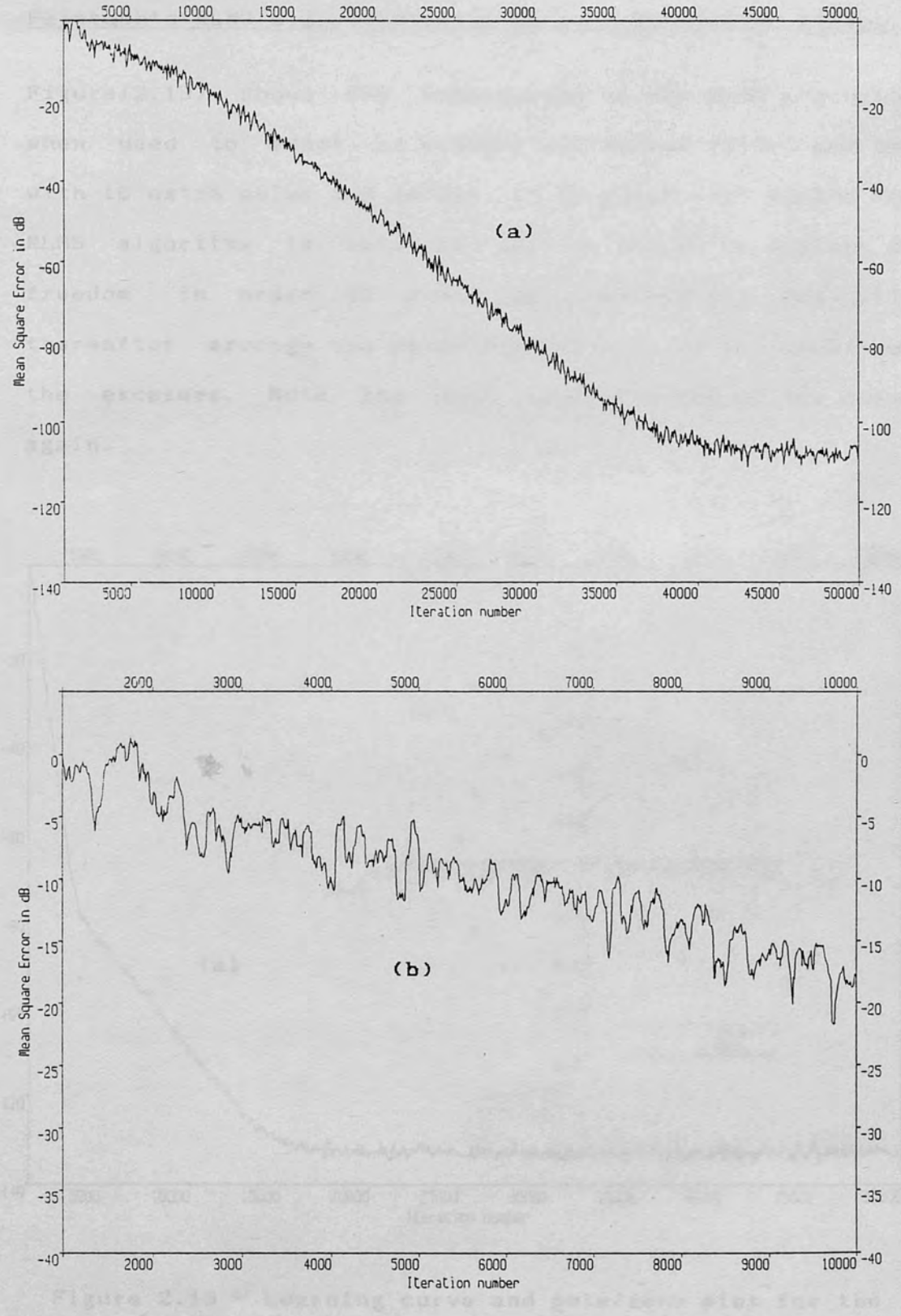


Figure 2.18 - Adaptation of an exactly sufficient length IIR filter in the system identification of Johnson & Larimores example plant using Feintuch's RLMS algorithm.

Feintuch's RLMS algorithm using an over-sufficient filter.

Figure(2.19) shows the convergence of the RLMS algorithm when used to adapt an exactly sufficient filter and one with 16 extra poles and zeroes. It is clear that again, the RLMS algorithm is able to utilise the extra degrees of freedom in order to speed up convergence, but will thereafter arrange the poles and zeros so as to cancel out the excesses. Note the dual slope nature of the curve again.

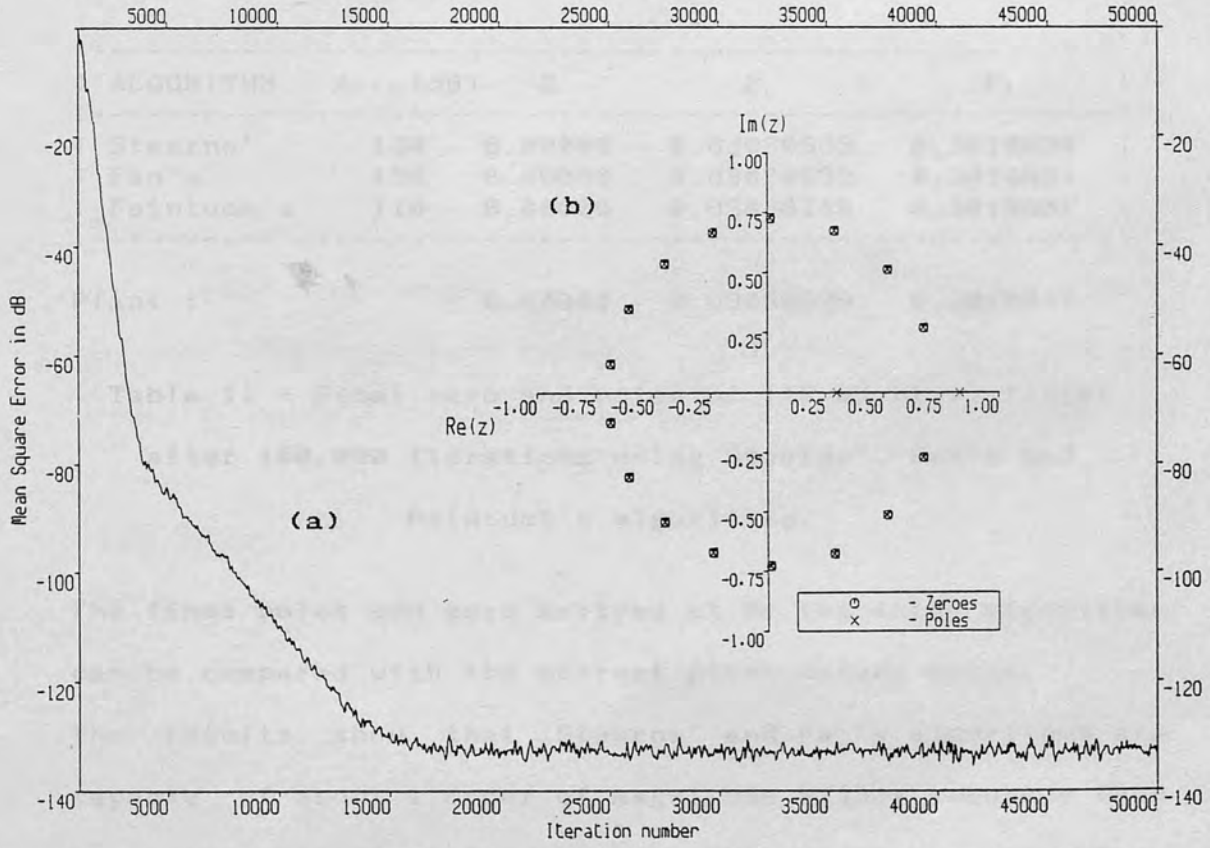


Figure 2.19 - Learning curve and pole/zero plot for the system identification adaptation of an over-sufficient IIR filter with 16 extra poles and 16 extra zeroes, using Feintuch's RLMS algorithm.

2.8.3 Summary of results for IIR system identification

Table II below shows a summary of the MSE attenuation and location of the final poles and zero values obtained after 100,000 iterations of each algorithm with both direct and recursive convergence coefficients of 0.01 with a sufficient length filter.

A_{TTN} is the MSE Attenuation in dB

Z is the real part of the position of the final zero

P_1 & P_2 are the real parts of the positions of the final poles

ALGORITHM	A_{TTN} (dB)	Z	P_1	P_2
Stearns'	130	8.00000	0.83030633	0.3010934
Fan's	130	8.00000	0.83030632	0.3010934
Feintuch's	110	8.00000	0.83030749	0.3010901
Plant :		8.00000	0.83030629	0.3010937

Table II - Final zero and poles of IIR adaptive filter after 100,000 iterations using Stearns', Fan's and Feintuch's algorithms.

The final poles and zero arrived at by the three algorithms can be compared with the correct plant values above.

The results show that Stearns' and Fan's algorithms are capable of about 1 order of magnitude higher accuracy than is Feintuch's simplified method. This difference may often be insignificant.

The conclusion drawn is that the further expense (in terms of complexity) of using Stearns' or Fan's methods is only justified in this particular example if very high accuracy

is demanded. Feintuch's algorithm has performed very well, in contrast to previous authors' published comments and results [36,37].

The three algorithms understandably behave differently during the early stages of adaptation as can be seen in Figure{2.20} in which the three learning curves for the first 10000 iterations are superimposed: (a) Feintuch's RLMS algorithm, (b) Stearns' algorithm and (c) Fan's AFM algorithm.

Figure{2.21b) shows the trajectories of the real part of the poles showing convergence to 0.3010 and 0.8303 and showing very clearly the difference in convergence rate of the three algorithms. The zero trajectories are not shown since they conatain no useful new information.

Results have shown that all three algorithms are able to make good use of any over-sufficiency in the model filter in speeding up convergence. All three algorithms are equally successful at cancelling out any excess poles and zeroes.

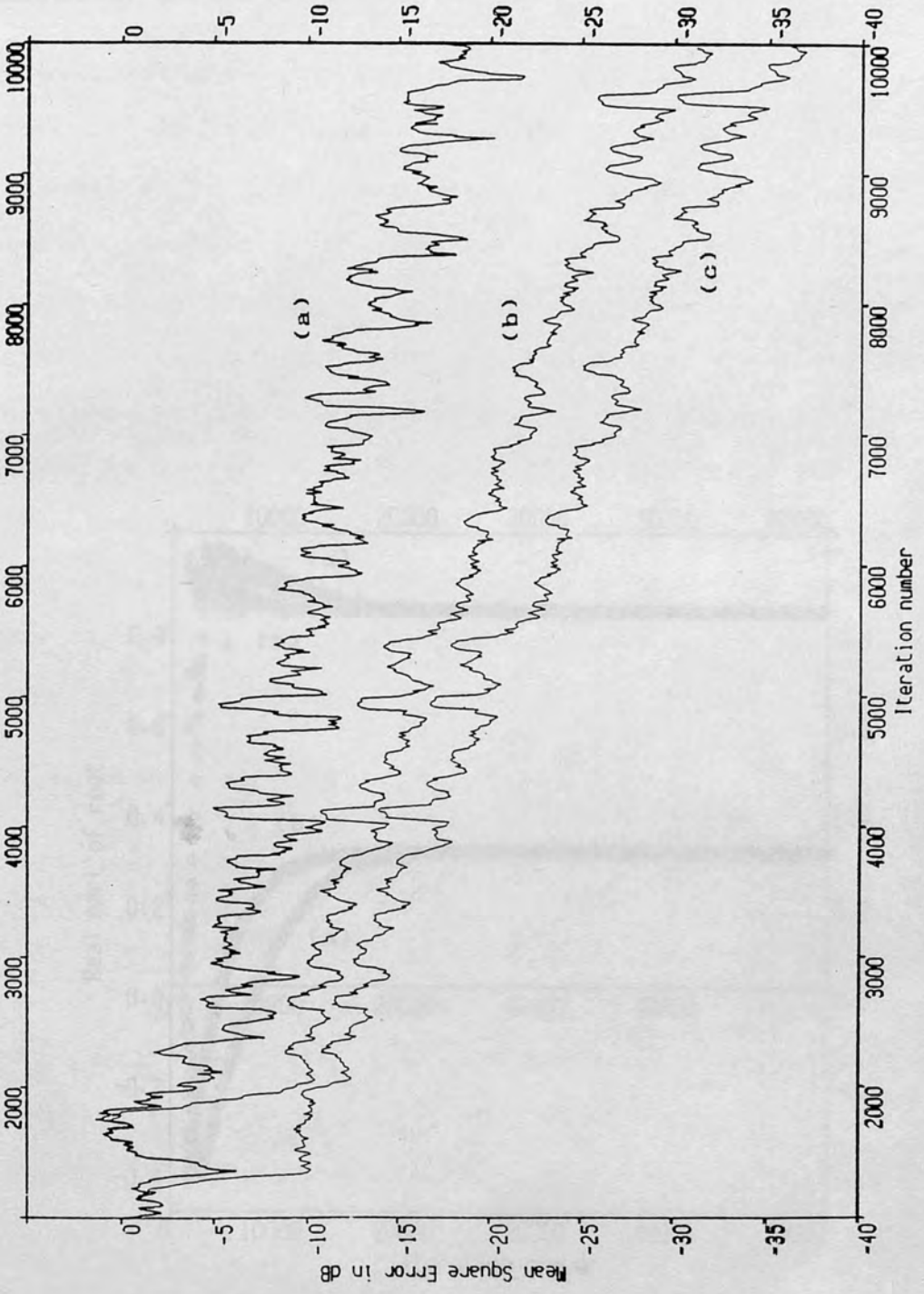


Figure 2.20 - First 10000 iterations of the learning curves for the three simple IIR adaptive algorithms in a system identification example superimposed to highlight the differences in convergence behaviour. (a) Feintuch's RLMS (b) Stearns' algorithm (c) Fan's AFM algorithm

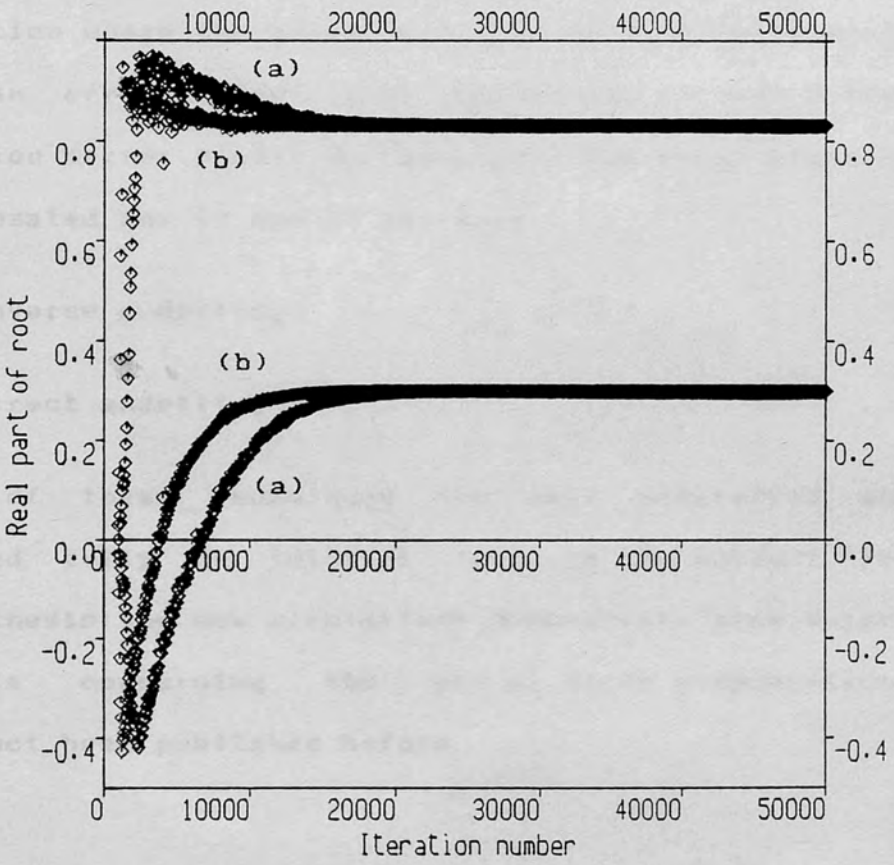


Figure 2.21 - Pole trajectories for the three simple IIR adaptive algorithms in system identification superimposed to highlight the differences in convergence behaviour. (a) Feintuch's RLMS and (b) Stearns' and Fan's algorithms indistinguishable

2.9 Error compensation

In many situations it may not be possible to sample signals directly at the summer (see Figure {2.22}). Instead the residual error signal may have to be monitored after it has passed through a further *error plant*. The eventual error signal $\varepsilon'(n)$ used by the algorithm will be incorrect unless modified to account for the presence of the error plant. This can be remedied by the use of a widely used technique known as error compensation [8,9]. Figure{2.22} shows a situation where the adaptive filter is faced with having to use an error signal that has passed through a transfer function (error plant) H_e. Basically the error plant can be compensated for in one of two ways :

i) inverse modelling

ii) direct modelling

Both of these techniques are well understood and are covered fully in tutorial texts on the subject [20]. In this thesis the new simulations demonstrate some surprising results concerning the use of error compensation that have not been published before.

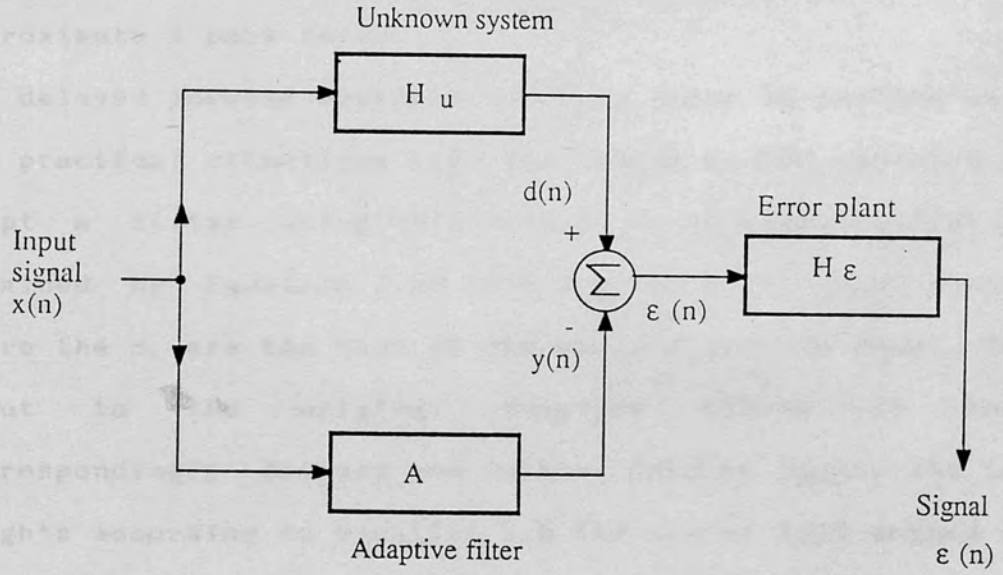


Figure 2.22 - System identification with an unavoidable 'error plant' H_ε

2.9.1 Inverse modelling

This involves placing a further filter in cascade with the error plant that is the best fit to the inverse of the error plant (H^{-1}_ε). The inverse model will have poles at the locations of the zeroes of the error plant and zeroes at the locations of the poles of the error plant. Thus the transfer function of the error plant in cascade with the

inverse model will be unity. It has been shown however, [20] that if the error plant is non-minimum phase (i.e. it has zeroes outside the unit circle on the complex z-plane) the inverse model will need to have poles outside the unit circle. If the filter is causal it will be unstable : Hence it must be non-causal to maintain overall stability of the cascaded pair. It is therefore unrealisable. However, Widrow demonstrates that an adequate stable causal inverse may be obtained by making the whole error system approximate a pure delay.

The delayed inverse approach has been shown to perform well in practical situations [8]. When using an LMS approach to adapt a filter using this delayed error signal $\varepsilon''(n)$ is obtained by Equation 2.23 (for a Q-tap error plant model) where the c_i are the taps of the delayed inverse model. The input to the original adaptive filter is then correspondingly delayed and is then used to update the tap weights according to Equation 2.6 the use of this method of compensation has been demonstrated by Poole et. al. [8] . Figure{2.23} shows the filter arrangement for this type of compensation.

$$\varepsilon''(n) = \sum_{i=0}^{Q-1} c_i \; e'(n-i-k) \quad \text{.....Equ.(2.23)}$$

Where k is the number of samples delay in the delayed inverse model.

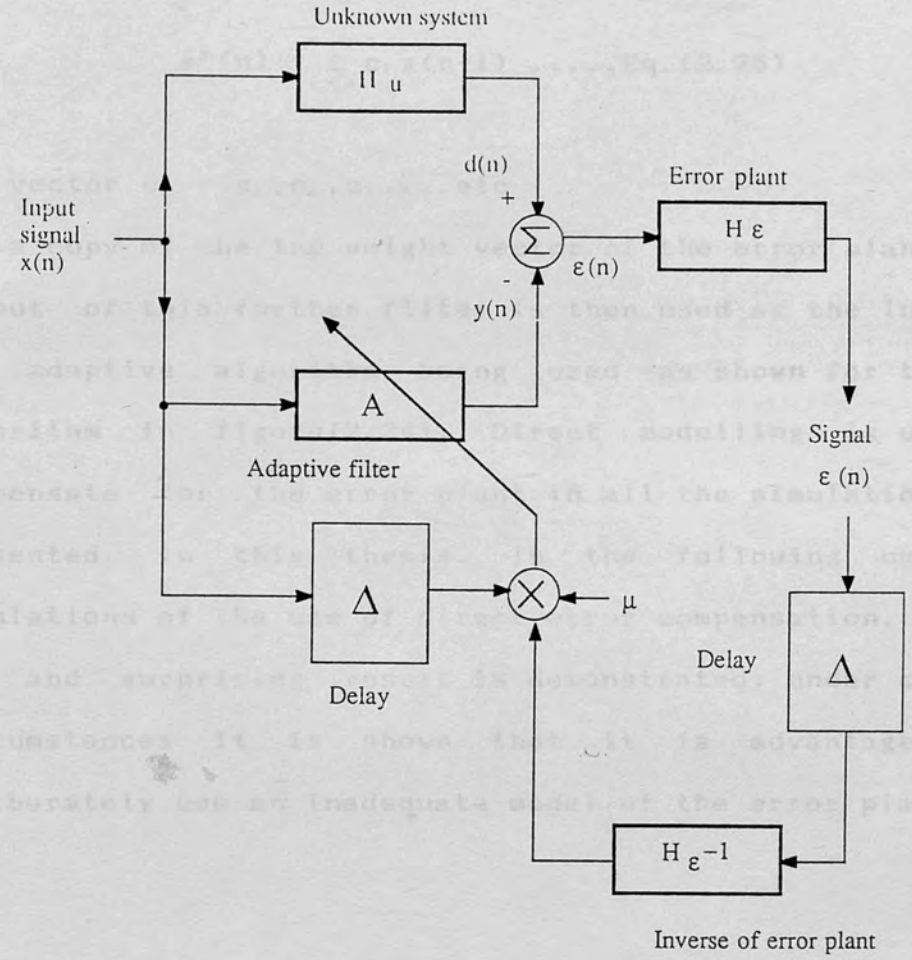


Figure 2.23 - Error plant compensation using the inverse modelling technique.

2.9.2 Direct modelling

This involves another filter whose transfer function is the best estimate (H'_{ε}) of the error plant H_{ε}. The input signal x(n) is post filtered through the model of the error plant to produce the signal x'(n) (as in Equation 2.25) that is used to update the filter taps using Equation 2.24).

$$a_i(n+1) = a_i(n) + \mu\varepsilon'(n)x'(n-i) \quad \text{.....Eq.(2.24)}$$

Where :

$$x'(n) = \sum_{i=0}^{N-1} c_i x(n-i) \quad \text{.....Eq.(2.25)}$$

The vector $c_i = c_1, c_2, c_3$....etc
is a copy of the tap weight vector of the error plant. The output of this further filter is then used as the input to any adaptive algorithm being used as shown for the LMS algorithm in figure{2.24}. Direct modelling is used to compensate for the error plant in all the simulation work presented in this thesis. In the following computer simulations of the use of direct error compensation, a both new and surprising result is demonstrated; under certain circumstances it is shown that it is advantageous to deliberately use an inadequate model of the error plant.

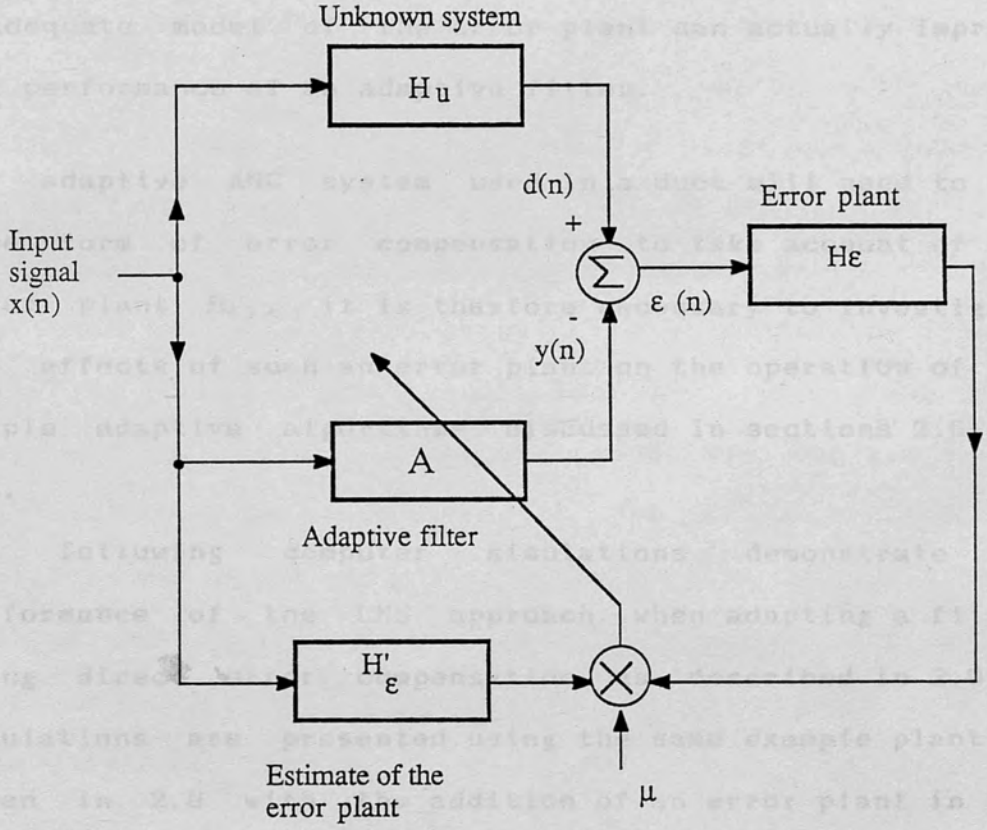


Figure 2.24 - Error plant compensation using the direct modelling technique.

2.10 The effects of direct error compensation - computer simulations 2

Results presented in this section demonstrate that when using the direct error compensation technique, using an inadequate model of the error plant can actually improve the performance of an adaptive filter.

Any adaptive ANC system used in a duct will need to use some form of error compensation to take account of the error plant H_{23}. It is therfore necessary to investigate the effects of such an error plant on the operation of the simple adaptive algorithms discussed in sections 2.6 and 2.7.

The following computer simulations demonstrate the performance of the LMS approach when adapting a filter using direct error compensation as described in 2.9.2. Simulations are presented using the same example plant as given in 2.8 with the addition of an error plant in the form of both an FIR filter and and IIR filter. The behaviour of the FIR LMS algorithm when used to adapt an FIR filter, and the three IIR adaptive algorithms already discussed when used to adapt IIR filters, are studied.

2.10.1 FIR adaptation using direct compensation for an FIR error plant

Using the filter arrangement shown in Figure{2.24} with a 128 tap FIR filter (A), the performance of the LMS algorithm combined with direct error compensation was investigated.

In this particular example, the error plant was chosen to have the form of a pure delay of 0,96 and 192 samples.

Figure{2.25} shows the three learning curves for the LMS algorithm (i) no error plant, (ii) 96 samples delay and (iii) 192 samples The results indicate clearly that when using the direct error compensation method, for the error plant in question consisting of pure delay, that the LMS algorithm performs well, producing again ≈ 120 dB MSE attenuation in all three cases. It is worth noticing the effect of the longer error plant on the convergence rate of the algorithm.

The above results demonstrate the effectiveness of direct error compensation as will be seen later in the simulations made for a duct ANC system.

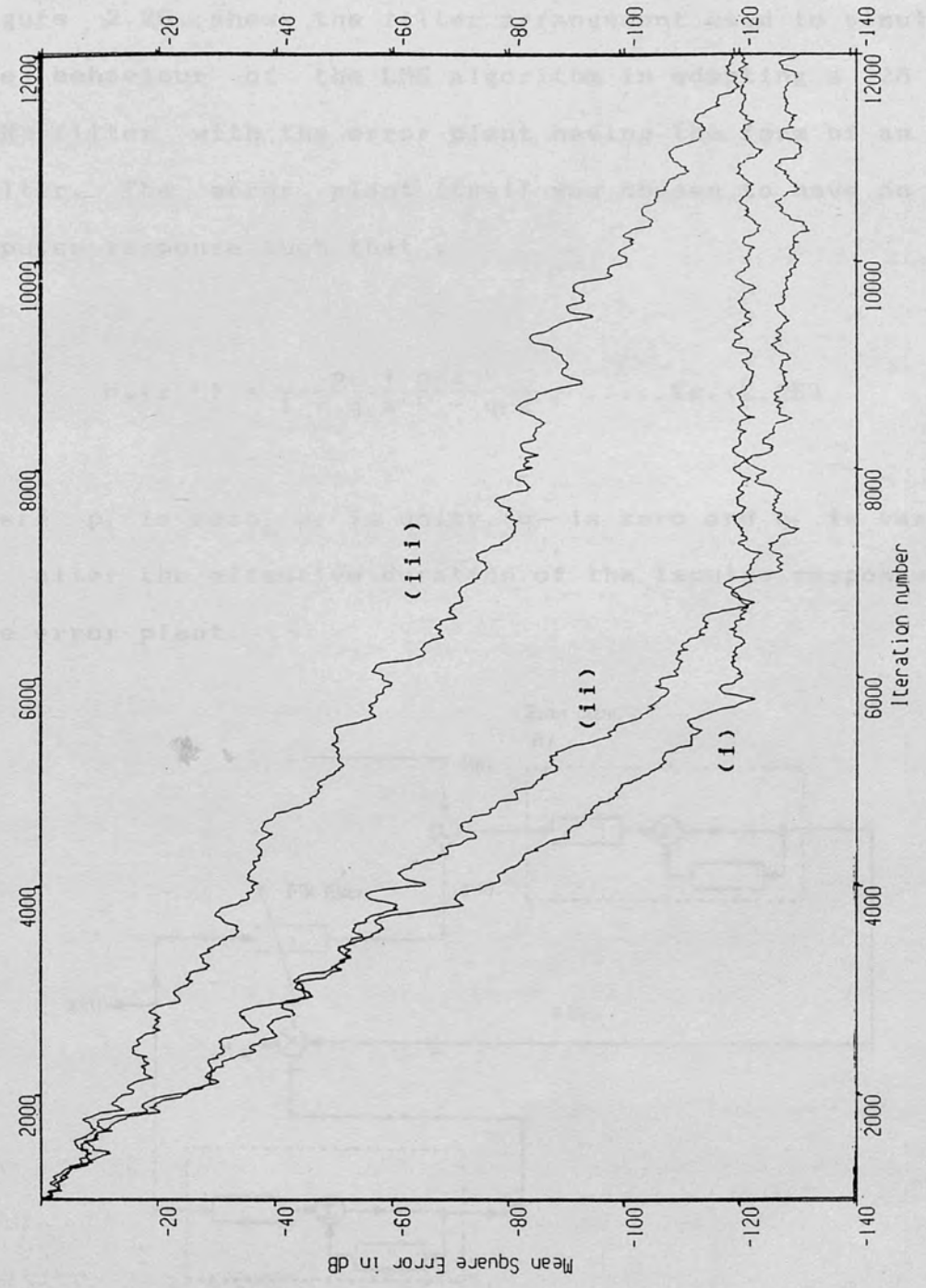


Figure 2.25 - Learning curves for the LMS algorithm adapting a 128 point FIR filter with an error plant consisting of (i) 0, (ii) 96, (iii) 192 step delays and using direct error compensation.

2.10.2 FIR adaptation using direct compensation for an IIR error plant

Figure 2.26 shows the filter arrangement used to simulate the behaviour of the LMS algorithm in adapting a 128 tap FIR filter with the error plant having the form of an IIR filter. The error plant itself was chosen to have an IIR impulse response such that :

$$H_\varepsilon(z^{-1}) = \frac{p_0 + p_1 z^{-1}}{1 - q_1 z^{-1} - q_2 z^{-2}} \quad \text{.....Eq.(2.25)}$$

Where p_0 is zero, p_1 is unity, q_1 is zero and q_2 is varied to alter the effective duration of the impulse response of the error plant.

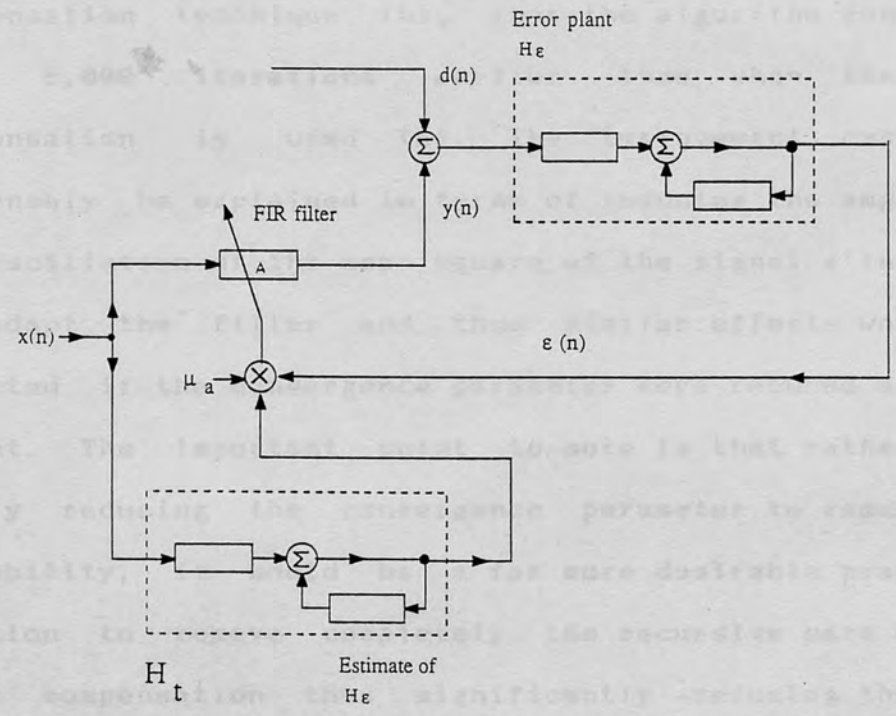


Figure 2.26 - Filter arrangement to examine the behaviour of the FIR LMS algorithm when use with direct compensation for an IIR error plant.

2.10.3 Inadequate compensation - improved performance

The compensation filter H_t and would ideally be an exact copy of the error plant model H_E (according to text book theory). However, if the recursive element of H_t is omitted, i.e. there is *inadequate* compensation, the LMS algorithm actually performs better in that, for a given convergence parameter, the rate of convergence is increased.

The learning curves in Figure 2.27 demonstrate the combined effects of increasing the length of the error plant itself *and* using both complete and inadequate compensation. Figure{2.27(i)} shows the learning curves for the algorithm when the error plant is comparatively short (with $q_2 = 0.4$), it is easy to see that when using the inadequate compensation technique (b), that the algorithm converges some 5,000 iterations earlier than when the full compensation is used (a). The improvement can only reasonably be explained in terms of reducing the amplitude of oscillation of the mean square of the signal x'(n) used to adapt the filter and thus similar effects would be expected if the convergence parameter were reduced a small amount. The important point to note is that rather than simply reducing the convergence parameter to remedy any instability, it would be a far more desirable practical solution to remove completely the recursive part of the error compensation thus significantly reducing the work load on the digital controller used.

It is worth noting that the initial convergence is in fact slower when using inadequate compensation since the two learning curves cross over at about iteration 4,000.

The improvement in performance when using the inadequate compensation technique is very marked in Figure{2.27 (ii)} where the actual error plant is made longer (q_2 =0.7). In this case the filter converges 10,000 iterations earlier when using the new inadequate technique (b). Again, the learning curves cross and the additional effect of increased misadjustment when using the inadequate compensation can be seen. An increase in misadjustment is of course to be expected since the algorithm has less detailed information with which to adapt the filter - the actual increase in misadjustment is in fact insignificant since it is of the order of ± 1 dB at around -120 dB.

This finding is a very important one since it means that the computation necessary to compensate for a long error plant (as in the highly reverberant duct) may well in fact be the same as that necessary for the anechoic duct!

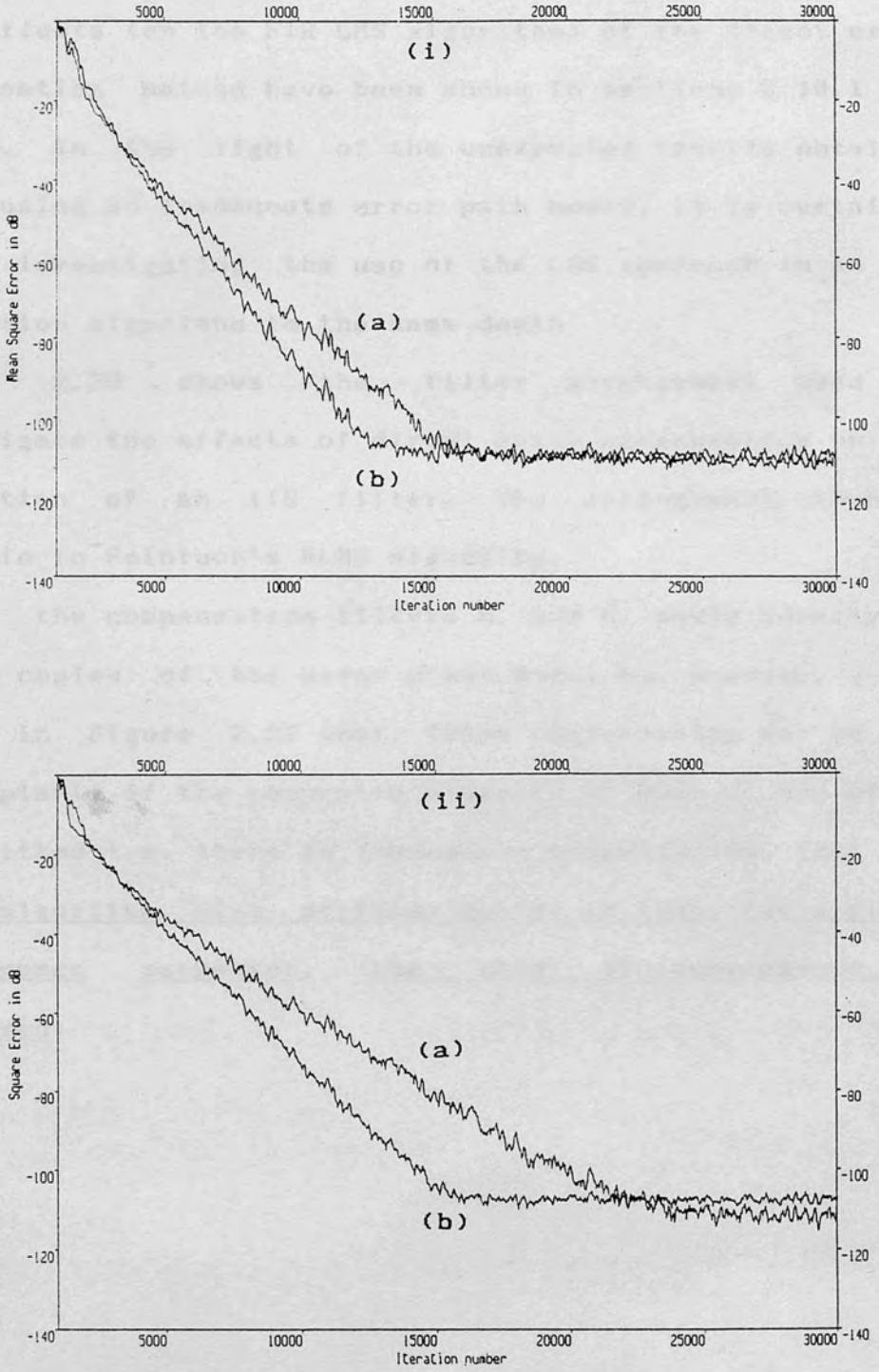


Figure 2.27 - Comparison of convergence of an FIR filter using the LMS algorithm (for 'full' and 'inadequate' error compensation) with (i) a short error plant and (ii) a long error plant

2.10.4 IIR adaptation using direct error compensation

The effects (on the FIR LMS algorithm) of the direct error compensation method have been shown in sections 2.10.1 and 2.10.2. In the light of the unexpected results obtained when using an inadequate error path model, it is certainly worth investigating the use of the LMS approach in an IIR adaptation algorithm to the same depth.

Figure 2.28 shows the filter arrangement used to investigate the effects of direct error compensation on the adaptation of an IIR filter. The arrangement shown is specific to Feintuch's RLMS algorithm.

Again, the compensation filters H_t and H_u would ideally be exact copies of the error plant model H_ε. However, it is shown in Figure 2.29 that, (when compensating for an IIR error plant) if the recursive elements of both H_t and of H_u are omitted i.e. there is *inadequate* compensation, that the RLMS algorithm also performs better in that, for a given convergence parameter, the rate of convergence is increased.

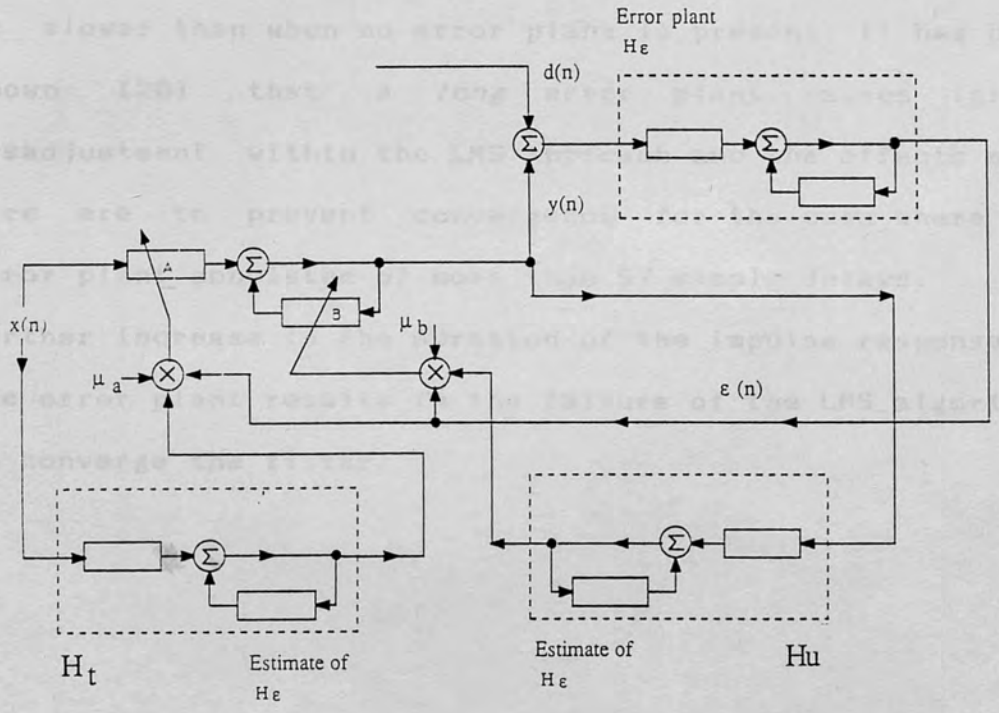


Figure 2.28 - Filter arrangement used to investigate the effects of direct error plant compensation on the Feintuch RLMS algorithm for adapting an IIR filter.

2.10.4.1 RLMS algorithm (FIR error plant)

Figure{2.29} shows learning curves for the case where Feintuch's RLMS algorithm is used to adapt an IIR filter with 2 direct taps and 3 recursive taps and with an error plant consisting of (a) 48 and (b) 57 sample delays. The results show that for the case of the 57 sample delay

(b), the learning curve is very 'spikey' due to increased misadjustment and it was found that increasing the delay further resulted in the algorithm eventually causing the MSE to blow up. For the case of the 48 sample delay (a), the algorithm produces convergence but again there is a marked increase in misadjustment and thus the convergence is slower than when no error plant is present. It has been shown [20] that a *long* error plant causes larger misadjustment within the LMS approach and the effects seen here are to prevent convergence for the case where the error plant consisted of more than 57 sample delays.

Further increase in the duration of the impulse response of the error plant results in the failure of the LMS algorithm to converge the filter.

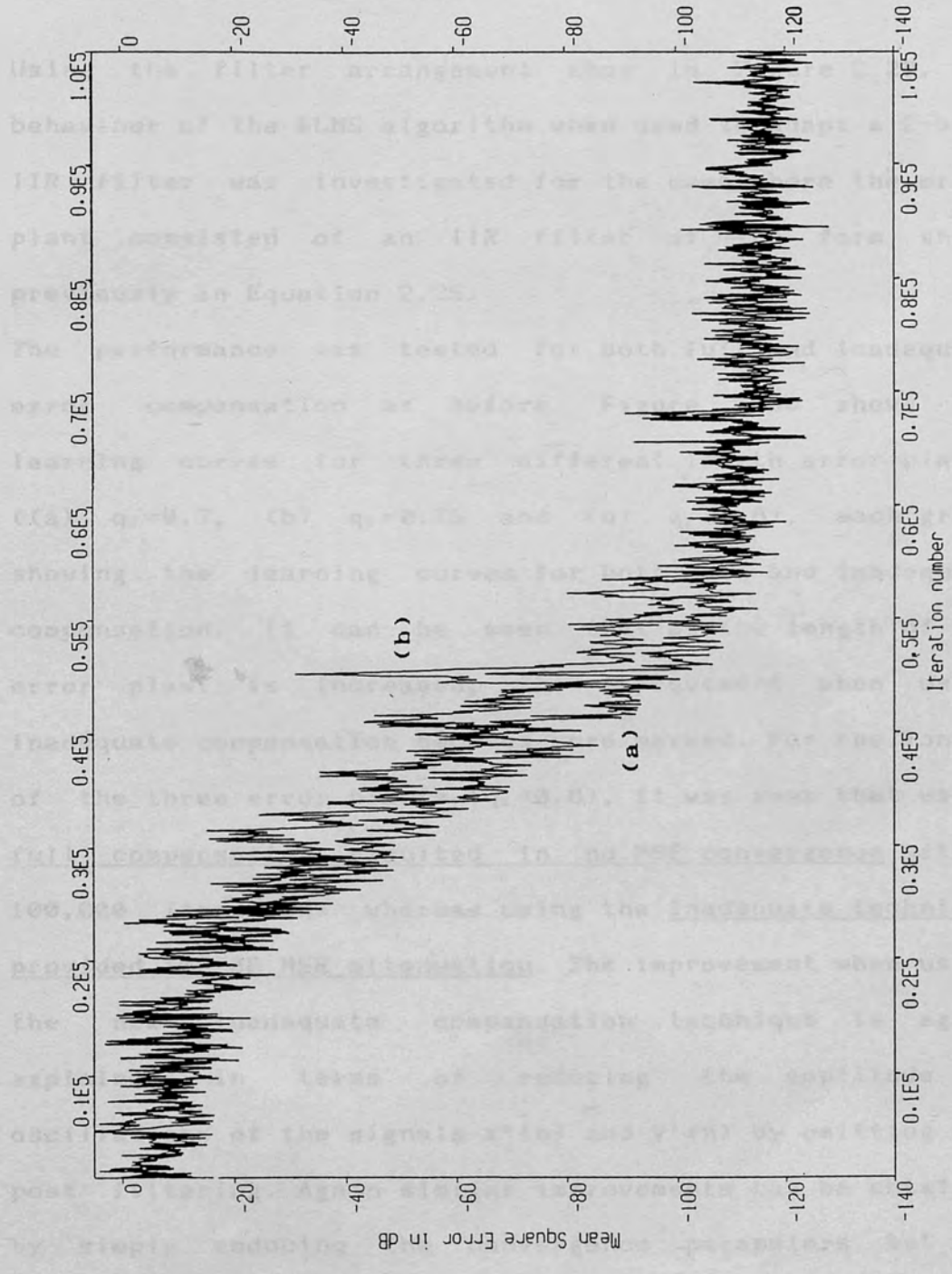


Figure 2.29 - Comparison of convergence of an IIR filter using Feintuch's RLMS algorithm using : (a) a 48 step delay, (b) a 57 step delay

2.10.5 IIR adaptation using direct compensation (IIR error plant)

2.10.5.1 The RLMS algorithm

Using the filter arrangement show in Figure 2.28, the behaviour of the RLMS algorithm when used to adapt a 2-by-3 IIR filter was investigated for the case where the error plant consisted of an IIR filter of the form shown previously in Equation 2.25.

The performance was tested for both full and inadequate error compensation as before. Figure 2.30 shows the learning curves for three different length error plants ((a) $q_2=0.7$, (b) $q_2=0.75$ and (c) $q_2=0.8$), each graph showing the learning curves for both full and inadequate compensation. It can be seen that as the length of the error plant is increased, the improvement when using inadequate compensation becomes more marked. For the longer of the three error plants ($q_2=0.8$), it was seen that using full compensation resulted in no MSE convergence within 100,000 iterations whereas using the inadequate technique provided 100 dB MSE attenuation. The improvement when using the new inadequate compensation technique is again explained in terms of reducing the amplitude of oscillations of the signals x'(n) and y'(n) by omitting the post filtering. Again similar improvements can be obtained by simply reducing the convergence parameters but as explained previously, this is a far less desirable solution than simply removing the recursive parts of the compensation filters H_t and H_u.

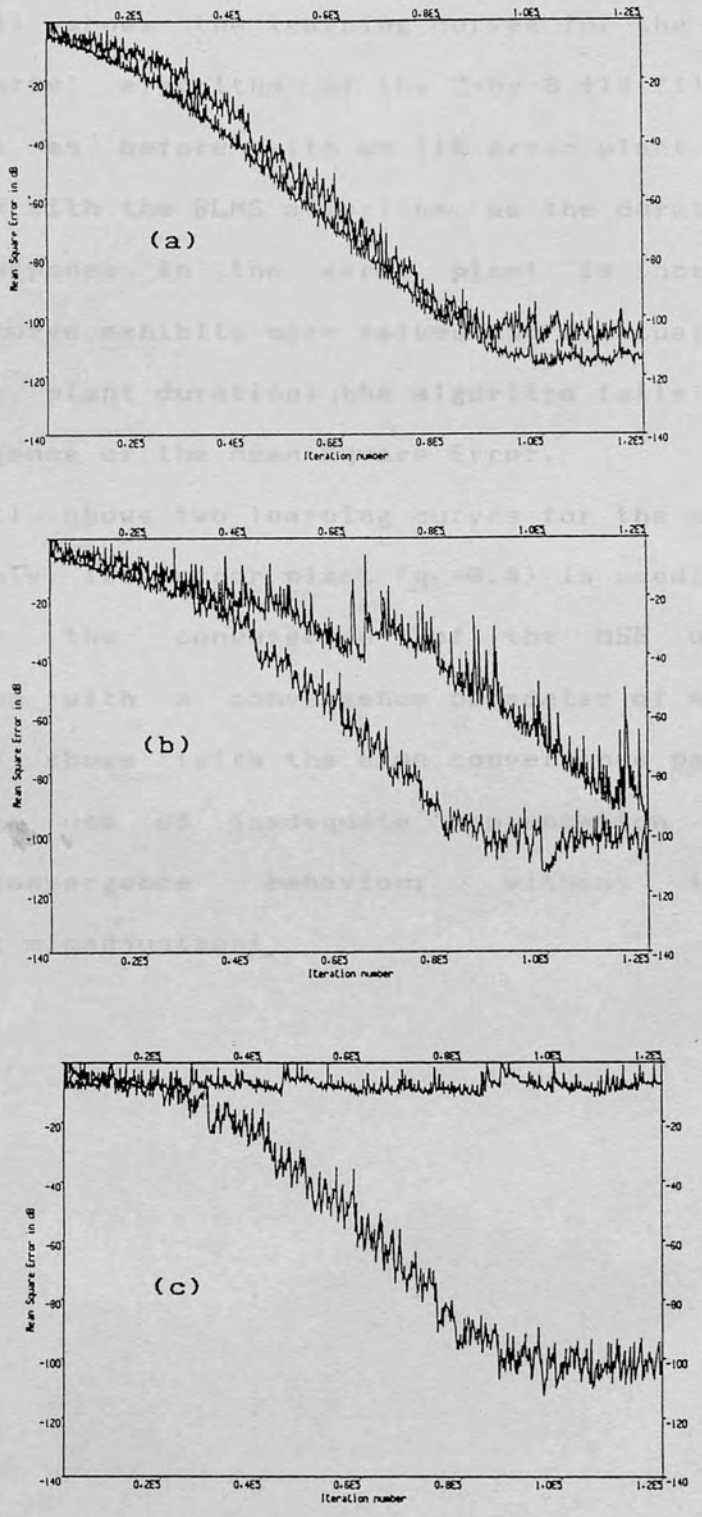


Figure 2.30 - Comparison of convergence of an IIR filter using Feintuch's RLMS algorithm using full and 'inadequate' error compensation with different IIR error plants

2.10.5.2 Stearns' algorithm

Figure{2.31} shows the learning curves for the adaptation using Stearns' algorithm of the 2-by-3 IIR filter to the same plant as before with an IIR error plant of varying length. As with the RLMS algorithm, as the duration of the impulse response in the error plant is increased, the learning curve exhibits more spikes and eventually (with a long error plant duration) the algorithm fails to produce any convergence of the Mean Square Error.

Figure{2.31} shows two learning curves for the cases where a relatively long error plant (q_2=0.4) is used; the first (a) shows the convergence of the MSE using full compensation with a convergence parameter of 4×10^{-3}, the second (b) shows (with the same convergence parameter of 4×10^{-3}) the use of inadequate compensation to restore normal convergence behaviour without introducing significant misadjustment

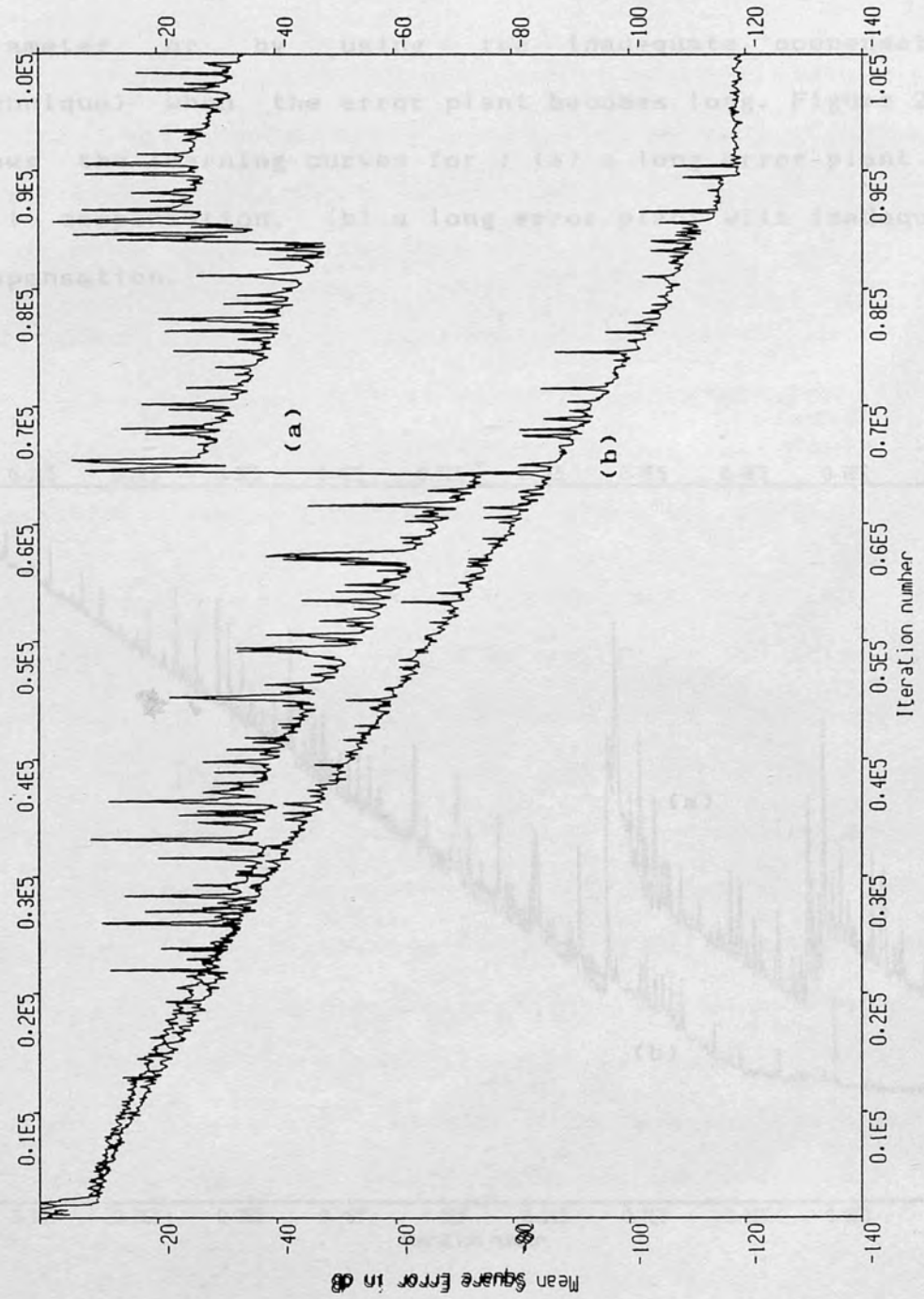


Figure 2.31 - Comparison of convergence of an IIR filter using Stearns' algorithm using (a) full and (b) 'inadequate' error compensation with IIR error plants

2.10.5.3 Fan's AFM algorithm

As with Feintuch's RLMS and Stearns' algorithms, Fan's AFM algorithm performs well for short error plants and requires slower convergence (by either reducing the convergence parameter or by using the inadequate compensation technique) when the error plant becomes long. Figure 2.32 shows the learning curves for : (a) a long error plant and full compensation, (b) a long error plant with inadequate compensation.

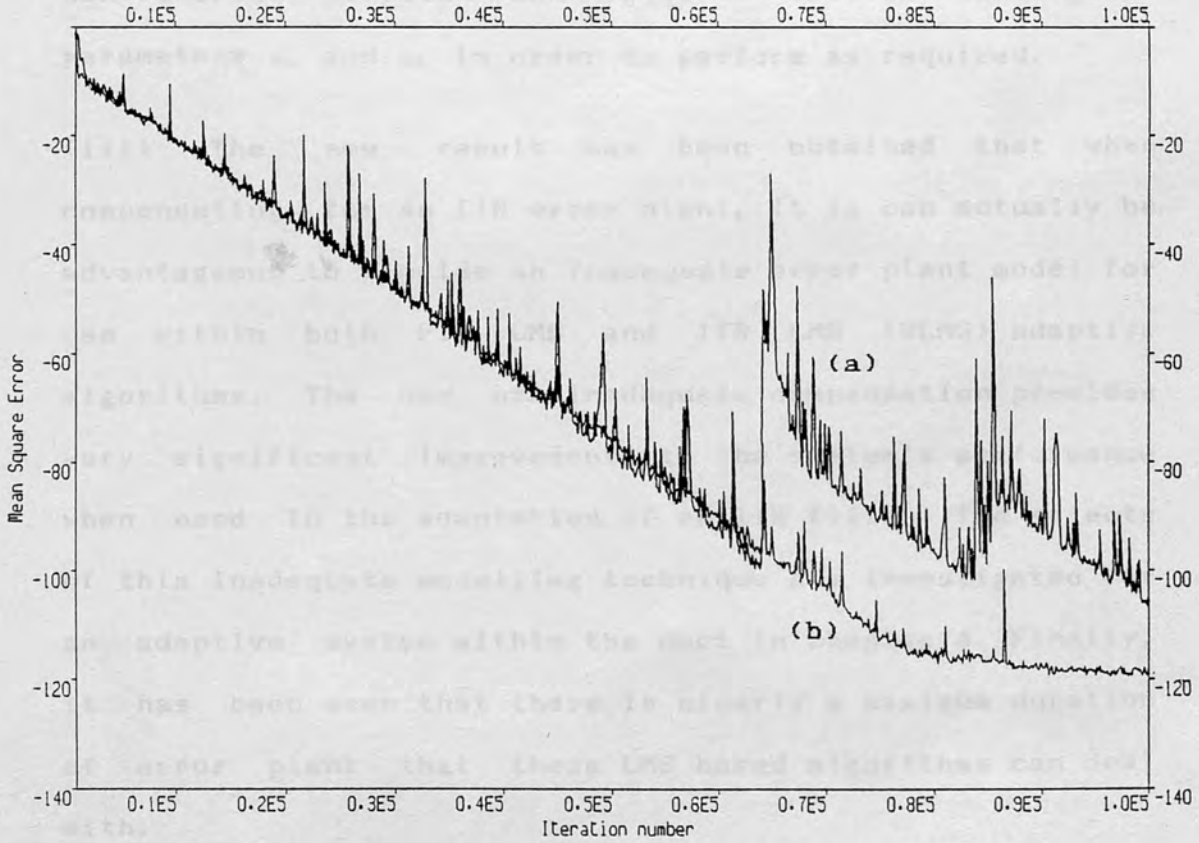


Figure 2.32 - Comparison of convergence of an IIR filter (b) using Fans' AFM algorithm using (a) full and 'inadequate' error compensation with IIR error plants

2.10.6 Conclusions concerning the use of direct error compensation.

The conclusions drawn here are as follows :

(i) In general, the direct error compensation method performs well when using the simple LMS type algorithms, demonstrating increased misadjustment as the length of the actual error plant is increased.

(ii) It would seem that the recursive LMS algorithm is more sensitive to the effects of error compensation than is the non-recursive version and requires smaller convergence parameters μ_a and μ_b in order to perform as required.

(iii) The new result has been obtained that when compensating for an IIR error plant, it is can actually be advantageous to provide an *inadequate* error plant model for use within both FIR LMS and IIR LMS (RLMS) adaptive algorithms. The use of inadequate compensation provides very significant improvement to the system's performance when used in the adaptation of an IIR filter. The effects of this inadequate modelling technique are investigated for an adaptive system within the duct in chapter 4. Finally, it has been seen that there is clearly a maximum duration of error plant that these LMS based algorithms can deal with.

3.1 Introduction

The following chapter is concerned with the various arrangements of adaptive digital controllers currently used in duct ANC systems and their applicability in different situations. A detailed discussion of the use of different filter arrangements is given followed by an up to date summary of the practical systems in existance [8,9,40].

3.2 Forms of the digital controller

The controller in Figure{1.3} (section 1.4.1) may have one of several possible forms :

3.2.1 All zero controller.

The simplest form of the controller would be a single transversal filter. In general, the impulse response of this filter will need to be many times the pure delay between the detector and the secondary source in duration. Figure{3.1} shows the use of a single FIR filter (A) in the duct ANC problem.

The transfer function A is thus given by :

$$A = - \frac{H_{03}}{H_{01}H_{23} - H_{03}H_{21}} \quad \text{.....Eq.(3.1)}$$

The single FIR filter has a finite length impulse response, thus, this method will always be an approximate one as the impulse response required (see equation 3.1) is infinite in duration. It will be seen that, in practice, this single FIR filter form of the controller is only applicable to the

case where there is very significant loss within the duct; i.e. when the effects of the acoustic feedback path H_{21} are small such that the impulse response of the filter A need not be very long. Simulations within a duct have been performed using the FIR LMS approach, which show the effect of truncating the impulse response of the filter A when used in ducts of varying loss.

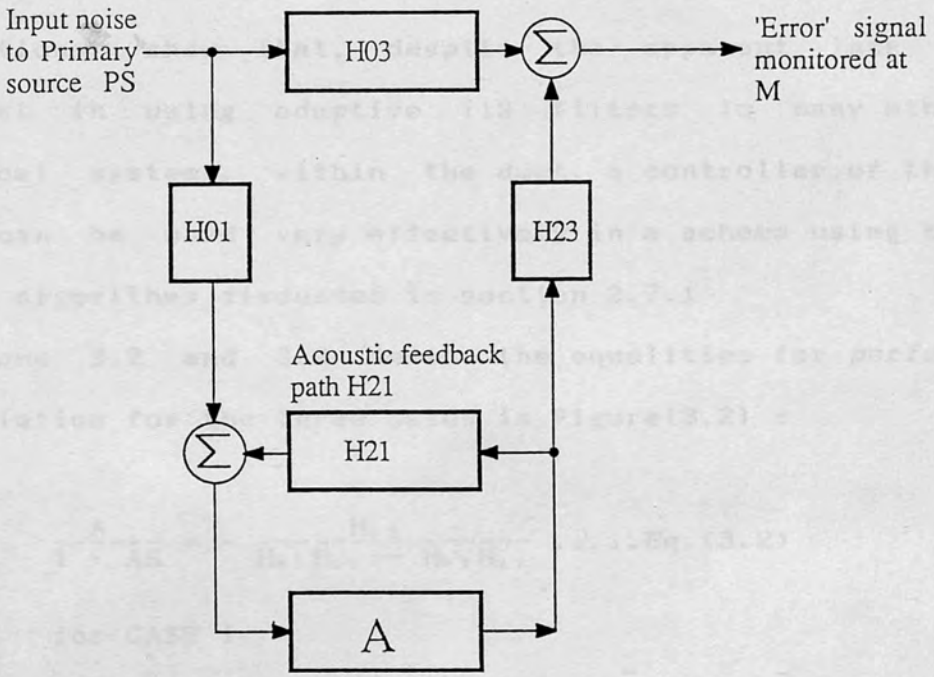


Figure 3.1 - A single FIR controller used as a duct ANC controller

3.2.2 Pole/zero controllers

The pole/zero form of the controller consists of two all zero elements (A and B) so arranged to produce a single pole/zero (IIR) filter. There are three possible arrangements of the FIR elements A and B, these being referred to in this thesis as the Parallel (I), RLMS (II) and Reversed RLMS (III). Filter arrangements for these three forms can be seen in Figure{3.2} (a),(b) and (c) respectively. It is important to compare these three alternative forms when used in both a fixed filtering and an adaptive environment.

This IIR form of the controller is capable of modelling an infinite duration impulse response and as such is applicable to all cases including those where the propagation loss in the duct is negligible. Computer simulations show that, despite the apparent lack of interest in using adaptive IIR filters in many other practical systems, within the duct, a controller of this form can be used very effectively in a scheme using the simple algorithms discussed in section 2.7.1

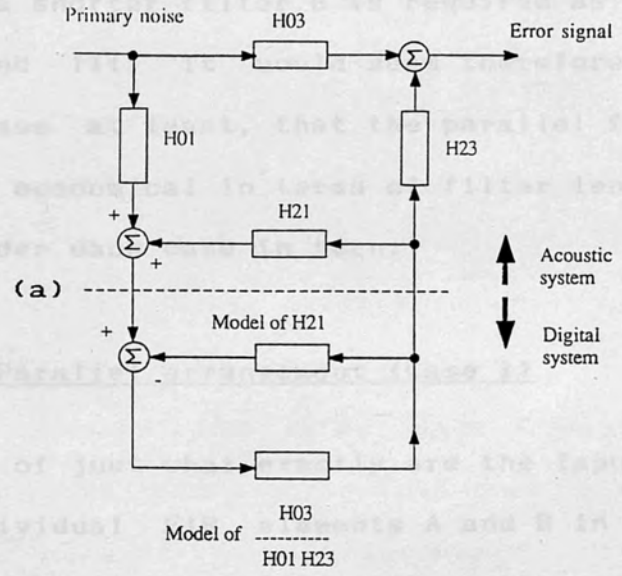
Equations 3.2 and 3.3 define the equalities for *perfect* cancellation for the three cases in Figure{3.2} :

$$\frac{A}{1 - AB} = - \frac{H_{03}}{H_{01}H_{23} - H_{03}H_{21}} \quad \text{.....Eq.(3.2)}$$

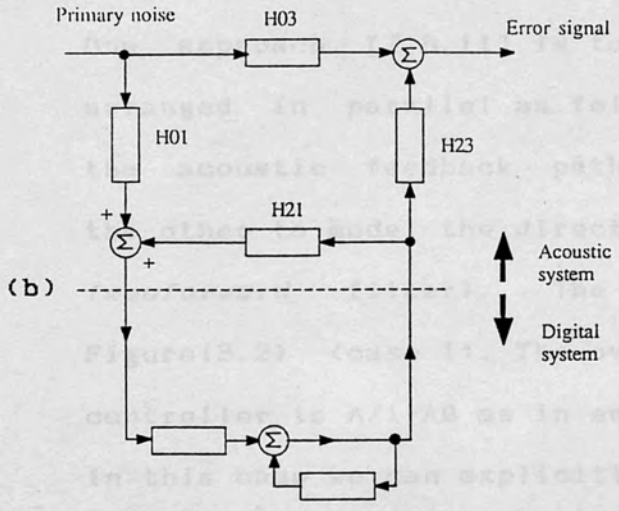
for CASE I

$$\frac{A}{1 - B} = - \frac{H_{03}}{H_{03}H_{23} - H_{03}H_{21}} \quad \text{.....Eq.(3.3)}$$

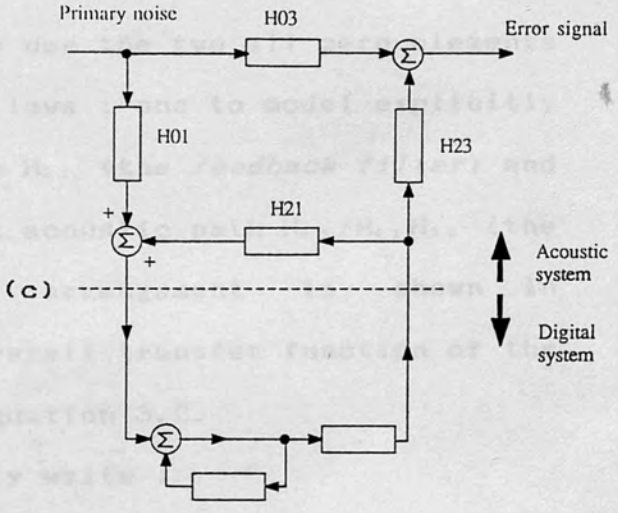
for CASES II & III



CASE I



CASE II



CASE III

Figure 3.2 - The three forms of pole/zero controller for a duct ANC system.

Any of the three arrangements could equally well be used in the fixed filter controller provided of course that the filters A and B have the required responses. It is also observed that since in CASE I, the denominator contains the product AB, a shorter filter B is required as compared with CASES II and III; it would seem therefore in the fixed filtering case at least, that the parallel forms (CASE I) is the most economical in terms of filter length required. Let us consider each case in turn:

3.2.2.1 The Parallel arrangement (Case I)

The question of just what exactly are the impulse responses of the individual FIR elements A and B in the different forms requires a more detailed look at the various possible filter arrangments :

One approach [7,8,11] is to use the two all zero elements arranged in parallel as follows ; one to model explicitly the acoustic feedback path H_{21} (the *feedback filter*) and the other to model the direct acoustic path $H_{03}/H_{01}H_{23}$ (the *feedforward* filter). The arrangement is shown in Figure{3.2} (case I). The overall transfer function of the controller is A/1-AB as in equation 3.2.
In this case we can explicitly write :

$$A = -\frac{H_{03}}{H_{01}H_{23}}$$

and .....Eq.(3.4)

$$B = H_{21}$$

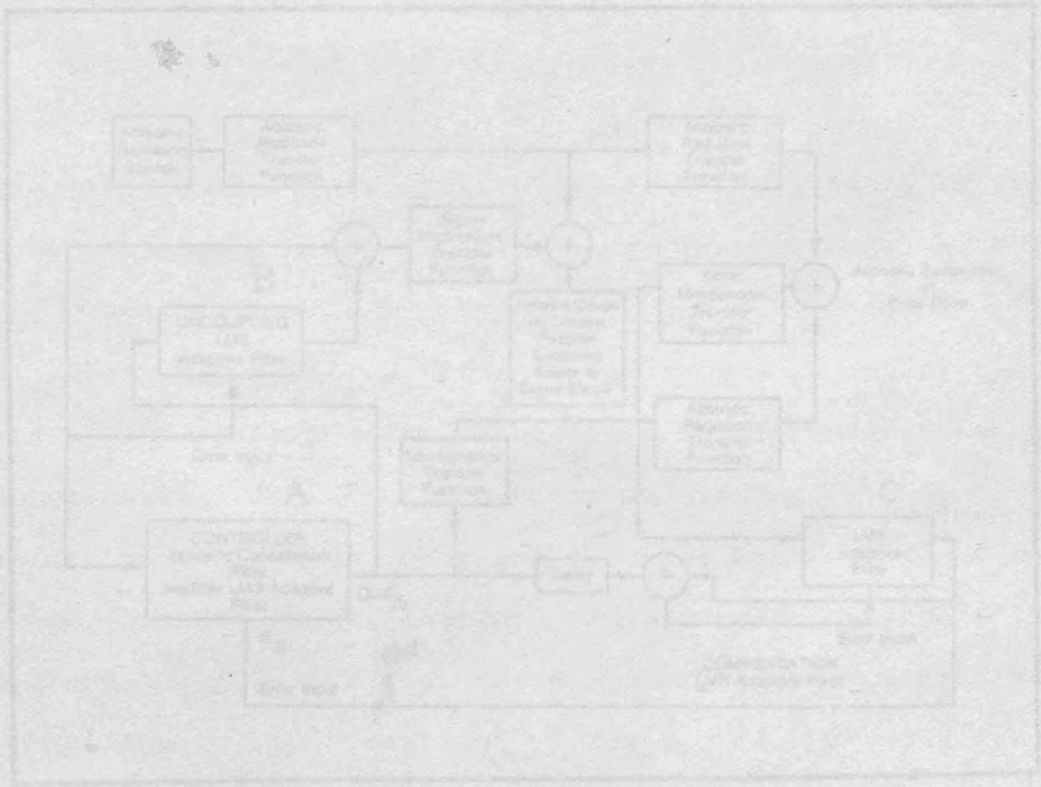
This form is probably the simplest to understand since it is easy to identify the individual modelling requirements of both the FIR elements A and B from equations 3.4. However, in the adaptive environment, this scheme has a possible serious disadvantage in that both elements A and B cannot be adapted at the same time [8,9]. It is understood that in reality, once the elements A and B have been initialised using off-line system identifiction techniques such as described in detail by Gurrie [7], it may suffice to make only the direct element (element A) adaptive and that doing so will automatically counteract any effects of small incorrectness in the feedback element B. Since the system cannot be made truly adaptive, it will always be necessary to perform the off-line system identification in order to initialise the filters A and B. A detailed description of a method for this off-line initialisation is given by Gurrie in his PhD thesis on the subject.

3.2.2.2 The RLMS and reversed RLMS schemes (cases II & III)

Another approach is to use the familiar IIR filter arrangement shown in Figure{2.2}. It has been shown [41] that, of the three possible pole/zero configurations, the RLMS (Case II) filter arrangement is best suited to an adaptive IIR scheme since it is an *observable* form. In short, the filter arrangement is said to be observable if any instability within the arrangement is evident at the final output of the system and can therefore be corrected for by an adaptive process.

The reversed RLMS approach is not an observable form [41]

and is therefore given no further consideration as a possibility. Computer simulations in the duct demonstrate the effectiveness of an adaptive controller based on the RLMS configuration using only the simple IIR adaptive algorithms discussed previously.



3.3 Practical adaptive systems to date

3.3.1 Poole and Warnaka's scheme

In 1984, Poole and Warnaka [8] published their adaptive system involving the use of three independant LMS filters. Their system was based on the parallel (Case I) scheme described in section 3.2.2.1, the third filter being used to model the inverse of the error path as described in section 2.9.2. Figure{3.3} below (taken from their paper [8]) shows the adaptive scheme.

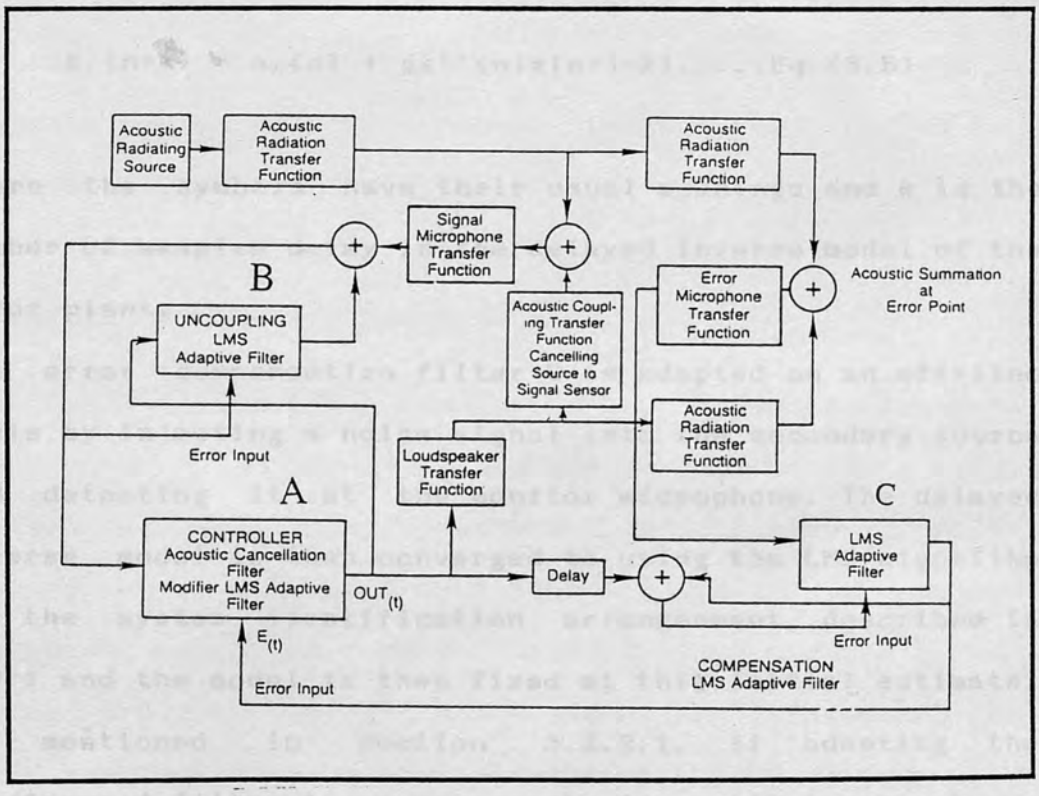


Figure 3.3 - Poole & Warnaka's scheme for duct ANC

In their publication [8] Poole and Warnaka describe the system as *'using a modified Widrow-Hoff LMS algorithm'* to adapt the digital filters therein. In reality the modification made is simply that of delaying the input to the feedforward filter (A) within the update equation to take account of the unavoidable delay in the compensated error signal as described briefly in section 2.9.1 and shown in Equation 3.5. The system is in fact not fully adaptive as suggested in the publication, but relies on off-line initialisation of the *feedback* filter (B) as described fully by Gurrie [7]. Once initialised, the *feedforward* filter (A) is allowed to adapt continuously using the compensated (delayed) error signal $\varepsilon''(n)$ to update the tap weight vector using equation (3.5) below :

$$a_j(n+1) = a_j(n) + \mu\varepsilon''(n)x(n-j-k) \quad \text{.....Eq.(3.5)}$$

where the symbols have their usual meanings and k is the number of samples delay in the delayed inverse model of the error plant.

The error compensation filter C is adapted on an off-line basis by injecting a noise signal into the secondary source and detecting it at the monitor microphone. The delayed inverse model is then converged to using the LMS algorithm in the system identification arrangenment described in 2.9.1 and the model is then fixed at this initial estimate. As mentioned in section 3.2.2.1, if adapting the feedforward filter A *can* compensate for small incorrectness in feedback filter B then the system may perform as a

truly adaptive one provided the model of the error plant remains an adequate one. The error plant however not only involves the acoustic path between secondary source and monitor microphone but also the responses of both the monitor microphone and the secondary source themselves. As such the error plant will be continually changing whilst the adaptive feedforward filter A is unable to compensate for such changes using only the compensated error signal $\varepsilon''(n)$. Further, in order for the adapting feedforward filter A to compensate for incorrectness in B, it will need to have sufficient duration to be able to model the whole of the transfer function $H_{03}/(H_{01}H_{23} - H_{03}H_{21})$ and thus could equally well be used by itself as in the single all zero controller discussd in 3.2.1.

Poole and Warnaka's adaptive scheme is therfore not a particularly attractive candidate in the search for complete adaptivity and seems to have been shelved as a commercially available unit.

3.3.2 Eriksson's adaptive system.

In April 1987, Eriksson et. al. published a paper describing *'The selection and application of IIR adaptive filters for use in active attenuation'*. Eriksson described an adaptive scheme using the IIR filter arrangement discussed in 3.2.2.2 with the addition that Eriksson's system provided on-line modelling of the error plant thus producing complete overall adaptivity. The on-line error plant modelling is achieved by injecting a low amplitude white noise signal [42] into the secondary source and using

this signal to adapt a compensation filter C as shown in Figure{3.4} (taken from one of Eriksson's originals [42]). The adapted tap weight vector of this filter is then used to post filter the signals x(n) and y(n) prior to being used in the Feintuch RLMS update algorithm described in 2.7.1.3. Published results showing the performance of the system/algorithm have been few and there has been no simulation work demonstrating the appropriateness of the RLMS algorithm to the duct case. Eriksson himself has reported that *'all simulations are on simplified systems'*.

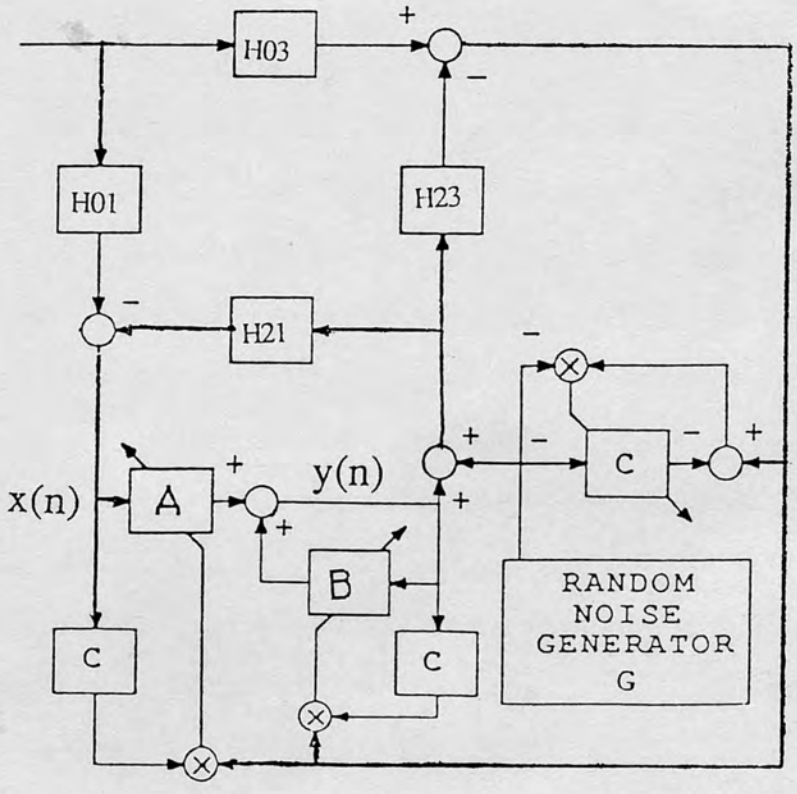


Figure 3.4 - Eriksson's fully adaptive scheme

3.3.3 Billoud and Galland

In April 1988 Billoud and Galland [40] presented experimental work demonstrating the use of IIR filters for active control in a duct. The filters used were adapted to the required responses in an off-line manner using Feintuch's RLMS algorithm and were combined to form a fixed filter system. Analysis was also made of the performance of such a scheme in a water system where the required speed of processing is obviously increased due to the increase in the speed of sound in the medium.

4.1 Simulating the duct.

This chapter is concerned with the performance of adaptive controllers within a simulated ventilation duct. Models of the duct were created by synthesizing the required duct impulse responses and inserting these into digital filters. Using the duct simulation programs, it is possible to specify the characteristics of a duct i.e. its length, transducer spacing, loss and the amount of reverberation and then simulate the action of a digital adaptive controller within that duct.

For simplicity in this chapter, all references to physical lengths and/or transducer spacings within the duct are specified in terms of their equivalent time delay of sound and are given as multiples of the sample period of the digital controller.

Figure (4.1) shows a diagrammatic representation of the system used to simulate the duct characteristics and implement a controller using a chosen algorithm. The component transfer functions H_{01}, H_{21}, H_{03} and H_{23}, were implemented as time domain IIR filters. The recursive part of each IIR filter is equal to the 'round trip' transfer function H_{rt} within the duct and thus all are identical for a given duct. The controller can have the form of either an FIR filter or an IIR filter.

In the simulations reported, the primary random noise was white and was obtained using the same random number generator as described in section 2.8.

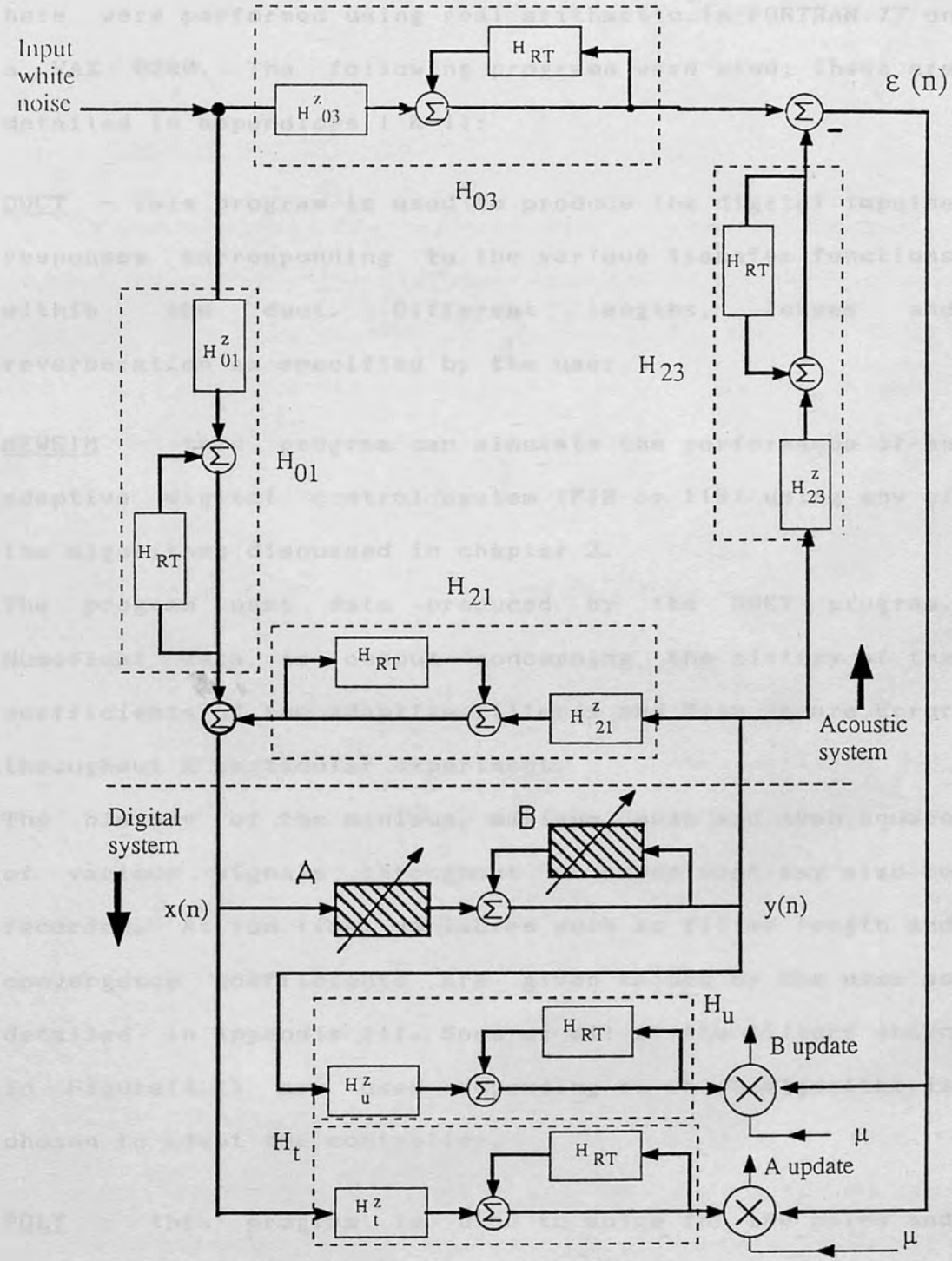


Figure 4.1 - The complete adaptive simulation system used to simulate the duct ANC system

4.1.1 FORTRAN simulation programs

As with previous experiments, the simulations presented here were performed using real arithmetic in FORTRAN 77 on a VAX 8200. The following programs were used; these are detailed in appendices I & II:

DUCT - this program is used to produce the digital impulse responses corresponding to the various transfer functions within the duct. Different lengths, losses and reverberation as specified by the user.

NEWSIM - this program can simulate the performance of an adaptive digital control system (FIR or IIR) using any of the algorithms discussed in chapter 2.
The program uses data produced by the DUCT program. Numerical data is output concerning the history of the coefficients of the adaptive filter/s and Mean Square Error throughout a particular experiment.
The history of the minimum, maximum, mean and mean square of various signals throughout an experiment may also be recorded. At run time, variables such as filter length and convergence coefficients are given values by the user as detailed in Appendix III. Some or all of the filters shown in Figure{4.1} are used depending on which algorithm is chosen to adapt the controller.

POLY - this program is used to solve for the poles and zeroes of the controller at any interval during an experiment, the program uses data produced by NEWSIM and uses NAG routines for solving the polynomials

PLOT - this program plots the learning curve for an experiment and also plots the poles and zeroes of the model on the complex z-plane. The program uses SIMPLEPLOT routines and produces files for the LN03+ laser printer.

4.1.2 Overall transfer function of the duct.

It is understood and has been demonstrated by simulation [10] that the overall transfer function of the reverberant duct is the same as that of the anechoic duct. For a duct exhibiting pure delay of n sample periods between detector and secondary source the required controller transfer function is given by equation 4.1 below :

$$H(z^{-1}) = \frac{\alpha^{n} z^{-n}}{1 - \alpha^{2n} z^{-2n}} \quad \text{....Eq.(4.1)}$$

Where $1-\alpha$ is the loss per sample period within the duct. Expanding Equation 4.1, it it easy to show that for a duct with no loss ($\alpha=1$), the corresponding impulse response does not decay with time. Conversely, if there is finite loss then the impulse response will decay with time and for large loss, the overall impulse response may actually be relatively short and thus may be modelled well by an FIR filter.

Figure {4.2} shows the shapes of the overall impulse response for: (a) a lossless anechoic duct with 20% loss per sample period ($\alpha=0.8$) and (b) for an anechoic duct with 10% loss per sample ($\alpha=0.9$).

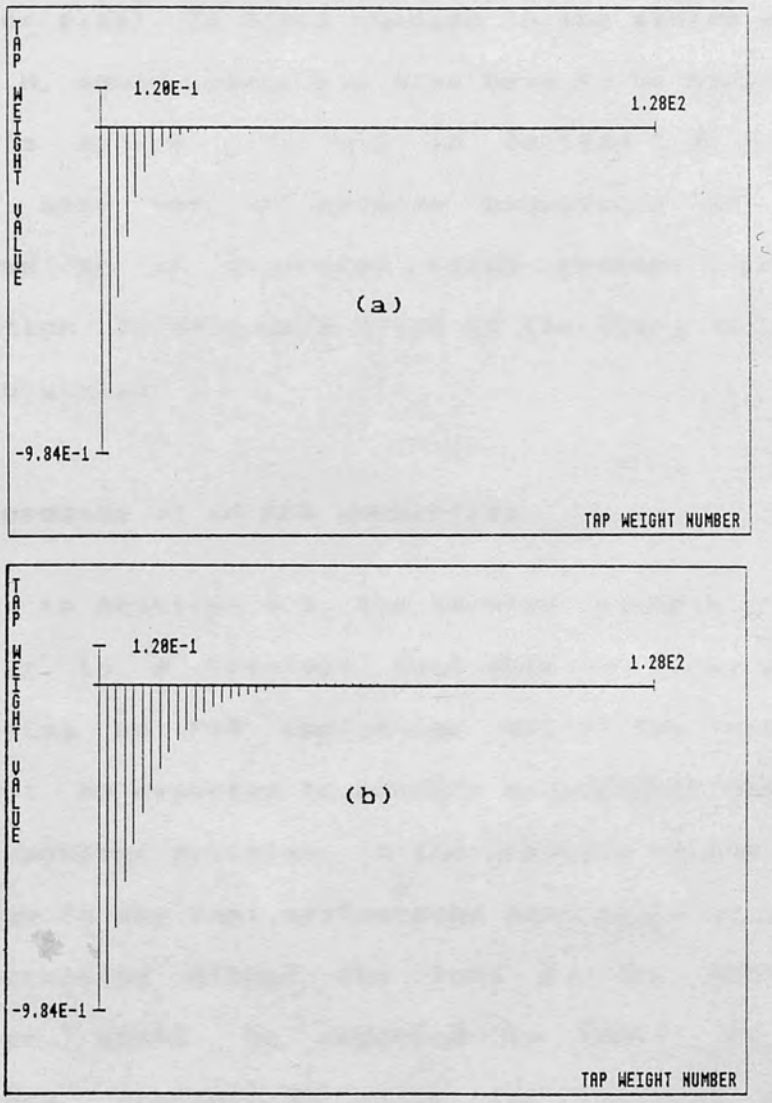


Figure 4.2 - The overall impulse response of (a) an anechoic duct with 20% loss and (b) an anechoic duct with 10% loss per sample period

Error plant

In the duct simulation, the error plant that must be compensated for has the transfer function H_{23} by post filtering the signals x(n) and y(n) as explained in section 2.9.2. The post filtering is achieved using the filters H_t and H_u. In a real system H_t and H_u can only be estimates of

the error plant. To track changes in the system parameters H_t and H_u would themselves also have to be adaptive as in Eriksson's system described in section 3.2. However, the purpose here was to examine behaviours of different algorithms in an otherwise fixed system. Therefore, the compensation filters were fixed at the ideal values unless otherwise stated.

4.2 Performance of an FIR controller

According to equation 4.1, the impulse response of an ideal controller in a lossless duct does not decay with time. Thus, using an FIR controller within the lossless duct would not be expected to produce significant cancellation at the monitor position. In the presence of any loss (and of course in any real system some loss is to be expected), then increasing either the loss or the length of the controller would be expected to result in improved performance. Figure{4.3} shows the learning curves and associated impulse responses for a 128 tap FIR filter when used as a controller in a duct exhibiting different losses: (a) 1%, (b) 10% and (c) 20% loss per sample period.

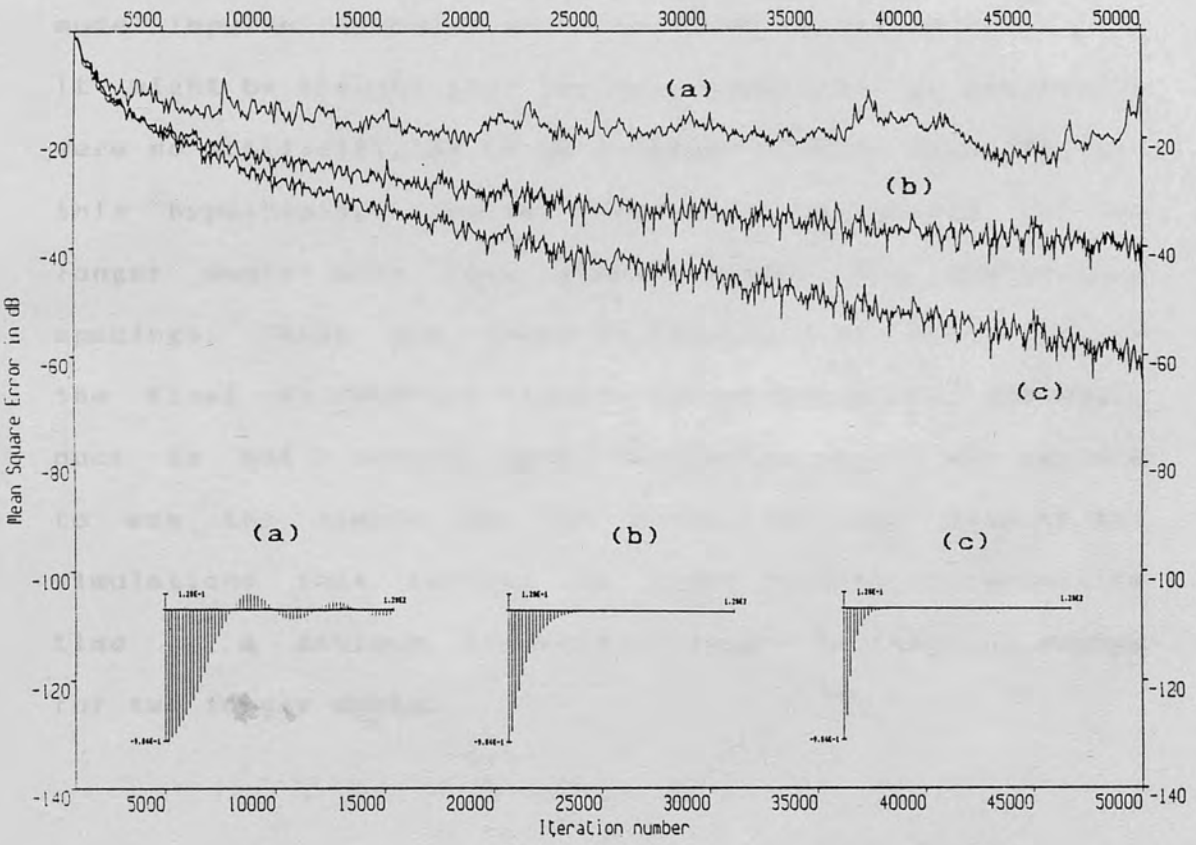


Figure 4.3 - LMS adaptation of a 128 tap FIR filter to an anechoic duct with dimensions of $L_0 = L_1 = L_2 = L_3 = L_4 = 1$ sample period. (a) 1%, (b) 10% and (c) 20% loss per sample period

It is understood that the figures 10% and 20% loss per sample period may not be realistic in practice but are intended only to illustrate the effect of adding more loss to a given duct.

The particular duct used to obtain these results had no

reflection at the ends and thus was anechoic. Results showing very similiar behaviour were obtained when using a duct with significant reverberation (these are not shown so as to avoid unecessary duplication). The final adapted model impulse responses are also shown in Figure 4.3

It might be thought that the very simple set of dimensions were so artificial, as to be a rather special case. To test this hypothesis, results were also calculated for two longer ducts with less simple ratios for the various spacings. These are shown in Figure {4.4}. Similarity of the final attenuation figures demonstrated that the small duct is not a special case. This being so, it was decided to use the simple set of dimensions for many of the simulations that follow, in order to keep the computing time to a minimum. Figure{4.4} shows the learning curves for two longer ducts.

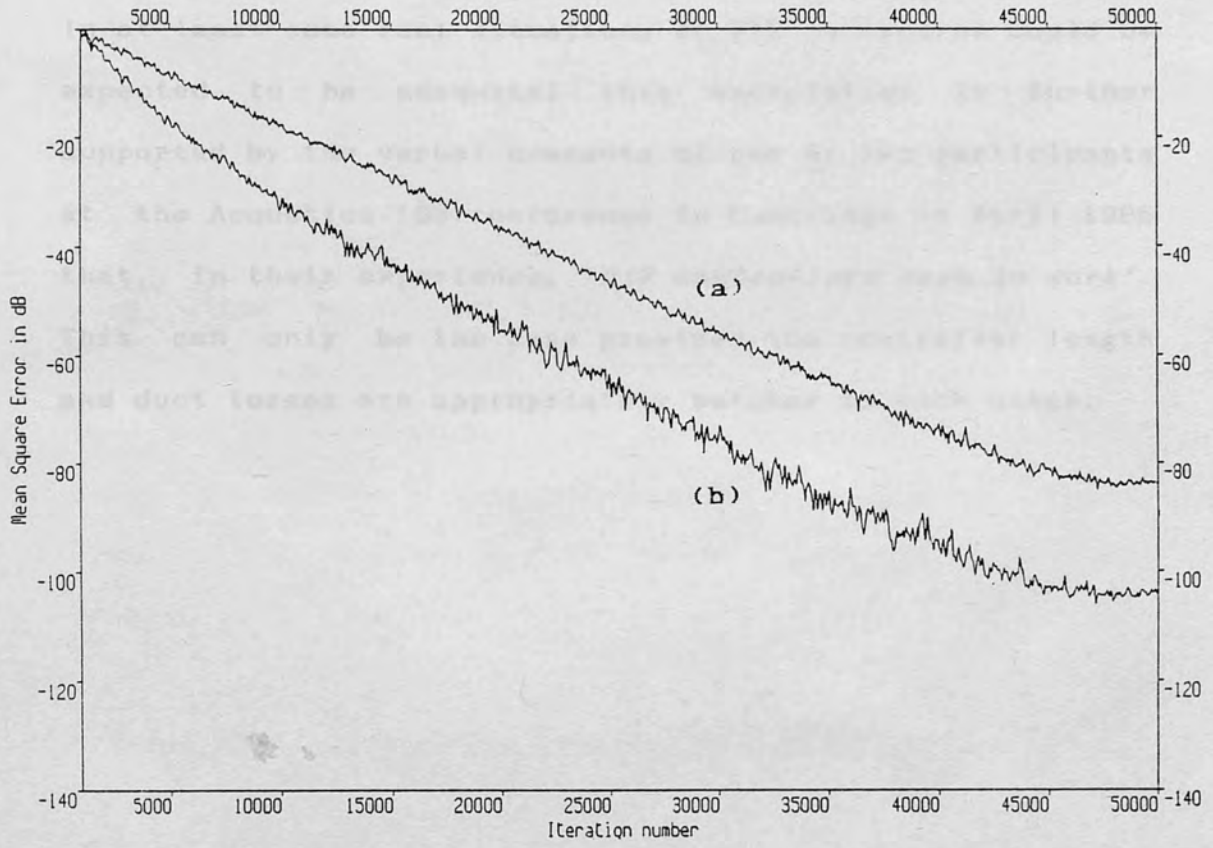


Figure 4.4 - Learning curves for the LMS adaptation of a 128 tap FIR to longer ducts : (a) $L_0=1, L_1=2, L_2=3, L_3=2, L_4=1$ & (b) $L_0=2, L_1=3, L_2=8, L_3=3, L_4=2$ (both with 20% loss per sample period)

Conclusions

The results show, as expected, that an FIR controller can only produce a significant level of sound attenuation if the loss in the duct is such that the magnitude of the impulse response falls to an insignificant level within the duration of the FIR controller.

Private communication with Gurrie revealed that for a particular wooden-walled duct, using a 256 point FIR filter, he could achieve some 30dB attenuation.
In at least some real situations an FIR controller could be expected to be adequate; this expectation is further supported by the verbal comments of one or two participants at the Acoustics '88 conference in Cambridge in April 1988 that, in their experience, *'FIR controllers seem to work'*.
This can only be the case provided the controller length and duct losses are appropriately matched to each other.

4.3 Performance of an IIR filter

N.B In all the following simulations, the desired poles of the controller are purely real [10]. For each experiment, the real parts of the poles are plotted for a given iteration if, and only if, the actual poles of the model are real at that particular iteration.

A discussion of the effects on an adaptive IIR filter of a long error path when using direct error compensation was given in sections 2.10.3 and 2.10.4. It is important that the reader understands the results and the conclusions drawn as these are used to explain the observations made within the duct system when dealing with high reverberation.

4.3.1 A controller using Feintuch's RLMS algorithm

4.3.1.1 Lossless duct with low reverberation

Learning curves (a), (b) and (c) on Figure {4.5} show convergence in the Mean Square Error of the controller's trajectories for the adaptation of an exactly sufficient IIR filter (with 2 direct taps and 3 recursive taps). The duct dimensions were the same as those given in the caption to Figure 4.3. Various different reflection coefficients r_1 and r_2 (corresponding to different amounts of reverberation) were used ranging from 0 (anechoic) to 0.5.

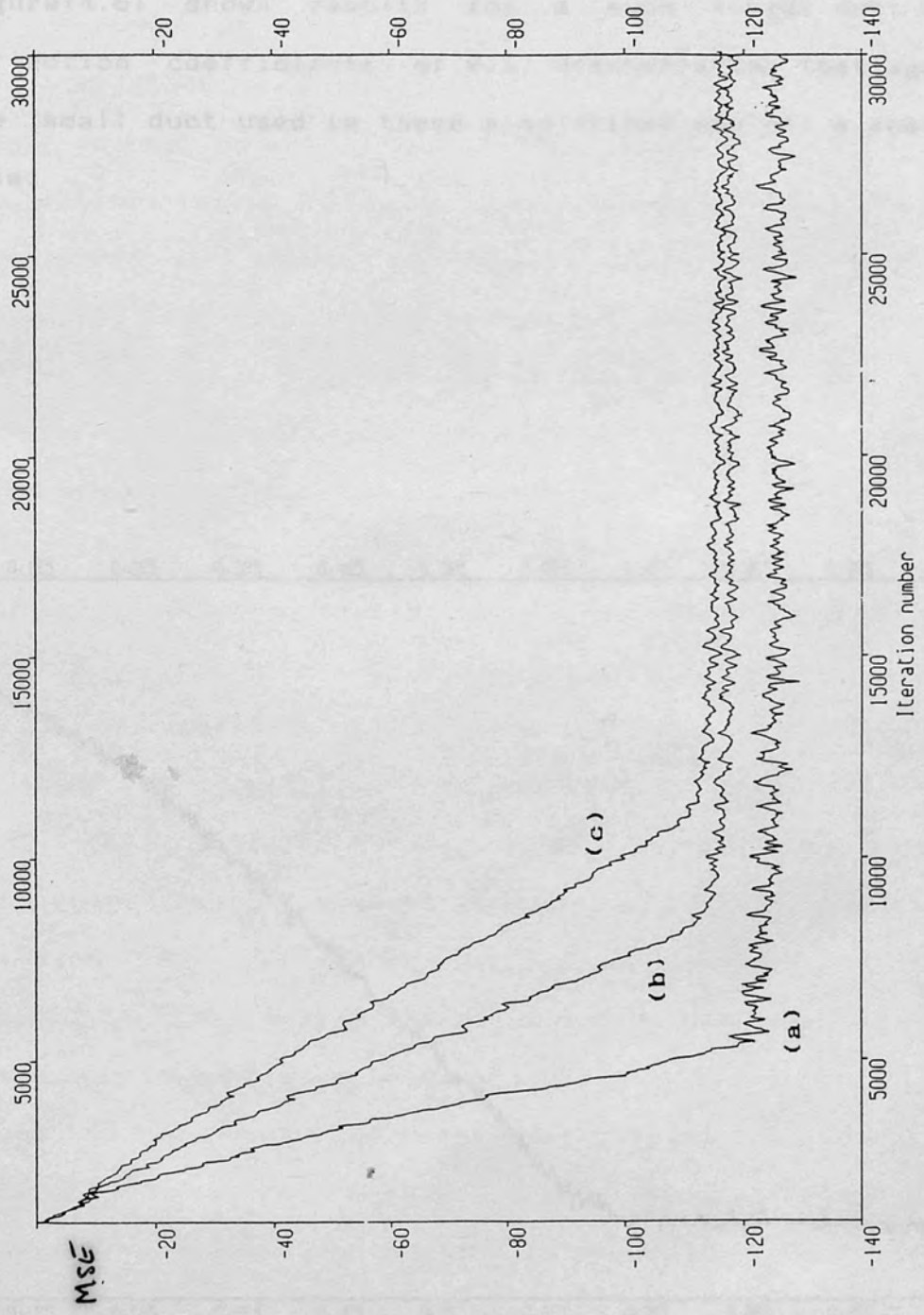


Figure 4.5 - Learning curves for the adaptation of an exactly sufficient IIR controller using Feintuch's RLMS algorithm in a lossles duct : (a) $r_1 = r_2 = 0$, (b) $r_1 = r_2 = 0.2$, (c) $r_1 = r_2 = 0.5$

The poles of the ideal controller are purely real and are located at ± 1+j0 for the lossless duct [10].

Figure(4.6) shows results for a much longer duct with reflection coefficients of 0.5, demonstrating that again, the small duct used in these simulations was not a special case.

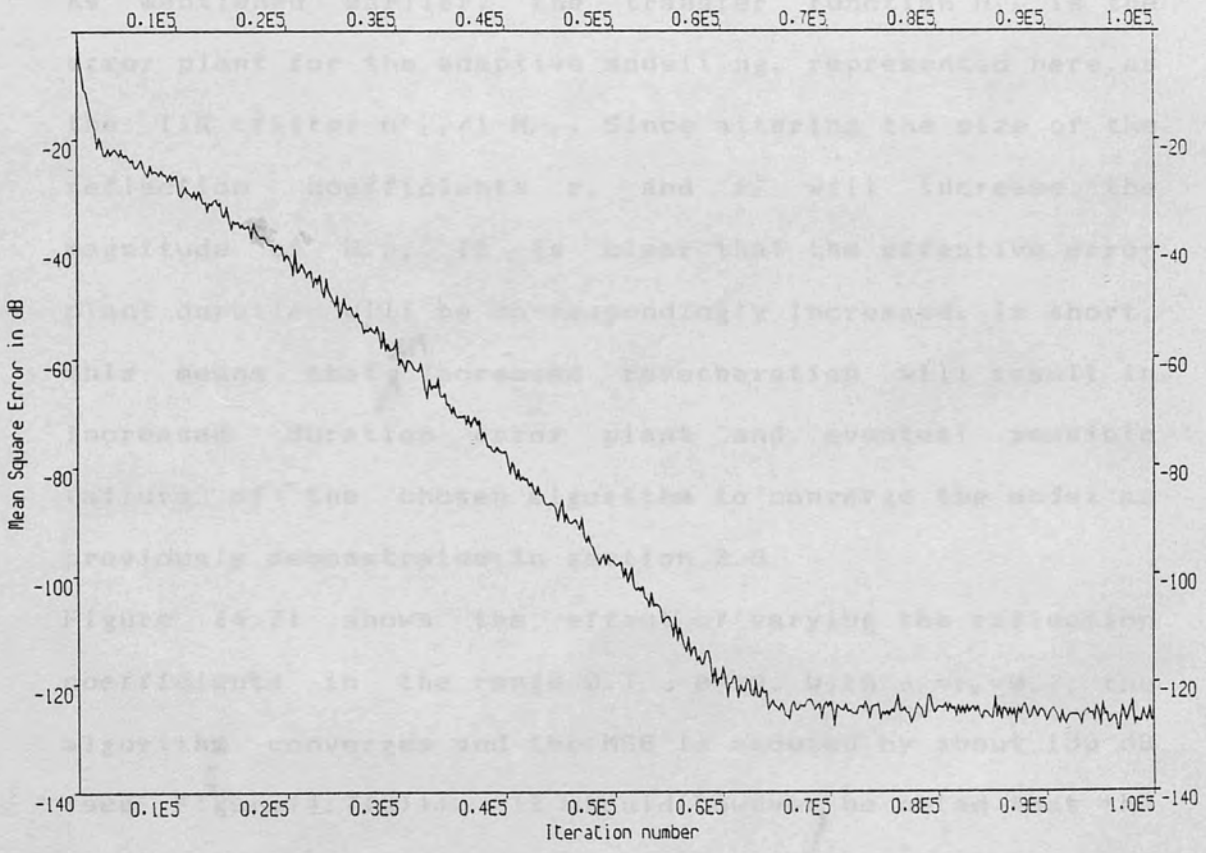


Figure 4.6 - Feintuch's RLMS algorithm used to adapt an exactly sufficient IIR controller to a long duct :

$L_a = 2, L_1 = 3, L_2 = 8, L_3 = 3, L_4 = 2$ sample periods

4.3.1.2 Higher reverberation

The amount of reverberation does not change the characteristics required of the controller as these are specified by Equation 4.1 [10]. However, some important observations were made of the effects on adaptation of increasing the amount of reverberation in the duct (by increasing the size of the reflection coefficients at each end of the duct). This was not unexpected as the amount of reverberation alters the characteristics of the signals available to the adaptive algorithm.

As mentioned earlier, the transfer function H_{23} is the error plant for the adaptive modelling, represented here as the IIR filter $H^{*}_{23}/1-H_{re}$. Since altering the size of the reflection coefficients r_1 and r_2 will increase the magnitude of H_{re}, it is clear that the effective error plant duration will be correspondingly increased. In short, this means that an increased reverberation will result in increased duration error plant and eventual possible failure of the chosen algorithm to converge the model as previously demonstrated in section 2.8.

Figure {4.7} shows the effect of varying the reflection coefficients in the range $0.7 \Rightarrow 0.79$. With $r_1 = r_2 = 0.7$, the algorithm converges and the MSE is reduced by about 130 dB (see Figure{4.7(a)}). It should however be noted that the misadjustment is larger than with the shorter error plant and that the poles slightly overshoot the target values before converging. With a reflection coefficient of 0.785 (see Figure{4.7(b)}), there is a significant delay in

adaptation over the first 8000 iterations and large spikes begin to appear in the learning curve suggesting that reduced convergence parameters are needed.

The poles were seen initially to overshoot further the target values of 1.0. When the reflection coefficients are increased to 0.79 (see Figure{4.7(c)}) in value, the algorithm fails to produce any significant attenuation at all, even though the poles are within about 15% of their desired values. This is not nearly close enough.

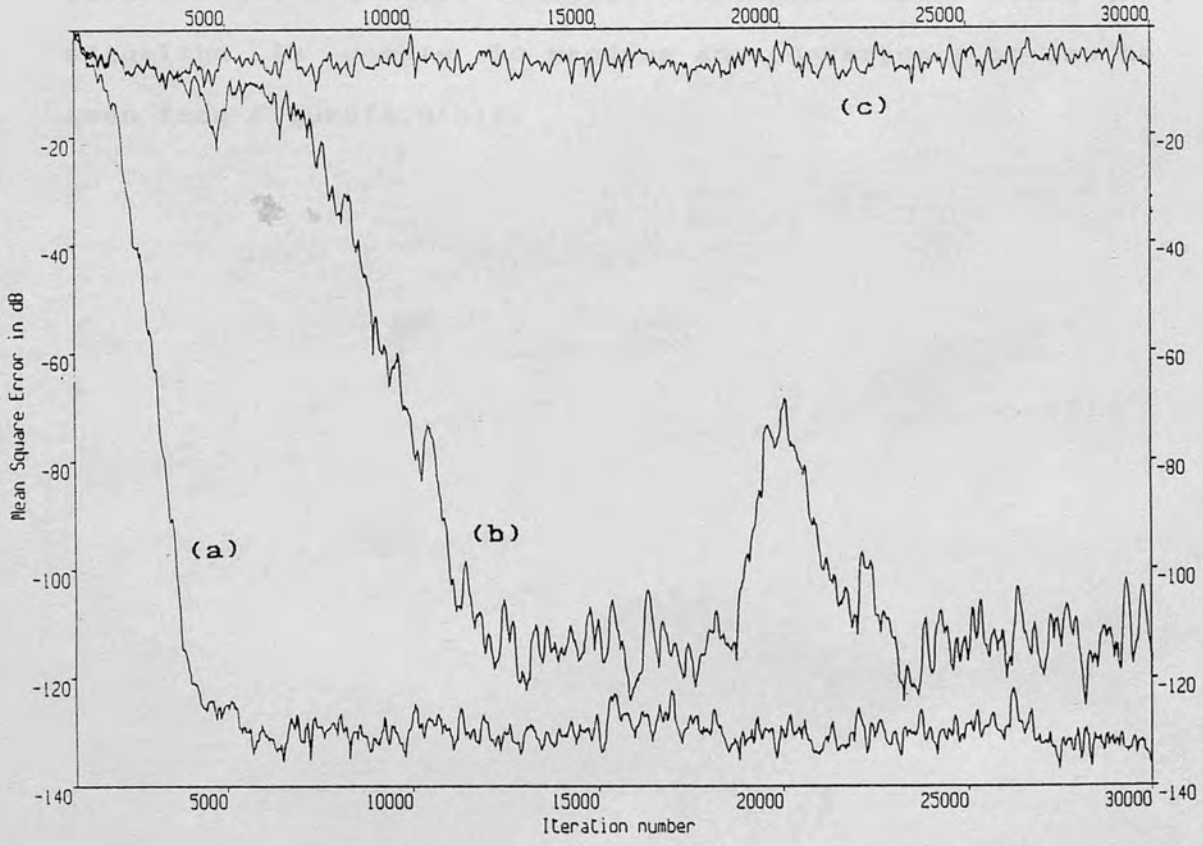


Figure 4.7 - Feintuch's algorithm, the effects on convergence of further increase in the reverberation; (a) $r_1 = r_2 = 0.7$, (b) $r_1 = r_2 = 0.785$, (c) $r_1 = r_2 = 0.79$

4.3.1.3 The effect of lossy propagation

Results obtained after the introduction of 1% loss per sample period when modelling the high reverberation duct (r1=r2=0.79) are shown in Figure {4.8}. With the addition of this loss, normal convergence is restored to about 110dB as can be seen in Figure{4.8(a)}. Further increase in the amount of reverberation again results in the algorithm being unable to control the movements of the poles sufficiently to give any worthwhile attenuation. Again, the delay in adaptation is observed, this is highlighted in Figure {4.8(b)} in which the reflection coefficients are equal to 0.833. For values above 0.833 the algorithm is unable to produce any convergence as can be seen from Figure{4.8(c)}.

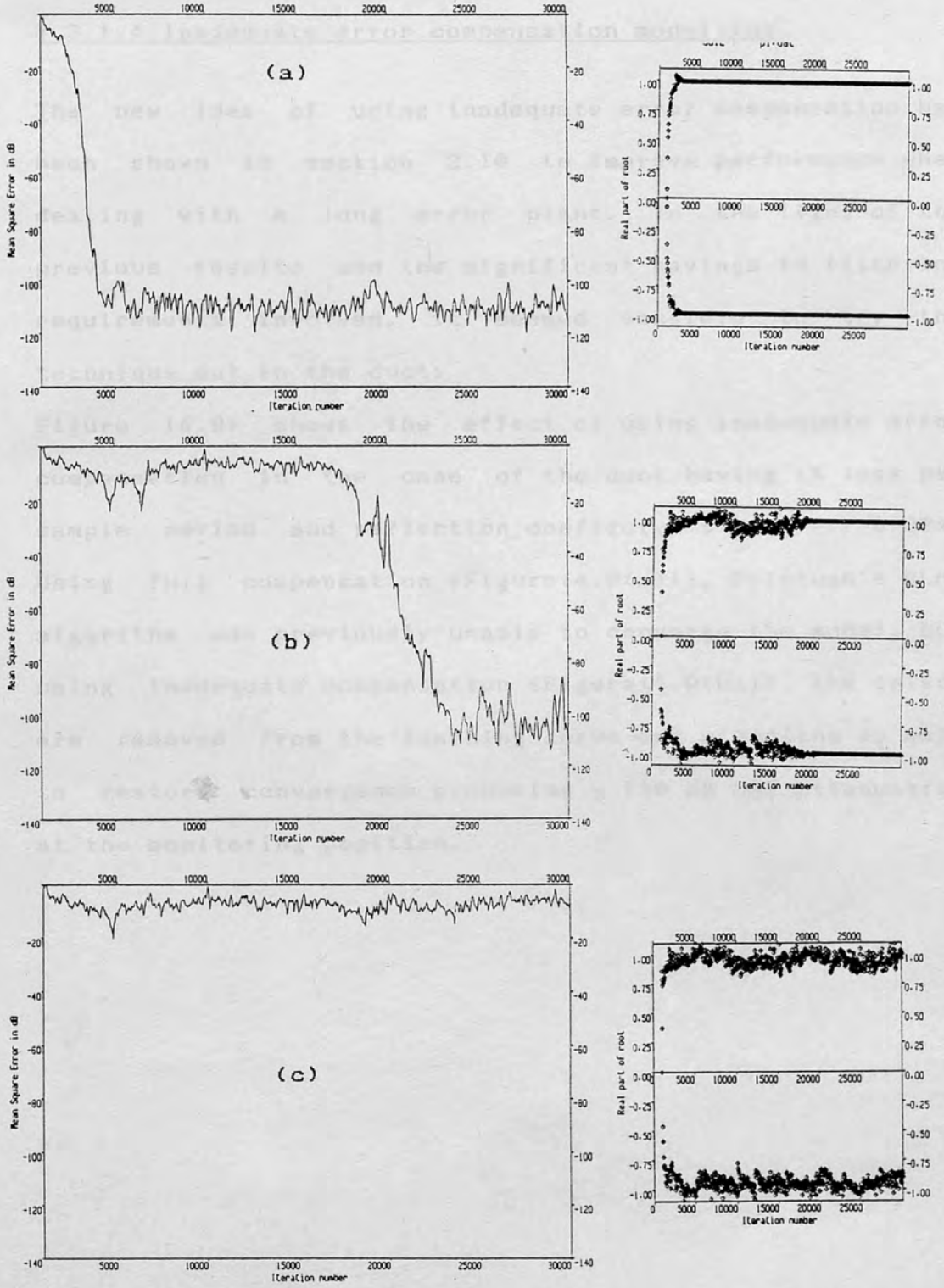


Figure 4.8 - Learning curves and pole trajectories for Adaptation using Feintuch's algorithm to a duct exhibiting 1% loss per sample period and higher reverberations (a) $r_1 = r_2 = 0.79$, (b) $r_1 = r_2 = 0.833$, (c) $r_1 = r_2 = 0.834$

4.3.1.4 Inadequate error compensation modelling

The new idea of using inadequate error compensation has been shown in section 2.10 to improve performance when dealing with a long error plant. In the light of the previous results and the significant savings in filtering requirements involved, it seemed sensible to try the technique out in the duct.

Figure {4.9} shows the effect of using inadequate error compensation in the case of the duct having 1% loss per sample period and reflection coefficients of $r_1 = r_2 = 0.834$. Using full compensation (Figure{4.9(a)}), Feintuch's RLMS algorithm was previously unable to converge the model, but using inadequate compensation (Figure{4.9(b)}), the spikes are removed from the learning curve and algorithm is able to restore convergence producing ≃ 120 dB MSE attenuation at the monitoring position.

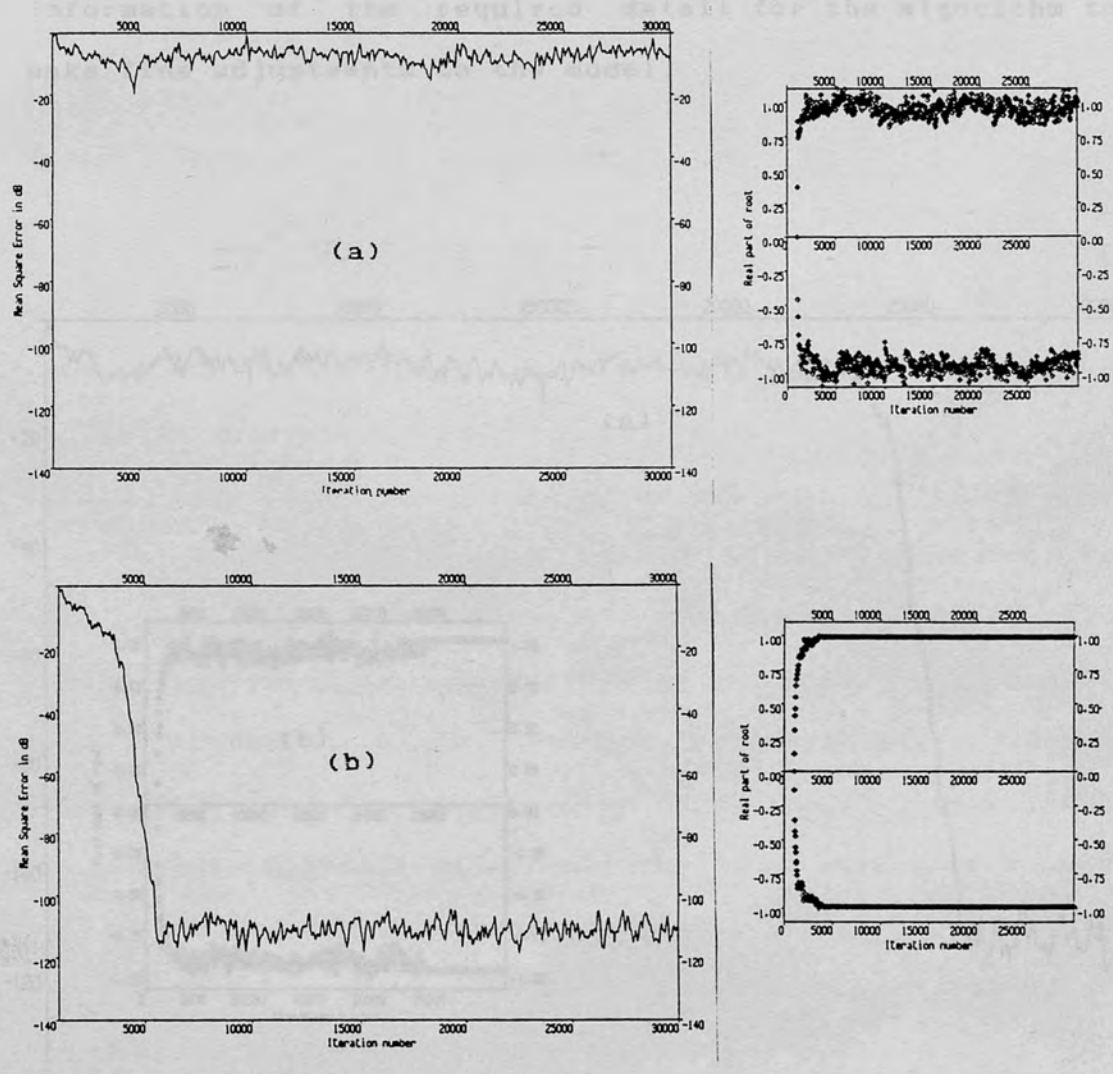


Figure 4.9 - The effect of using inadequate compensation Feintuch's RLMS algorithm in a reverberant duct ($r_1 = r_2 = 0.834$) exhibiting 1% loss; (a) 'full' compensation, (b) inadequate compensation

A further increase in the reverberation (with inadequate compensation) causes the delay in convergence which can be seen in Figure(4.10) as the 'flat' region on the learning beginning to reappear. In this case, the algorithm is able to converge the model but that there is the initial delay in convergence during which there does not seem to be information of the required detail for the algorithm to make fine adjustments to the model.

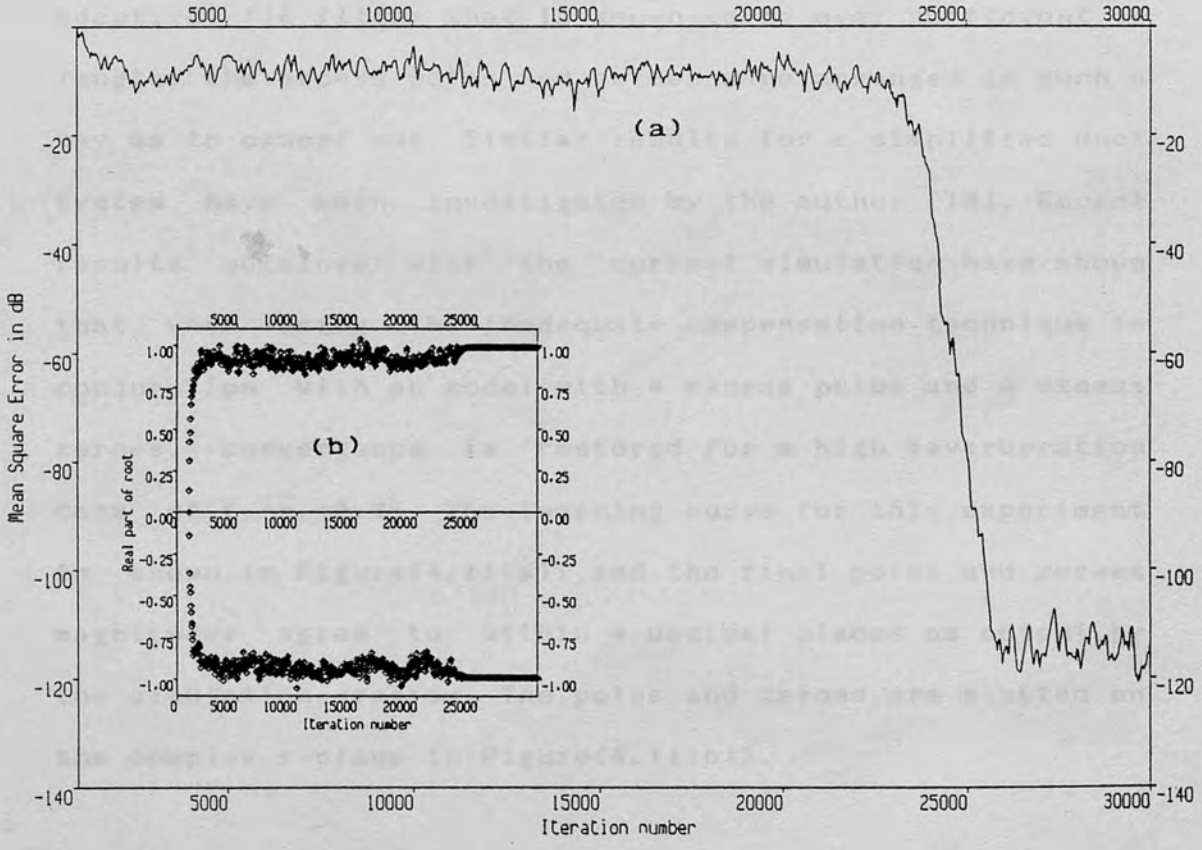


Figure 4.10 - Effects of increasing the reverberation and using inadequate error compensation. $r_1 = r_2 = 0.85$: (a) learning curve, (b) Pole trajectories

With reference to Figure{4.10}, it is clear that something happens at about 24,000 iterations that enables the algorithm to make the required fine adjustments to the model, but as yet the author is unable to explain this action except to make the obvious comment that the required detailed information is delayed more as the duration of the error plant increases.

4.3.1.5 Using an over-sufficient filter

It has been shown in section 2.8 that, when using an adaptive IIR filter that is known to be over sufficient in length, the excess poles and zeroes were arranged in such a way as to cancel out. Similar results for a simplified duct system have been investigated by the author [10]. Recent results obtained with the current simulation have shown that when using the inadequate compensation technique in conjunction with an model with 4 excess poles and 4 excess zeroes, convergence is restored for a high reverberation case of $r_1 = r_2 = 0.85$. The learning curve for this experiment is shown in Figure{4.11(a)} and the final poles and zeroes magnitudes agree to within 4 decimal places as output by the simulation program. The poles and zeroes are plotted on the complex z-plane in Figure{4.11(b)}.

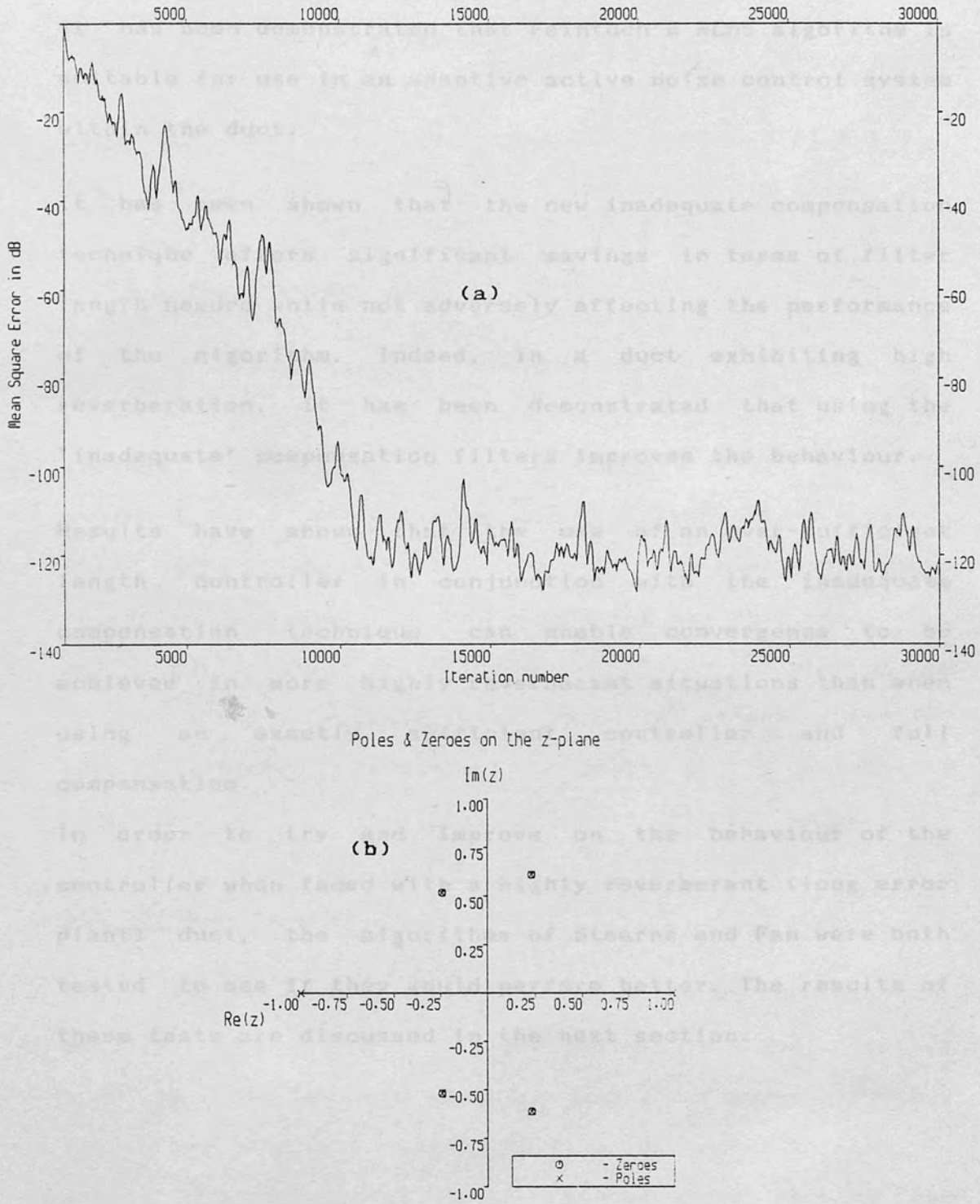


Figure 4.11 - Adapting an over-sufficient IIR filter with 4 extra poles and 4 extra zeroes, using Feintuch's RLMS algorithm and 'inadequate' error plant compensation (a) learning curve (b) Complex z-plane

4.3.1.6 Summary and conclusions

It has been demonstrated that Feintuch's RLMS algorithm is suitable for use in an adaptive active noise control system within the duct.

It has been shown that the new inadequate compensation technique offers significant savings in terms of filter length needed while not adversely affecting the performance of the algorithm. Indeed, in a duct exhibiting high reverberation, it has been demonstrated that using the 'inadequate' compensation filters improves the behaviour.

Results have shown that the use of an over-sufficient length controller in conjunction with the inadequate compensation technique can enable convergence to be achieved in more highly reverberant situations than when using an exactly sufficient controller and full compensation.

In order to try and improve on the behaviour of the controller when faced with a highly reverberant (long error plant) duct, the algorithms of Stearns and Fan were both tested to see if they would perform better. The results of these tests are discussed in the next section.

4.3.2 Using Stearns' algorithm in the duct

Using the same arrangement as in Figure{4.1}, (with the addition of post filtering x'(n) and y'(n)), Stearns' algorithm was simulated in the duct. As in section 4.3.1, the filter used was exactly sufficient in dimensions.

4.3.2.1 The lossless duct

Figure{4.12(a)} shows the learning curve for an experiment identical to that done in section 4.3.1 for the lossless anechoic duct. The result demonstrates, as would be expected, the need for a smaller convergence parameters when using Stearns' algorithm. The smaller convergence parameter is needed since, when using Stearns' algorithm, the signals x'(n) and y'(n) are post filtered through the filter 1/1-B. Figure{4.12(b)} shows the effect of halving the convergence coefficients producing better convergence behaviour.

4.3.2.2 Introducing loss

Figure{4.13} shows the learning curve for an anechoic duct with 1% loss per sample period. This was very much smoother than for the lossless anechoic case above and so it would seem that Stearns' algorithm is affected far more by the signal characteristics in the case of a duct with no loss than is Feintuch's algorithm.

Since it is somewhat unrealistic for a duct to be completely lossless, no further such simulations are reported.

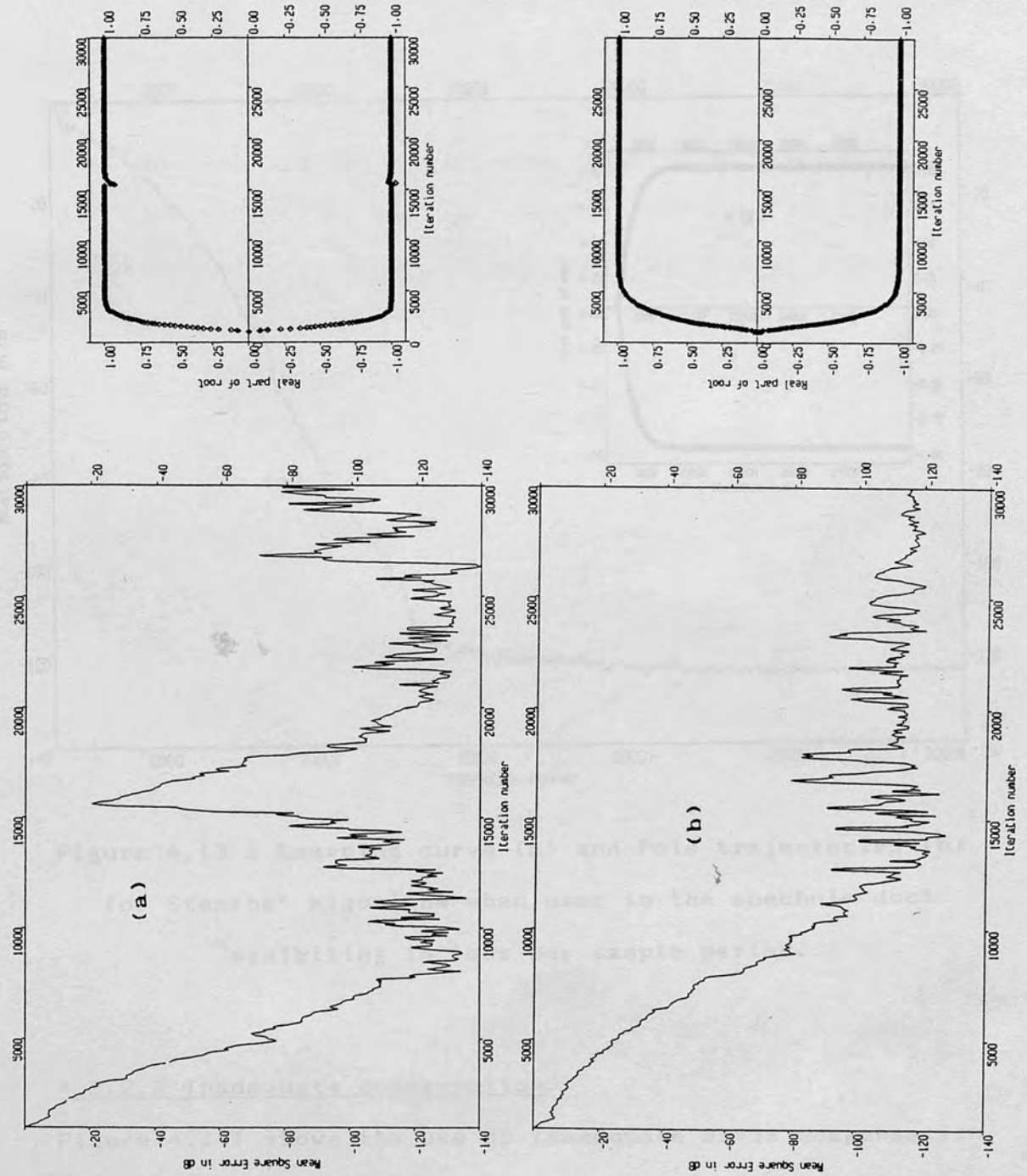


Figure 4.12 - Stearns' algorithm in the anechoic duct; (a) same convergence parameter as RLMS (10^{-3}), (b) convergence parameter reduced to (5×10^{-4})

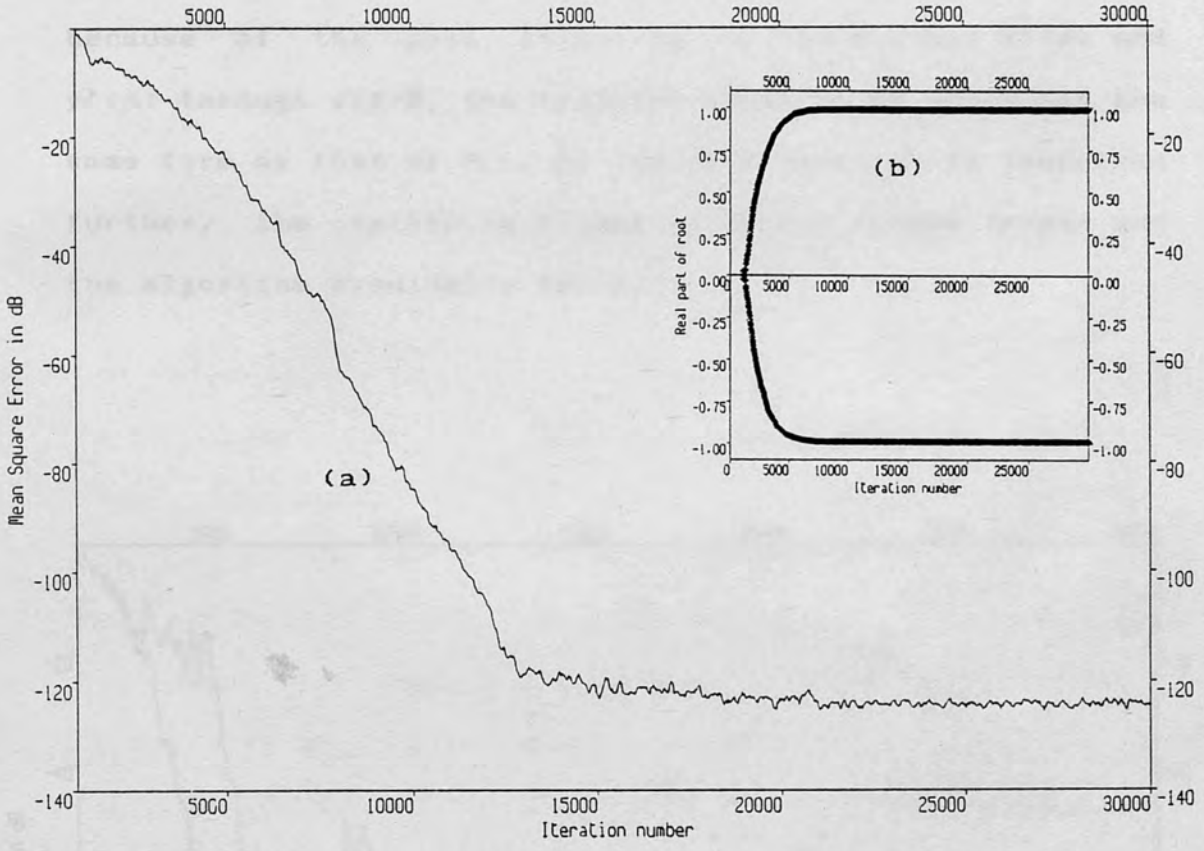


Figure 4.13 - Learning curve (a) and Pole trajectories (b) for Stearns' algorithm when used in the anechoic duct exhibiting 1% loss per sample period.

4.3.2.2 Inadequate compensation

Figure(4.14) shows the use of inadequate error compensation with an increase the amount of reverberation when using Stearns' algorithm. It was seen that when using the inadequate compensation technique, the algorithm demonstrates similar improved behaviour to Feintuch's

algorithm. The 'flat' delayed region on the learning curve is however not present. Again, the introduction of inadequate compensation is successful in restoring convergence at higher reverberations. The effects of this are far less obvious in the case of Stearns' algorithm because of the post filtering of the signals x'(n) and y'(n) through 1/1-B, the transfer function of which has the same form as that of $H_{2,3}$. As the reverberation is increased further, the spikes in Figure {4.14(b)} become larger and the algorithm eventually fails.

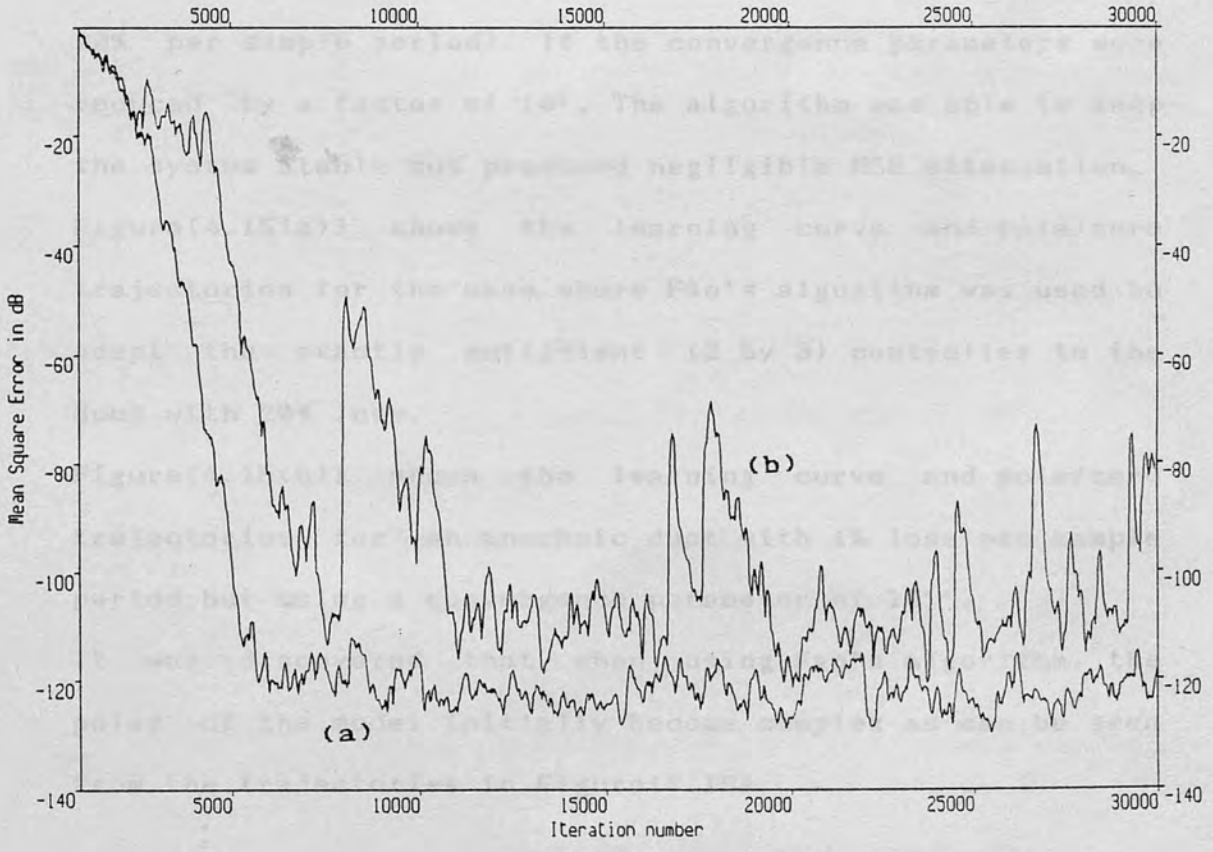


Figure 4.14 - Increasing the reverberation when using Stearns' algorithm; (a) $r_1 = r_2 = 0.6$, (b) $r_1 = r_2 = 0.785$

4.3.2.3 Conclusions

The results in this section suggest that the use of this more complicated algorithm is not justified in this particular duct model since there is no marked improvement over the simpler Feintuch RLMS algorithm detailed in the previous section.

4.3.3 Fan's algorithm used within the duct

The surprising result was obtained, that when using Fan's AFM algorithm, the system was unable to converge and in fact became unstable in all the experiments tried except those for ducts exhibiting very substantial losses (over 20% per sample period). If the convergence parameters were reduced by a factor of 10^2, The algorithm was able to keep the system stable but produced negligible MSE attenuation.
Figure{4.15(a)} shows the learning curve and pole/zero trajectories for the case where Fan's algorithm was used to adapt the exactly sufficient (2 by 3) controller to the duct with 20% loss.
Figure{4.15(b)} shows the learning curve and pole/zero trajectories for an anechoic duct with 1% loss per sample period but using a convergence parameter of 10^{-4}.
It was discovered that when using Fan's algorithm, the poles of the model initially become complex as can be seen from the trajectories in Figure{4.15}.

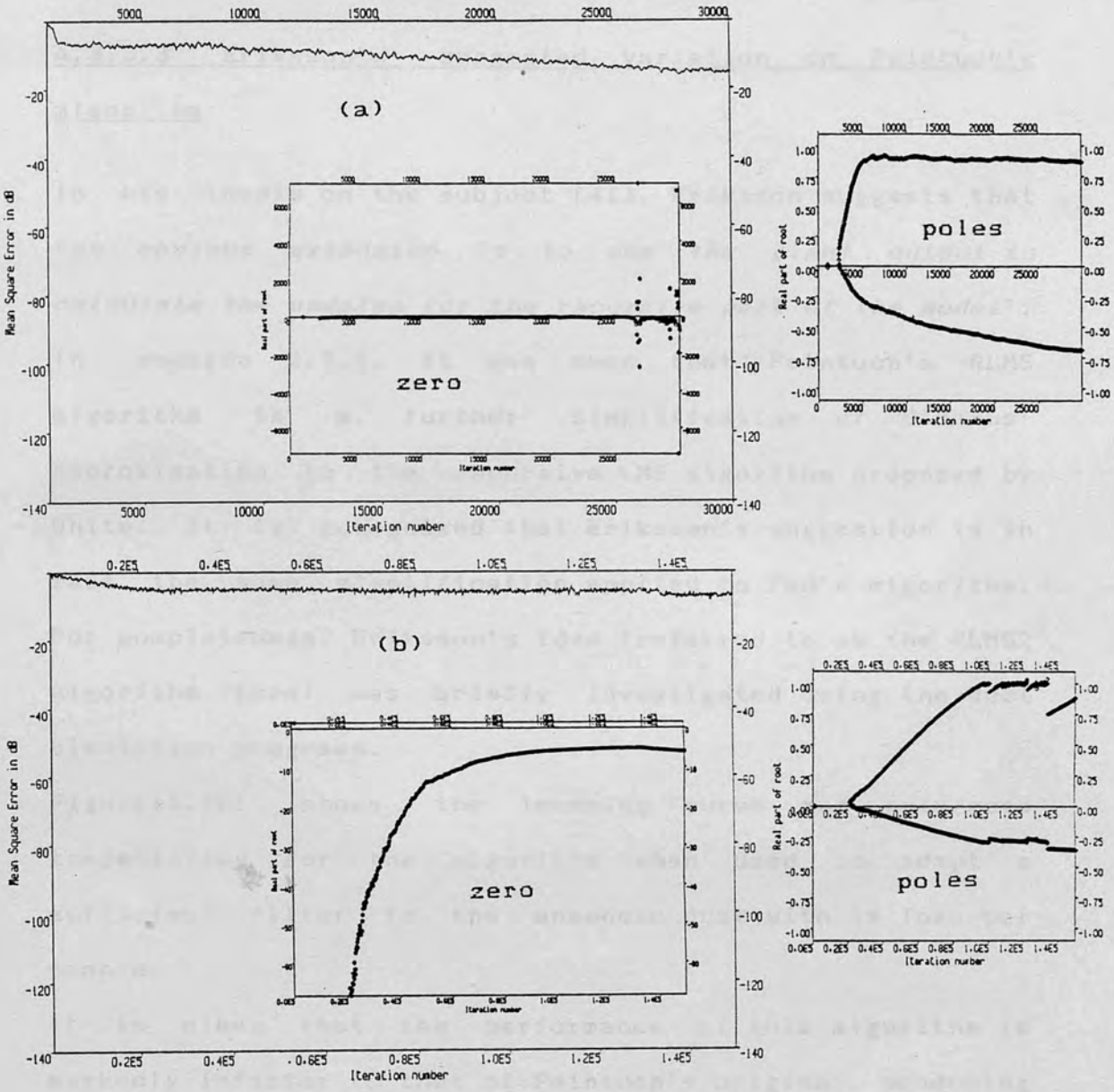


Figure 4.15 - results using Fan's AFM algorithm as a controller within the duct: (a) an anechoic, 20% loss per sample period, (b) 1% loss per sample period and Reduced convergence parameter (μ=0.0001)

4.3.3.1 Conclusions

Far from being an answer to the apparent problems at high reverberation, Fan's algorithm has been shown to be unsuitable for use in an ANC system in the duct.

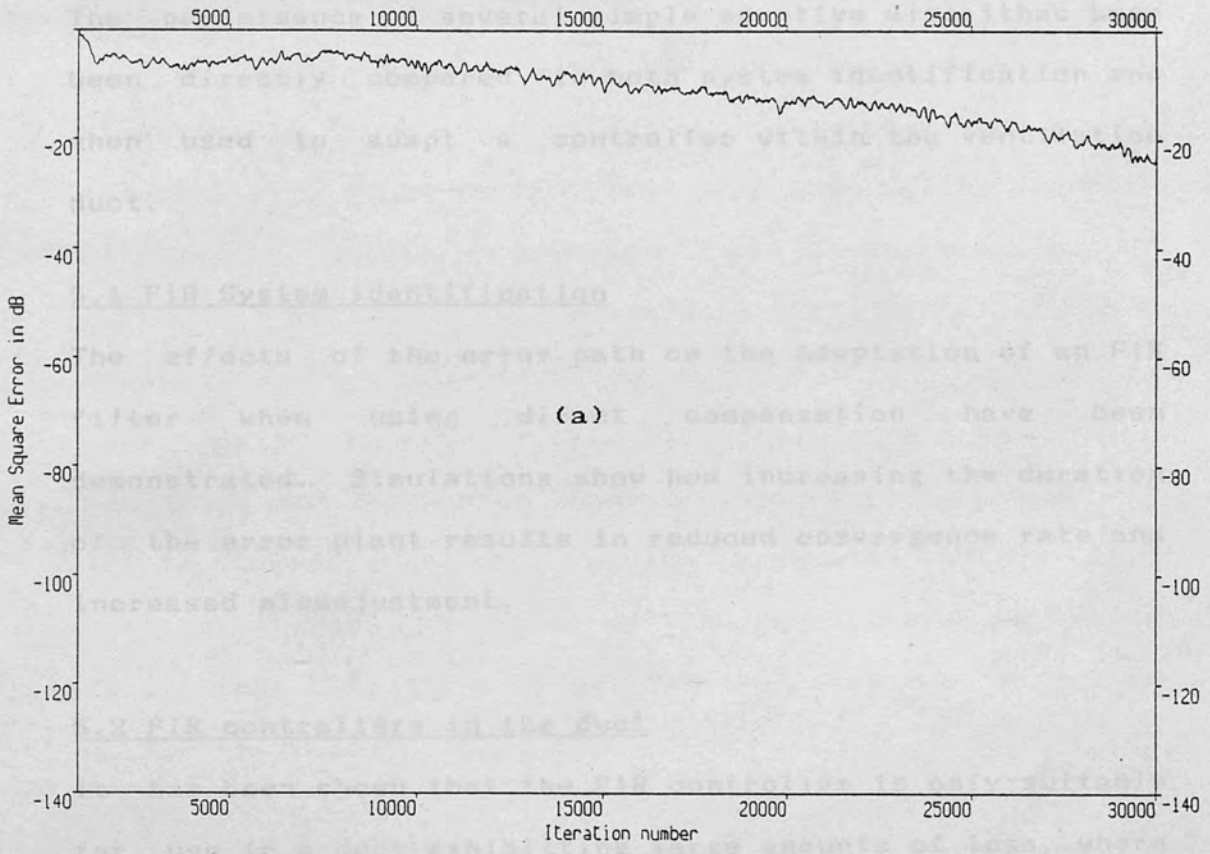
4.3.3.3 Eriksson's suggested variation on Feintuch's algorithm

In his thesis on the subject [41], Eriksson suggests that *'an obvious extension is to use the plant output to calculate the updates for the recursive part of the model'*. In section 2.7.1, it was seen that Feintuch's RLMS algorithm is a further simplification of Stearns' approximation to the recursive LMS algorithm proposed by White. It is recognized that Eriksson's suggestion is in fact the same simplification applied to Fan's algorithm. For completeness, Eriksson's idea (referred to as the RLMS2 algorithm here) was briefly investigated using the duct simulation programs.

Figure{4.16} shows the learning curve and pole/zero trajectories for the algorithm when used to adapt a sufficient filter to the anechoic duct with 1% loss per sample.

It is clear that the performance of this algorithm is markedly inferior to that of Feintuch's original, producing only about 20dB MSE attenuation after the same number of iterations in the identical experiment. It was also seen that one of the poles widely overshoots its target value before converging.

Because of its inferior performance in the simple lossless anechoic duct, the RLMS2 algorithm is given no further consideration in this thesis.



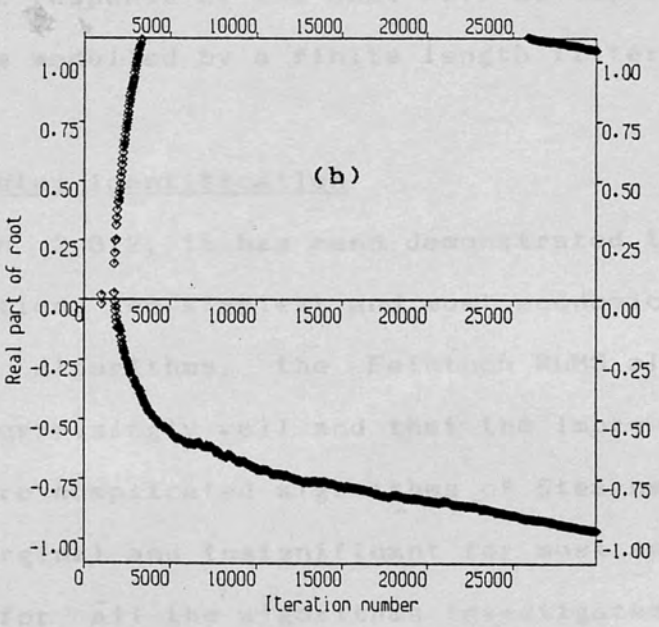


Figure 4.16 - Eriksson's (RLMS2) algorithm used in the anechoic duct with 1% loss per sample period. (a) learning curve (b) Pole trajectories

Chapter 5 - Conclusions

The performance of several simple adaptive algorithms have been directly compared in both system identification and when used to adapt a controller within the ventilation duct.

5.1 FIR System Identification

The effects of the error path on the adaptation of an FIR filter when using direct compensation have been demonstrated. Simulations show how increasing the duration of the error plant results in reduced convergence rate and increased misadjustment.

5.2 FIR controllers in the duct

It has been shown that the FIR controller is only suitable for use in a duct exhibitting large amounts of loss, where the impulse response of the duct will be heavily damped and thus can be modelled by a finite length filter.

5.3 IIR system identification

In section 2.8.2, it has been demonstrated that in system identification, the simplest and most economical of the IIR adaptation algorithms, the Feintuch RLMS algorithm [27], performs surprisingly well and that the improvement offered by the more complicated algorithms of Stearns [30] and Fan [28] is marginal and insignificant for most applications. Further, for all the algorithms investigated it was found that to use 'inadequate' error compensation by modelling only the zero part of an IIR the error plant, gave improved convergence when using direct error compensation.

5.4 IIR controllers in the duct

As the basis for a duct ANC system, the Feintuch RLMS algorithm is again shown to perform remarkably well and the results show that in fact, the more complicated algorithm of Stearns actually provides inferior performance.

Further, it has been shown that Fan's AFM algorithm is unsuitable for use in the duct.

When used in the duct, the inadequate compensation technique has again been shown to improve performance.

It is hoped that the results presented in this thesis provide answers to some of the questions about how the simple adaptive algorithms can be expected to perform within the field of duct ANC.

References

[1] P Lueg, "Process Of Silencing Sound Oscillations" US Patent number 2,043,416, 1936.

[2] J.E. Ffowcs-Williams, "Anti-sound," Proc. R. Soc. Lond. Vol. a 395, pp. 63-88: 1984

[3] G.E.Warnaka, "Active Attenuation - The State Of The Art", Noise Control Engineering, May-June 1982, pp 100-110

[4] M. Jessel and G. Mangiante. "Active sound absorbers in air ducts". Journal Of Sound and Vibration 23, pp. 383-390. 1972.

[5] M.A. Swinbanks. "The active control of sound propagating in ducts", Journl Of Sound and Vibration 27, pp. 411-436, 1973.

[6] Kh. Eghtesadi, W.K.W. Hong and H.G. Leventhall, "The tight coupled monopole active attenuator in a duct", Noise Control Engineering Journal, Vol 20 no. 1, pp. 16-20. Jan. 1983

[7] T.M. Gurrie and S.J. Flockton, "Application of system modelling to the design of digital systems for active control of acoustic noise in a duct," Proc. Inst. of Acoustics Spring Conference, Cambridge, April 1988.

[8] L.A. Poole, G.E. Warnaka and R.C. Cutter, "The implementation of digital filters using a modified Widrow-Hoff algorithm for the adaptive cancellation of acoustic noise," Proc. ICASSP 84, pp. 21.7.1-21.7.4, 1984.

[9] L.J. Eriksson, M.C. Allie and R.A. Greiner, The selection and application of an IIR adaptive filter for use in active sound attenuation," IEEE Trans. Acoust., Speech, Siganl Processing, Vol. ASSP-35, pp. 433-437, April 1987.

[10] S.J. Flockton and N.J. Abbott, "Simulation of the behaviour of the Eriksson adaptive noise control system in a reverberant duct," Proc. Inst. of Acoustics Spring Conference, Cambridge, April 1988.

[11] S.J. Flockton, "Cancellation of acoustice noise in a pipe using digital adaptive filters", Proc. ICASSP 87, pp. 5.8.1-5.8.4, April 1987.

[12] Texas TMS32010/32020 user manual, TEXAS Instruments.

[13] NEC Electronics Europe (1983) 'Workshop-Digital Signal Processor µPD 7720 Dusseldörf

[14] Klocker & Kevin. 'The architecture and applications of the Motorola DSP6000 Digital Signal Processor', Proc ICASSP-87 pp. 523-526

[15] C.F. Ross, "An Adaptive Filter For Broaband Active Sound Control", Journal Of Sound and Vibration 80(3), pp.381-388, 1982

[16] A Roure, Self-Adaptive Broadband Active Sound Control System", Journal Of Sound and Vibration 101(3), pp.429-441, 1985.

[17] La Fontaine and Shepherd, "An Experimental Study Of A Broadband Active Attenuator For Cancellation Of Random Noise In Ducrt", Journal Of Sound and Vibration 91(3), pp.351-362, 1983.

[18] Trinder and Nelson, "Active Control Of Finite Length Ducts", Journal Of Sound and Vibration 89(1), 1983

[19] C. Bean, Active Control Of Acoustic Noise In A Small Enclosure", Proc. Inst. Acoust. Vol. 10, part 2 pp.617-619. 1988.

[20] B. Widrow, "Adaptive Filters", Aspects Of Network and Sysytem Theory, R.E. Kalman and N. De Claris (Eds). New York: Holt, Rinehart and Winston, pp.563-587, 1970

[21] B. Widrow and S.D. Stearns "Adaptive Signal Processing", Prentice- Hall, inc, Englewood Cliffs, N.J. 07632. ISBN 0-13-004029-0.

[22] S. Haykin, "Introduction To Adaptive Filters" Macmillan, ISBN 0-02 -949460-0

[23] C.S. Williams, Designing Digital Filters", Prentice-Hall International Editions, ISBN 0-13-201831-4

[24] Bershad and Feintuch, "A Normalised Frequency Domain LMS Adaptive Algorithm", IEEE Trans. ASSP vol. 34, no. 3, 1986.

[25] D.L. Duttweiler, "A Single Chip VLSI Echo Canceller", Bell Technical Journal vol. 59, no. 2 Feb. 1980.

[26] M.G. Larimore, J.R. Treichler and C.R. Johnson, Jr.,"SHARF: An Algorithm For Adapting IIR Digital Filters," IEEE Trans. ASSP. vol. ASSP-28, p.428, August 1980.

[27] P.L. Feintuch, "An Adaptive Recursive LMS filter," Proc. IEEE, vol.64, pp.1622-1624, November 1976.

[28] H. Fan, "A New Adaptive IIR Filter," IEEE. Trans. Circuits and Systems, vol. CAS-33, no. 10, pp. 939-946, 1986.

[29] F.K. Soong & A.M. Peterson. "Fast Least Sqaures (LS) In The Voice Echo Cancellation Application," Proc. ICASSP 1982. pp. 1898-1430. 1982

[30] S.D. Stearns, G.R. Elliot & N. Ahmed, "On Adaptive Recursive filtering," Proc. 10th Asilomar Conf. Circuits, Sustems and Computers, pp.5-10, 1976.

[31] C.W.K. Gritton & D.W. Lin, "Echo Cancellation Algortihms," IEEE ASSP magazine, April 1984, pp. 30-37.

[32] L.J. Griffiths, A simple Adaptive Algorithm For Real-Time Processing In Antenna Arrays," Proc. IEEE, vol. 57, pp.1696-1704 October 1969.

[33] S.D. Stearns, "Error Surfaces Of Recursive Adaptive filters," IEEE Trans. ASSP, vol ASSP-29, no.3, June 1981.

[34] S.A. White, "An Adaptive Recusive filter," Proc. 9th Asilomat Conf Circuits, Systems and Computers, November 1975, pp.21-25.

[35] T.C Hsia, A Simplified Adaptive Recursive filter Design," Proc. IEEE, vol.69,p1153, September 1981.

[36] B. Widrow & M. Mc Cool, "Comments on 'An Adaptive Recursive LMS Filter'," @roc. IEEE, vol.65, pp.1402-1404, September 1977.

[37] C.R. Johnson & M.G. Larimore, "Comments On and Additions To 'An Adaptive Recursive LMS Filter'," Proc. IEEE vol. 65, pp. 1399- 1401, September 1977.

[38] P.L. Feintuch, *reply* by Feintuch in [37].

[39] T. Murali & B.V. Rao, "An Improved Recursive LMS Algorithm Using a Modified Gradient Filter," Proc. IEEE, vol. 73, no. 8, august 1985.

[40] G. Billoud & M.A. Galland, "Application Of Real Time Recursive filters to Active Control of sound in finitie Length Ducts," Proc. Inst. Acoust. vol. 10, part 2, pp.627-634. April 1988.

[41] L.J. Eriksson, "Active Sound Attenuation Using Adaptive Signal Processing Techniques," Ph.D. Dissertation, Univ. Wisconsin, Madison, August 1985.

[42] L.J. Eriksson, "Active Noise Control Using Adaptive digital signal Processing," Proc. IEEE conf. 1988.

The following program produces the digital impulse respsonses H01,H03,H21,H23 and Hrt and puts the numbers into files of the same names (unless told otherwise) in 1PE14.6 format.

To give the default names H01...etc for the files you must enter <SPACE> name when asked for each filename.

```
      PARAMETER (MAX=1000)
      REAL H01(MAX),H03(MAX),H23(MAX),H21(MAX),HRT(MAX),R1,R2
      REAL LOSS,PERCENTPSAM
      INTEGER L0,L1,L2,L3,L4,H01MAX,H03MAX,H23MAX,H21MAX,HRTMAX
      CHARACTER*5 N01,N03,N23,N21,NRT
      CHARACTER*20 STRING

  1   FORMAT(/,1X,'L0 ?')
  2   FORMAT(/,1X,'L1 ?')
  3   FORMAT(/,1X,'L2 ?')
  4   FORMAT(/,1X,'L3 ?')
  5   FORMAT(/,1X,'L4 ?')
  6   FORMAT(i3)
  7   FORMAT(A)
  8   FORMAT(/,1X,'H01 FILNAME (H01Z) ?')
  9   FORMAT(/,1X,'H03 FILNAME (H03Z) ?')
 10   FORMAT(/,1X,'H23 FILNAME (H23Z) ?')
 11   FORMAT(/,1X,'H21 FILNAME (H21Z) ?')
 12   FORMAT(/,1X,'HRT FILNAME (HRTZ) ?')
 13   FORMAT(/,1X,'R1 ?')
 14   FORMAT(/,1X,'R2 ?')
 15   format(e14.6)
 16   FORMAT(1PE14.6)
 17   FORMAT(1X,'Percent loss per sample period ?')

C ********** READ IN DUCT DIMENSIONS L0,L1,L2,L3 & L4 **********

      WRITE(5,1)
      READ(5,6) L0
      WRITE(5,2)
      READ(5,6) L1
      WRITE(5,3)
      READ(5,6) L2
      WRITE(5,4)
      READ(5,6) L3
      WRITE(5,5)
      READ(5,6) L4
      WRITE(5,17)

C     ********************************************************
C     READ IN PERCENTAGE LOSS PER SAMPLE PERIOD AND CALCULATE
C     LOSS COEFFICIENT
C     ********************************************************

      READ(5,15) PERCENTPSAM
      LOSS=(100.0-PERCENTPSAM)/100.
```

```
C     ********** READ IN REFLECTION COEFFICIENTS R1 & R2 **********

      WRITE(5,13)
      READ(5,15) R1
      WRITE(5,14)
      READ(5,15) R2

C     ********** READ IN FILNAMES OF IMPULSE REPONSES **********

      WRITE(5,8)
      READ(5,7) N01

      IF(N01.EQ.' ') N01='H01Z'
      WRITE(5,9)
      READ(5,7) N03
      IF(N03.EQ.' ') N03='H03Z'
      WRITE(5,10)
      READ(5,7) N23
      IF(N23.EQ.' ') N23='H23Z'
      WRITE(5,11)
      READ(5,7) N21
      IF(N21.EQ.' ') N21='H21Z'
      WRITE(5,12)
      READ(5,7) NRT
      IF(NRT.EQ.' ') NRT='HRT'

C     ********** FIND LENGTH OF EACH FIR ELEMENT **********

      H01MAX=1+L1+(2*L0)+(2*L2)+(2*L3)+(2*L4)
      H03MAX=1+L1+L2+L3+(2*L0)+(2*L4)
      H23MAX=1+L3+(2*L0)+(2*L1)+(2*L2)+(2*L4)
      H21MAX=1+L2+(2*L0)+(2*L1)+(2*L3)+(2*L4)
      HRTMAX=1+2*(L0+L1+L2+L3+L4)

100   FORMAT(1X,'H01Z Dimension = ',T20,i3)
101   FORMAT(1x,'H03Z Dimension = ',T20,i3)
102   FORMAT(1X,'H23Z Dimension = ',T20,i3)
103   FORMAT(1X,'H21Z Dimension = ',T20,i3)
104   FORMAT(1X,'HRT  Dimension = ',T20,i3)

      WRITE(5,100) H01MAX
      WRITE(5,101) H03MAX
      WRITE(5,102) H23MAX
      WRITE(5,103) H21MAX
      WRITE(5,104) HRTMAX


C     ********** OPEN OUTPUT FILES **********

      OPEN(1,FILE=N01//'.DAT',STATUS='NEW')
      OPEN(2,FILE=N03//'.DAT',STATUS='NEW')
      OPEN(3,FILE=N23//'.DAT',STATUS='NEW')
      OPEN(4,FILE=N21//'.DAT',STATUS='NEW')
      OPEN(6,FILE=NRT//'.DAT',STATUS='NEW')
      OPEN(7,FILE='DUCTDATA.DAT',STATUS='NEW')
```

```
C     ********** ZERO ALL IMPULSE RESPONSES **********

      CALL ZERO(H01,H01MAX)
      CALL ZERO(H03,H03MAX)
      CALL ZERO(H23,H23MAX)
      CALL ZERO(H21,H21MAX)
      CALL ZERO(HRT,HRTMAX)

C     ********** FILL ARRAYS WITH REQUIRED COEFFS **********

      H01(1+L1)=H01(1+L1)+(LOSS**L1)
      H01(1+L1+2*L0)=H01(1+L1+2*L0)+R1*(LOSS**(L1+2*L0))

      TINT=L1+(2*L2)+(2*L3)+(2*L4)
      H01(1+TINT)=H01(TINT)+R2*(LOSS**TINT)

      TINT=L1+(2*L0)+(2*L2)+(2*L3)+(2*L4)
      H01(1+TINT)=H01(1+TINT)+R1*R2*(LOSS**TINT)

      TINT=L1+L2+L3
      H03(1+TINT)=H03(1+TINT)+(LOSS**TINT)
      TINT=L1+L2+L3+(2*L0)
      H03(1+TINT)=H03(1+TINT)+R1*(LOSS**TINT)
      TINT=L1+L2+L3+(2*L4)
      H03(1+TINT)=H03(1+TINT)+R2*(LOSS**TINT)
      TINT=L1+L2+L3+(2*L0)+(2*L4)
      H03(1+TINT)=H03(1+TINT)+R1*R2*(LOSS**TINT)

      H23(1+L3)=H23(1+L3)+(LOSS**L3)
      H23(1+L3+(2*L4))=H23(1+L3+(2*L4))+R2*(LOSS**(L3+(2*L4)))
      TINT=L3+(2*L0)+(2*L1)+(2*L2)
      H23(1+TINT)=H23(1+TINT)+R1*(LOSS**TINT)
      TINT=L3+(2*L0)+(2*L1)+(2*L2)+(2*L4)
      H23(1+TINT)=H23(1+TINT)+R1*R2*(LOSS**TINT)

      H21(1+L2)=H21(1+L2)+(LOSS**L2)
      TINT=L2+(2*L0)+(2*L1)
      H21(1+TINT)=H21(1+TINT)+R1*(LOSS**TINT)
      TINT=L2+(2*L3)+(2*L4)
      H21(1+TINT)=H21(1+TINT)+R2*(LOSS**TINT)
      TINT=L2+(2*L0)+(2*L1)+(2*L3)+(2*L4)
      H21(1+TINT)=H21(1+TINT)+R1*R2*(LOSS**TINT)

      HRT(1+2*(L0+L1+L2+L3+L4))=R1*R2*(LOSS**(2*(L0+L1+L2+L3+L4)))


C     ********** WRITE IR's TO FILES **********

      WRITE(1,16) (H01(I),I=1,H01MAX)
      WRITE(2,16) (H03(I),I=1,H03MAX)
      WRITE(3,16) (H23(I),I=1,H23MAX)
      WRITE(4,16) (H21(I),I=1,H21MAX)
      WRITE(6,16) (HRT(I),I=1,HRTMAX)
```

```
105   FORMAT(///,1X,A,/)
106   FORMAT(/,1X,'Impulse Responses for duct with :')
107   FORMAT(/,1X,'L0 is ',i3)
108   FORMAT(/,1X,'L1 is ',i3)
109   FORMAT(/,1X,'L2 is ',i3)
110   FORMAT(/,1X,'L3 is ',i3)
111   FORMAT(/,1X,'L4 is ',i3)
112   FORMAT(/,1X,'R1 is ',1PE11.3)
113   FORMAT(/,1X,'R2 is ',1PE11.3)
114   FORMAT(/,1X,'Filename is ',T20,A)
115   FORMAT(/,1X,'% LOSS per sample period is ',1PE11.3)

      WRITE(7,106)
      WRITE(7,115) PERCENTPSAM
      WRITE(7,107) L0
      WRITE(7,108) L1
      WRITE(7,109) L2
      WRITE(7,110) L3
      WRITE(7,111) L4
      WRITE(7,112) R1
      WRITE(7,113) R2

      STRING='H01Z :'
      WRITE(7,105) STRING
      WRITE(7,16) (H01(I),I=1,H01MAX)
      WRITE(7,114) N01
      WRITE(7,100) H01MAX

      STRING='H03Z :'
      WRITE(7,105) STRING
      WRITE(7,16) (H03(I),I=1,H03MAX)
      WRITE(7,114) N03
      WRITE(7,101) H03MAX

      STRING='H23Z :'
      WRITE(7,105) STRING
      WRITE(7,16) (H23(I),I=1,H23MAX)
      WRITE(7,114) N23
      WRITE(7,102) H23MAX

      STRING='H21Z :'
      WRITE(7,105) STRING
      WRITE(7,16) (H21(I),I=1,H21MAX)
      WRITE(7,114) N21
      WRITE(7,103) H21MAX

      STRING='Hround trip :'
      WRITE(7,105) STRING
      WRITE(7,16) (HRT(I),I=1,HRTMAX)
      WRITE(7,114) NRT
      WRITE(7,104) HRTMAX
```

```
C     ********** CLOSE FILES **********

      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
      CLOSE(6)
      CLOSE(7)

      END

C     *************************
C     FUNCTIONS AND SUBROUTINES
C     *************************

      SUBROUTINE ZERO(A,N)
      REAL A(N)
      DO 100 J=1,N
  100 A(J)=0.
      RETURN
      END
```

Appendix II - The main simulation program NEWSIM

The following FORTRAN program was used to simulate the performance of several different adaptive algorithms when used in system identification and to adapt a controller within a one dimensional ventilation duct.

The program is a stand alone module producing files containing numerical data. The source program is available on disk in DOS format or Acorn BBC format.

At the begining of a run, the user is asked for the name of an input file. The format of this input file is shown below. The program prompt, variable name and the format that the variable is required to in within the input file are shown.

INPUT PROMPT	VARIABLE	FORMAT
Which Algorithm (enter 1,2,3,4 or 5)')	ALG	I7
Feintuch RLMS.....1		
RLMS2.....2		
Stearns' Algorithm.....3		
FAN Algorithm.....4		
FIR LMS Algorithm.....5		
How many iterations	NOIT	I7
Convergence parameter for direct filter A	MUEA	E13.5
Convergence parameter for recursive filter B	MUEB	E13.5
DO YOU WANT COEFFICIENTS OUTPUT	YN	A
DO YOU WANT THE SIGNAL FILE OUTPUT	YN	A
How many coefficients in H01Z	D01Z	I7
Input data filename :-	AFILE	A
How many coefficients in H01P	D01P	I7
Input data filename :-	AFILE	A
How many coefficients in H03Z	D03Z	I7
Input data filename :-	AFILE	A
How many coefficients in H03P	D03P	I7
Input data filename :-	AFILE	A
How many coefficients in H23Z	D23Z	I7
Input data filename :-	AFILE	A
How many coefficients in H23P	D23P	I7
Input data filename :-	AFILE	A
How many coefficients in H21Z	D21Z	I7
Input data filename :-	AFILE	A
How many coefficients in H21P	D21P	I7
Input data filename :-	AFILE	A
How many coefficients in HTZ	DTZ	I7
Input data filename :-	AFILE	A
How many coefficients in HTP	DTP	I7
Input data filename :-	AFILE	A
How many coefficients in HUZ	DUZ	I7
Input data filename :-	AFILE	A
How many coefficients in HUP	DUP	I7
Input data filename :-	AFILE	A
How many coefficients in HESZ	DESZ	I7
Input data filename :-	AFILE	A
How many coefficients in HESP	DESP	I7
Input data filename :-	AFILE	A
How many coefficients in direct filter "A"	FF	I7

INPUT PROMPT	VARIABLE	FORMAT
How many coefficients in recursive filter "B"	FB	I7
Mean square error taken over how many samples	MEM	I7
At what interval should information be output	STEP	I7
Start outputting data at iteration number	STARTPOINT	I7
Start filtering at what iteration number	SWON	I7
Start adapting filter at what iteration	ADSTART	I7
Output file name	OFILE	A
Type in remarks (Single line only)')	REMARK	A

```
*************************************************************************
 The actual FORTRAN 77 program is shown below
*************************************************************************

C      *********************
C      INITIALIZE VARIABLES
C      *********************

       IMPLICIT NONE

       CHARACTER*2 YN
       CHARACTER*4 OUTFILE,INFILE
       CHARACTER*70 REMARK
       CHARACTER*10 ALGORITHM

       INTEGER I,J,K,D01Z,D01P,D03Z,D03P,D23Z,D23P,D21Z,D21P
       INTEGER DTZ,DTP,DUZ,DUP,MEM,MAX,FF,FB,SWON,FFOUT,ADSTART
       INTEGER ITERATIONS,STEP,FLAG,DEN,LAST,STARTPOINT,NUMP
       INTEGER DESZ,DESP,CFLAG,TFLAG,alg

       PARAMETER (MAX=1024)
       REAL A(MAX),B(MAX)
       REAL AIN,ADIRECT,BRECURSIVE,FILTEROUT
       REAL H01Z(MAX),H01P(MAX)
       REAL H01IN,H01DIRECT,H01RECURSIVE,H01OUT
       REAL H03Z(MAX),H03P(MAX)
       REAL H03IN,H03DIRECT,H03RECURSIVE,H03OUT
       REAL H21Z(MAX),H21P(MAX)
       REAL H21IN,H21DIRECT,H21RECURSIVE,H21OUT
       REAL H23Z(MAX),H23P(MAX)
       REAL H23IN,H23DIRECT,H23RECURSIVE,H23OUT
       REAL HXPFDIRECT,HXPFRECURSIVE,HXPFOUT

       REAL HYPFDIRECT,HYPFRECURSIVE,HYPFOUT
       REAL HXPFFB(MAX),HYPFFB(MAX)
       REAL HTZ(MAX),HTP(MAX)
       REAL HTIN,HTDIRECT,HTRECURSIVE,HTOUT
       REAL HUZ(MAX),HUP(MAX)
       REAL HUIN,HUDIRECT,HURECURSIVE,HUOUT
       REAL AFF(MAX),BFB(MAX)
       REAL HESIN,HESDIRECT,HESRECURSIVE,HESOUT
       REAL HESZ(MAX),HESP(MAX)
       REAL HESFF(MAX),HESFB(MAX)
       REAL H01FF(MAX),H01FB(MAX)
       REAL H03FF(MAX),H03FB(MAX)
       REAL H21FF(MAX),H21FB(MAX)
       REAL H23FF(MAX),H23FB(MAX)
       REAL HTFF(MAX),HTFB(MAX)
       REAL HUFF(MAX),HUFB(MAX)
       REAL ERROR(MAX),MSE(1000000)
       REAL XDASHED(MAX),YDASHED(MAX)
       REAL XDOUBLEDASHED(MAX),YDOUBLEDASHED(MAX)
       REAL RANDOM,NOISE,MUEA,MUEB

       K=12357
       CFLAG=0
       TFLAG=0
       FLAG=0
```

```
      NUMP=0
      H01OUT=0.0
      H03OUT=0.0
      H21OUT=0.0
      H23OUT=0.0
      HXPFOUT=0.0
      HYPFOUT=0.0
      DEN=0
      FFOUT=1

C     **********************************
C     READ INFO TO BE USED IN ADAPTION
C     **********************************

2001  FORMAT(A)
2002  FORMAT(/,1X,'Input file name ?')
      WRITE(5,2002)
      READ(5,2001) INFILE
      OPEN(9,FILE=INFILE//'.IN',STATUS='OLD')

2003  FORMAT(1X,'Which Algorithm (enter 1,2,3,4 or 5)')
2004  FORMAT(1X,'Feintuch RLMS.....1')
2005  FORMAT(1X,'RLMS2.....2')
2006  FORMAT(1X,'STEARNS Algorithm.....3')
2007  FORMAT(1X,'FAN Algorithm.....4')
2010  format(1x,'FIR LMS Algorithm.....5',/)
      write(5,2004)
      write(5,2005)
      write(5,2006)
      write(5,2007)
      write(5,2010)
      write(5,2003)
      read(9,2008) alg


2008  format(i7)

      if(alg.eq.1) algorithm='RLMS1'
      if(alg.eq.2) algorithm='RLMS2'
      if(alg.eq.3) algorithm='STEARNS'
      if(alg.eq.4) algorithm='FAN'
      if(alg.eq.5) algorithm='LMS'

2000  FORMAT(/,1X,'How many iterations')
1999  FORMAT(/,1X,'Convergence parameter for direct filter A?')
1998  FORMAT(/,1X,'Convergence parameter for recursive filter B?')
1997  FORMAT(E13.5)
      WRITE(5,2000)
      READ(9,999) ITERATIONS
      WRITE(5,1999)
      READ(9,1997) MUEA
      MUEA=-MUEA
      WRITE(5,1998)
      READ(9,1997) MUEB
      MUEB=-MUEB
```

```
1996  FORMAT(1X,'DO YOU WANT COEFFICIENTS OUTPUT ?')
      WRITE(5,1996)
      READ(9,1994) YN
      IF(YN.EQ.'Y'.OR.YN.EQ.'y') CFLAG=1
1995  FORMAT(1X,'DO YOU WANT THE SIGNAL FILE OUTPUT ?')
      WRITE(5,1995)
      READ(9,1994) YN
      IF(YN.EQ.'Y'.OR.YN.EQ.'y') TFLAG=1

1994  FORMAT(A)
999   FORMAT(I7)
998   FORMAT(/,1X,'How many coefficients in  H01Z ?')
      WRITE(5,998)
      READ(9,999) D01Z
      IF(D01Z.NE.0) THEN
      CALL ARRAYFILL(H01Z,1,D01Z)
997   FORMAT(/,1X,'How many coefficients in  H01P ?')
      WRITE(5,997)
      READ(9,999) D01P
      IF(D01P.NE.0) THEN
      CALL ARRAYFILL(H01P,1,D01P)
      ELSE
      D01P=2
      H01P(2)=0.0
      END IF
      ELSE
      D01Z=1
      H01Z(1)=1.0
      D01P=2
      H01P(2)=0.0
      END IF

996   FORMAT(/,1X,'How many coefficients in  H03Z ?')
      WRITE(5,996)
      READ(9,999) D03Z
      IF (D03Z.NE.0) THEN
      CALL ARRAYFILL(H03Z,1,D03Z)
995   FORMAT(/,1X,'How many coefficients in  H03P ?')
      WRITE(5,995)
      READ(9,999) D03P
      IF(D03P.NE.0) THEN
      CALL ARRAYFILL(H03P,1,D03P)
      ELSE
      D03P=2
      H03P(2)=0.0
      END IF
      ELSE
      D03Z=1
      H03Z(1)=1.0
      D03P=2
      H03P(2)=0.0
      END IF
```

```
994   FORMAT(/,1X,'How many coefficients in  H23Z ?')
      WRITE(5,994)
      READ(9,999) D23Z
      IF (D23Z.NE.0) THEN
      CALL ARRAYFILL(H23Z,1,D23Z)
993   FORMAT(/,1X,'How many coefficients in  H23P ?')
      WRITE(5,993)
      READ(9,999) D23P
      IF(D23P.NE.0) THEN
      CALL ARRAYFILL(H23P,1,D23P)
      ELSE
      D23P=2
      H23P(2)=0.0
      END IF

      ELSE
      D23Z=1
      H23Z(1)=1.0
      D23P=2
      H23P(2)=0.0
      END IF
992   FORMAT(/,1X,'How many coefficients in  H21Z ?')
      WRITE(5,992)
      READ(9,999) D21Z
      IF (D21Z.NE.0) THEN
      CALL ARRAYFILL(H21Z,1,D21Z)
991   FORMAT(/,1X,'How many coefficients in  H21P ?')
      WRITE(5,991)
      READ(9,999) D21P
      IF(D21P.NE.0) THEN
      CALL ARRAYFILL(H21P,1,D21P)
      ELSE
      D21P=2
      H21P(2)=0.0
      END IF
      ELSE
      D21Z=2
      H21Z(2)=1.0
      D21P=2
      H21P(2)=0.0
      END IF
```

```
990   FORMAT(/,1X,'How many coefficients in  HTZ ?')
      WRITE(5,990)
      READ(9,999) DTZ
      IF (DTZ.NE.0) THEN
      CALL ARRAYFILL(HTZ,1,DTZ)
989   FORMAT(/,1X,'How many coefficients in  HTP ?')
      WRITE(5,989)
      READ(9,999) DTP
      IF(DTP.NE.0) THEN
      CALL ARRAYFILL(HTP,1,DTP)
      ELSE
      DTP=2
      HTP(2)=0.0
      END IF
      ELSE
      DTZ=1
      HTZ(1)=1.0
      DTP=2
      HTP(2)=0.0
      END IF

988   FORMAT(/,1X,'How many coefficients in  HUZ ?')
      WRITE(5,988)
      READ(9,999) DUZ
      IF (DUZ.NE.0) THEN
      CALL ARRAYFILL(HUZ,1,DUZ)
987   FORMAT(/,1X,'How many coefficients in  HUP ?')
      WRITE(5,987)
      READ(9,999) DUP
      IF(DUP.NE.0) THEN
      CALL ARRAYFILL(HUP,1,DUP)
      ELSE
      DUP=2
      HUP(2)=0.0
      END IF
      ELSE
      DUZ=1
      HUZ(1)=1.0
      DUP=2
      HUP(2)=0.0
      END IF
```

```
888   FORMAT(/,1X,'How many coefficients in  HESZ ?')
      WRITE(5,888)
      READ(9,999) DESZ
      IF (DESZ.NE.0) THEN
      CALL ARRAYFILL(HESZ,1,DESZ)
887   FORMAT(/,1X,'How many coefficients in  HUES ?')
      WRITE(5,887)
      READ(9,999) DESP
      IF(DESP.NE.0) THEN
      CALL ARRAYFILL(HESP,1,DESP)
      ELSE
      DESP=2
      HESP(2)=0.0
      END IF
      ELSE
      DESZ=1
      HESZ(1)=1.0
      DESP=2
      HESP(2)=0.0
      END IF

986   FORMAT(/,1X,'How many coefficients in direct filter "A" ? ')
985   FORMAT(/,1X,'How many coefficients in recursive filter "B" ? ')
      WRITE(5,986)
      READ(9,999) FF
      IF (FF.NE.0) THEN
      CALL ARRAYFILL(A,1,FF)
      WRITE(5,985)
      READ(9,999) FB
      IF(FB.NE.0) THEN
      CALL ARRAYFILL(B,1,FB)
      ELSE
      FB=2
      B(2)=0.0
      END IF
      ELSE
      FF=1
      A(1)=1.0
      FB=2
      B(2)=0.0
      END IF

984   FORMAT(/,1X,'Mean square error taken over how many samples ? ')
      WRITE(5,984)
      READ(9,999) MEM

983   FORMAT(/,1X,'At what interval should information be output ?')
      WRITE(5,983)
      READ(9,999) STEP

982   FORMAT(/,1X,'Start outputting data at iteration number ?')
981   WRITE(5,982)
      READ(9,999) STARTPOINT
      IF (STARTPOINT.LT.1) GO TO 981
```

```
111   FORMAT(/,1X,'Start filtering at what iteration number ?')
110   write(5,111)
      READ(9,999) swon
      if (swon.lt.1) go to 110

109   FORMAT(/,1X,'Start adapting filter at what iteration ?')
      write(5,109)
108   READ(9,999) adstart
      if (adstart.lt.1) go to 108

C     ************************
C     ZERO ALL WORKING ARRAYS
C     ************************

      CALL ZERO(AFF,FF)
      CALL ZERO(BFB,FB)
      CALL ZERO(H03FF,D03Z)
      CALL ZERO(H03FB,D03P)
      CALL ZERO(H01FF,D01Z)
      CALL ZERO(H01FB,D01P)
      CALL ZERO(H23FF,D23Z)
      CALL ZERO(H23FB,D23P)
      CALL ZERO(H21FF,D21Z)
      CALL ZERO(H21FB,D21P)
      CALL ZERO(HXPFFB,FB)
      CALL ZERO(HYPFFB,FB)
      CALL ZERO(HTFF,DTZ)
      CALL ZERO(HTFB,DTP)
      CALL ZERO(HUFF,DUZ)
      CALL ZERO(HUFB,DUP)
      CALL ZERO(MSE,ITERATIONS)
      CALL ZERO(ERROR,MEM)
      CALL ZERO(XDASHED,FF)
      CALL ZERO(YDASHED,FB)
      CALL ZERO(XDOUBLEDASHED,FF)
      CALL ZERO(YDOUBLEDASHED,FB)
      CALL ZERO(HESFF,DESZ)
      CALL ZERO(HESFB,DESP)

C     ******************************************************
C     OPEN FILES FOR INFORMATION OUTPUT DURING EXPERIMENT
C     ******************************************************

980   FORMAT(/,1X,'Output file name ?')
      WRITE(5,980)
      READ(9,978) OUTFILE

      OPEN(1,FILE=OUTFILE//'.DAT',STATUS='NEW')
      OPEN(2,FILE=OUTFILE//'.MSE',STATUS='NEW')
      OPEN(3,FILE=OUTFILE//'.INFO',STATUS='NEW')
      OPEN(4,FILE=OUTFILE//'.SCOEFFS',STATUS='NEW')
      OPEN(8,FILE=OUTFILE//'.TST',STATUS='NEW')
```

```
979   FORMAT(/,1X,'Type in remarks (Single line only)')
      WRITE(5,979)
978   FORMAT(A)
      READ(9,978) REMARK

      CLOSE(9)

333   format(1x,'Simulation using FEINTUCH RLMS algorithm')
334   FORMAT(1X,'Simulation using the RLMS2 algorithm')
335   FORMAT(1X,'Simulation using the FAN algorithm')
336   FORMAT(1X,'Simulation using the STEARNS algorithm')
337   format(1x,'Simulation using the LMS algorithm')

      IF(ALGORITHM.EQ.'RLMS1') WRITE(1,333)
      IF(ALGORITHM.EQ.'RLMS2') WRITE(1,334)
      IF(ALGORITHM.EQ.'FAN') WRITE(1,335)
      IF(ALGORITHM.EQ.'STEARNS') WRITE(1,336)

      IF(ALGORITHM.EQ.'LMS') THEN

      WRITE(1,337)
      MUEB=0.0

      END IF

977   FORMAT(1X,'Remarks : ',T15,A)
      WRITE(1,977) REMARK

976   FORMAT(1X,'Output file set is called ',T40,A)

      WRITE(1,976) OUTFILE

975   FORMAT(1X,'Number of iterations is ',T40,I7)
      WRITE(1,975) ITERATIONS

974   FORMAT(1X,'Output step is ',T40,I7)
      WRITE(1,974) STEP

456   FORMAT(1X,'Filtering started at iteration ',I7)
      WRITE(1,456) SWON

123   FORMAT(1X,'Adaptation started at iteration ',i7)
      WRITE(1,123) ADSTART

664   FORMAT(1X,'Mean Square Error averaged over',T40,I4,'
     *Iterations')

      WRITE(1,664) MEM

973   FORMAT(/,1X,'Direct convergence coefficient is ',T40,1PE11.3)
972   FORMAT(1X,'Recursive convergence coefficient is ',T40,1PE11.3)
971   FORMAT(/,1X,'Number of direct taps is',T40,I7)
970   FORMAT(1X,'Number if recursive taps is',T40,I7)
      WRITE(1,973) MUEA
      WRITE(1,972) MUEB
      WRITE(1,971) FF
      WRITE(1,970) FB
```

```
969   FORMAT(/,1X,'***** ACOUSTIC VECTOR H03Z & H03P *****',/)
968   FORMAT(/,1X,'***** ACOUSTIC VECTOR H21Z & H21P *****',/)
967   FORMAT(/)

966   FORMAT(1P4E14.6)
      WRITE(1,969)
      WRITE(1,966) (H03Z(I),I=1,D03Z)
      WRITE(1,967)
      WRITE(1,966) (H03P(I),I=1,D03P)
      WRITE(1,968)
      WRITE(1,966) (H21Z(I),I=1,D21Z)
      WRITE(1,967)
      WRITE(1,966) (H21P(I),I=1,D21P)

965   FORMAT(/,1X,'***** ACOUSTIC VECTOR H01Z & H01P *****',/)
964   FORMAT(/,1X,'***** ACOUSTIC VECTOR H23Z & H23P *****',/)
      WRITE(1,965)
      WRITE(1,966) (H01Z(I),I=1,D01Z)
      WRITE(1,967)
      WRITE(1,966) (H01P(I),I=1,D01P)
      WRITE(1,964)
      WRITE(1,966) (H23Z(I),I=1,D23Z)
      WRITE(1,967)
      WRITE(1,966) (H23P(I),I=1,D23P)


666   FORMAT(/,1X,'***** FILTER HTz/(1-HTp) *****',/)
665   FORMAT(/,1X,'***** FILTER HUz/(1-HUp) *****',/)
      WRITE(1,666)
      WRITE(1,966) (HTZ(I),I=1,DTZ)
      WRITE(1,967)
      WRITE(1,966) (HTP(I),I=1,DTP)
      WRITE(1,665)
      WRITE(1,966) (HUZ(I),I=1,DUZ)

      WRITE(1,967)
      WRITE(1,966) (HUP(I),I=1,DUP)

555   FORMAT(/,1X,'***** ERROR SMOOTHING HESz/(1-HESp) *****',/)
      WRITE(1,555)
      WRITE(1,966) (HESZ(I),I=1,DESZ)
      WRITE(1,967)
      WRITE(1,966) (HESP(I),I=1,DESP)

963   FORMAT(/,1X,'***** MODEL FEEDFORWARD VECTOR A *****',/)
962   FORMAT(/,1X,'***** MODEL FEEDBACK VECTOR B *****',/)
      WRITE(1,963)
      WRITE(1,966) (A(I),I=1,FF)
      WRITE(1,962)
      WRITE(1,966) (B(I),I=1,FB)


961   FORMAT(/,1X,'ITERATIONS',22X,'MEAN SQUARE ERROR',/)
      WRITE(1,961)
```

```
C ************************************************************
C OUTPUT THE FOLLOWING INFO TO .SCOEFFS FILE :
C i) Comment
C ii) Number of iterations intended for experiment
C iii) Iteration number at which output to files should start.
C iv) Output Step.
C v) Number of direct coefficients in the adaptive filter.
C vi) Number of recursive coefficients in the adaptive
C filter.

C ************************************************************


960    FORMAT(A)
959    FORMAT(I7)
       WRite(4,960) REMARK
       WRite(4,959) ITERATIONS
       WRite(4,959) STARTPOINT
       WRite(4,959) STEP
       WRite(4,959) FF
       WRite(4,959) FB-1

958    FORMAT(E13.5)

       J=0
       WRite(4,959) J
       WRite(4,958) (A(I),I=1,FF)
       WRite(4,958) (B(I),I=2,FB)

957    FORMAT(T5,'ITER',T11,'H0OUT',T23,'H21OUT',T35,'h23OUT'
      *,t47,'FILIN',T59'FILOUT'
      *,T71,'ADIRECT',T83,'BRECUR',T95,'ERROR',T107,'X"',T119,'Y"')
       WRITE(8,957)
       WRITE(8,967)

C      ********************
C      BEGIN ITERATIVE LOOP
C      ********************

       DO 100 J=1,ITERATIONS
```

```
C     **********************
C     SHIFT ALL DATA ARRAYS
C     **********************

      CALL SHIFT(AFF,FF)
      CALL SHIFT(BFB,FB)
      CALL SHIFT(H01FF,D01Z)
      CALL SHIFT(H01FB,D01P)
      CALL SHIFT(H03FF,D03Z)
      CALL SHIFT(H03FB,D03P)
      CALL SHIFT(H21FF,D21Z)
      CALL SHIFT(H21FB,D21P)
      CALL SHIFT(H23FF,D23Z)
      CALL SHIFT(H23FB,D23P)
      CALL SHIFT(HXPFFB,FB)
      CALL SHIFT(HYPFFB,FB)
      CALL SHIFT(HTFF,DTZ)
      CALL SHIFT(HTFB,DTP)
      CALL SHIFT(HUFF,DUZ)
      CALL SHIFT(HUFB,DUP)
      CALL SHIFT(ERROR,MEM)
      CALL SHIFT(XDASHED,FF)
      CALL SHIFT(YDASHED,FB)
      CALL SHIFT(XDOUBLEDASHED,FF)
      CALL SHIFT(YDOUBLEDASHED,FB)
      CALL SHIFT(HESFF,DESZ)
      CALL SHIFT(HESFB,DESP)


C     ********************************************************************
C     GET NEW RANDOM SAMPLE AND CONVOLVE WITH ACOUSTICAL FEEDFORWARD
C     PATH H03Z/(1-H03P)
C     ********************************************************************

      NOISE=RANDOM(K)

      H03IN=NOISE
      H03FF(1)=H03IN
      CALL MULTIPLYARRAY(H03Z,1,D03Z,H03FF,H03DIRECT)
      CALL MULTIPLYARRAY(H03P,2,D03P,H03FB,H03RECURSIVE)
      H03OUT=H03DIRECT+H03RECURSIVE
      H03FB(1)=H03OUT

C     ********************************************************************
C     CONVOLVE PREVIOUS DIGITAL FILTER OUTPUT WITH ACOUSTICAL FEEDBACK
C     PATH H21Z/(1-H21P)
C     ********************************************************************

      CALL MULTIPLYARRAY(H21Z,2,D21Z,H21FF,H21DIRECT)
      CALL MULTIPLYARRAY(H21P,2,D21P,H21FB,H21RECURSIVE)
      H21OUT=H21DIRECT+H21RECURSIVE
      H21FB(1)=H21OUT

C     ****************************************************
C     CONVOLVE NEW RANDOM SAMPLE WITH H01Z/(1-H01P) TO GIVE
C     H01OUT.
C     ****************************************************
```

```
      H01IN=NOISE

      H01FF(1)=H01IN
      CALL MULTIPLYARRAY(H01Z,1,D01Z,H01FF,H01DIRECT)
      CALL MULTIPLYARRAY(H01P,2,D01P,H01FB,H01RECURSIVE)
      H01OUT=H01DIRECT+H01RECURSIVE
      H01FB(1)=H01OUT

      IF(J.GE.SWON) THEN

C     ****************************************************************
C     ADD ACOUSTIC FEEDBACK SIGNAL (H21OUT) TO H01OUT AND CONVOLVE
C     WITH DIGITAL FILTER A/(1-B) TO PRODUCE NEW FILTER OUTPUT.
C     ****************************************************************

      AIN=H01OUT+H21OUT
      AFF(1)=AIN
      CALL MULTIPLYARRAY(A,1,FF,AFF,ADIRECT)
      CALL MULTIPLYARRAY(B,2,FB,BFB,BRECURSIVE)
      FILTEROUT=ADIRECT+BRECURSIVE
      H21FF(1)=FILTEROUT
      BFB(1)=FILTEROUT

C     ****************************************************************
C     CONVOLVE FILTER OUTPUT WITH H23Z/(1-H23P) TO GIVE NEW H23OUT
C     ****************************************************************

      H23IN=FILTEROUT
      H23FF(1)=H23IN
      CALL MULTIPLYARRAY(H23Z,1,D23Z,H23FF,H23DIRECT)
      CALL MULTIPLYARRAY(H23P,2,D23P,H23FB,H23RECURSIVE)

      H23OUT=H23DIRECT+H23RECURSIVE
      H23FB(1)=H23OUT


      IF(ALGORITHM.EQ.'FAN'.OR.ALGORITHM.EQ.'STEARNS') THEN

C     ************************************************
C     IF USING FAN'S OR STEARNS' ALGORITHM THEN :
C     FILTER SIGNALS X & Y THROUGH HXPF & HYPF TO GIVE
C     SIGNALS XDASHED (X') & YDASHED (Y') RESPECTIVLEY
C     ************************************************

      HXPFDIRECT=AIN
      CALL MULTIPLYARRAY(B,2,FB,HXPFFB,HXPFRECURSIVE)
      HXPFOUT=HXPFDIRECT+HXPFRECURSIVE
      HXPFFB(1)=HXPFOUT
      XDASHED(1)=HXPFOUT

      IF(ALGORITHM.EQ.'FAN') THEN

      HYPFDIRECT=-H03OUT

      ELSE

      HYPFDIRECT=FILTEROUT
```

```
      END IF


      CALL MULTIPLYARRAY(B,2,FB,HYPFFB,HYPFRECURSIVE)
      HYPFOUT=HYPFDIRECT+HYPFRECURSIVE
      HYPFFB(1)=HYPFOUT
      YDASHED(1)=HYPFOUT
      ELSE
      HXPFOUT=AIN

      IF(ALGORITHM.EQ.'RLMS1') THEN
      HYPFOUT=FILTEROUT
      ELSE
      HYPFOUT=-H03OUT
      END IF

      END IF


C     ***************************************************
C     FILTER SIGNALS X' & Y' THROUGH T & U TO GIVE
C     SIGNALS X'' & Y'' RESPECTIVLEY
C     ***************************************************

      HTIN=HXPFOUT
      HTFF(1)=HTIN
      CALL MULTIPLYARRAY(HTZ,1,DTZ,HTFF,HTDIRECT)
      CALL MULTIPLYARRAY(HTP,2,DTP,HTFB,HTRECURSIVE)
      HTOUT=HTDIRECT+HTRECURSIVE
      HTFB(1)=HTOUT
      XDOUBLEDASHED(1)=HTOUT

      HUIN=HYPFOUT
      HUFF(1)=HUIN
      CALL MULTIPLYARRAY(HUZ,1,DUZ,HUFF,HUDIRECT)
      CALL MULTIPLYARRAY(HUP,2,DUP,HUFB,HURECURSIVE)
      HUOUT=HUDIRECT+HURECURSIVE
      HUFB(1)=HUOUT
      YDOUBLEDASHED(1)=HUOUT
      END IF


C     ***************************
C     CALCULATE NEW ERROR SIGNAL
C     ***************************

      ERROR(1)=H03OUT+H23OUT

C     ********************************************
C     POST FILTER ERROR THROUGH Hes TO GIVE HesOUT
C     ********************************************

      HESIN=ERROR(1)
      HESFF(1)=HESIN
      CALL MULTIPLYARRAY(HESZ,1,DESZ,HESFF,HESDIRECT)
      CALL MULTIPLYARRAY(HESP,2,DESP,HESFB,HESRECURSIVE)
      HESOUT=HESDIRECT+HESRECURSIVE
      HESFB(1)=HESOUT
```

```
      IF(J.GE.ADSTART) THEN

C     ************************************************************
C     UPDATE DIGITAL FILTER USING XDASHED & YDASHED AND HesOUT
C     ************************************************************

      CALL LMSUPDATE(A,1,FF,XDOUBLEDASHED,HESOUT,MUEA)
      CALL LMSUPDATE(B,2,FB,YDOUBLEDASHED,HESOUT,MUEB)

      END IF

      IF(ERROR(1).NE.0.0) THEN


C     ************************************************************
C     CALCULATE MEAN SQUARE ERROR AND OUTPUT INFORMATION.
C     (ERROR IS SQUARED AND AVERAGED OVER 'MEM' ITERATIONS).
C     ************************************************************

      IF (J.LT.MEM+FFOUT) THEN
      DEN=DEN+1
      ELSE
      DEN=MEM
      END IF

      CALL MEANSQUARE(ERROR,DEN,MSE(J))

      ELSE

      MSE(J)=0.0
      FFOUT=FFOUT+1

      END IF


C     ************************************************************
C     IF ITERATION COUNT IS A MULTIPLE OF 'STEP' THEN OUTPUT
C     THE FOLLOWING INFORMATION TO THE .SCOEFFS FILE:
C     i)   Iteration number
C     ii)  Direct coefficients 1..FF
C     iii) Recursive coefficients 2....FB
C     ************************************************************


      IF (J.GE.STARTPOINT) THEN
      FLAG=FLAG+1
      IF ((FLAG.EQ.STEP).or.(J.eq.STARTPOINT)) THEN

      IF(TFLAG.EQ.1) THEN
      WRITE(8,956) J,H0BOUT,H21OUT,H23OUT,AIN,FILTEROUT,ADIRECT,
     *BRECURSIVE,ERROR(1),XDASHED(1),YDASHED(1)
      END IF

  956 FORMAT(1X,I7,10(1PE11.3))
  955 FORMAT(1X,I7,1PE11.3)
  954 FORMAT(I7)
  953 FORMAT(E23.15)
```

```
      WRITE(2,955) J,MSE(J)

      IF (CFLAG.EQ.1) THEN
      write(4,954) J
      WRite(4,953) (A(I),I=1,FF)
      WRite(4,953) (B(I),I=2,FB)
      END IF

      FLAG=0
      NUMP=NUMP+1
      END IF
      END IF


C     ********************************************************
C     CHECK TO SEE IF MSE HAS GONE OVER A REASONABLE VALUE
C     ********************************************************

952   FORMAT(//,1X,'ADAPTATION ABORTED AT ITERATION ',I7,
     *MSE= ',1PE11.3)
      IF (MSE(J).GT.1E+03) THEN
      WRITE(1,952) J,MSE(J)
      WRITE(5,952) J,MSE(J)
      LAST=J
      GO TO 201
      END IF

 100  CONTINUE

C ********************************************************
C OUTPUT FOLLOWING INFORMATION TO .INFO FILE :
C i) NUMber of entries Plus 1 (NUMP).
C ii) Start Point (STARTPOINT).
C iii) Last Iteration output (LAST).
C iv) Number of direct coefficients in model (FF).
C v) Number of recursive coefficients in model (FB).
C vi) Pz dimension.
C vii) Pp dimension.
C viii) Qz dimension.
C ix) Qp dimension.
C x) Modulus of direct mu value.
C xi) Modulus of recursive mu value.
C ********************************************************


 200  LAST=ITERATIONS
 201  WRITE(3,954) NUMP
      WRITE(3,954) STARTPOINT
      WRITE(3,954) LAST
      WRITE(3,954) STEP
      WRITE(3,950) FF
      WRITE(3,950) FB
      WRITE(3,950) D03Z
      WRITE(3,950) D03P
      WRITE(3,950) D21Z
```

```
      WRITE(3,950) D21P
      WRITE(3,951) -MUEA
      WRITE(3,951) -MUEB

 951  FORMAT(1PE10.3)
 950  FORMAT(I3)

C     ******************************************************
C     OUTPUT TABLE OF MSE VERSUS ITERATION NUMBER FOR THE
C     FIRST K (WHERE K IS UP TO 200) ITERATIONS.
C     ******************************************************

      IF (ITERATIONS.LT.200) THEN
      K=ITERATIONS/2
      ELSE
      K=200
      END IF

 949  FORMAT(1X,I7,1P6E11.3)
      DO 300,J=STARTPOINT,STARTPOINT+K,6
 300  WRITE(1,949) J,(MSE(J+I),I=0,5)
 948  FORMAT(/)
      WRITE(1,948)
      DO 400 J=LAST-K,LAST,6
 400  WRITE(1,949) J,(MSE(J+I),I=0,5)

C     ************************************
C     OUTPUT THE ADAPTED ELEMENTS A AND B
C     ************************************

947   FORMAT(/,1X,'MODEL FEEDFORWARD VECTOR',/)
946   FORMAT(/,1X,'MODEL FEEDBACK VECTOR',/)
945   FORMAT(3(E23.15))
      WRITE(1,947)
      WRITE(1,945) (A(I),I=1,FF)
      WRITE(1,946)
      WRITE(1,945) (B(I),I=1,FB)

      OPEN(11,FILE=OUTFILE//'A.DAT',STATUS='NEW')
      OPEN(12,FILE=OUTFILE//'B.DAT',STATUS='NEW')
      OPEN(13,FILE=OUTFILE//'AL.DAT',STATUS='NEW')
      OPEN(14,FILE=OUTFILE//'BL.DAT',STATUS='NEW')

944   format(i7,E15.7)
      WRITE(13,944) (I,A(I),I=1,FF)
      WRITE(14,944) (I,B(I),I=1,FB)
      WRITE(11,943) (A(I),I=1,FF)
      WRITE(12,943) (B(I),I=1,FB)

943   FORMAT(1PE14.6)

C     ******************************
C     CLOSE THE CURRENT OUTPUT FILES
C     ******************************

      CLOSE(1)
```

```
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
      CLOSE(8)
      CLOSE(11)
      CLOSE(12)
      CLOSE(13)
      CLOSE(14)

      END

C     ************************
C     FUNCTIONS AND SUBROUTINES
C     ************************


      SUBROUTINE MEANSQUARE(A,N,MSQ)
      REAL A(N),MSQ
      SUM=0.
      DO 100 J=1,N
100   SUM=SUM+A(J)**2
      MSQ=SUM/N

      RETURN
      END

      REAL FUNCTION RANDOM(I)
      I=2045*I+1
      I=I-(I/1048576)*1048576
      RANDOM=1.-2.*(FLOAT(I+1)/1048577.0)
      RETURN
      END

      SUBROUTINE ZERO(A,N)
      REAL A(N)
      DO 100 J=1,N
 100  A(J)=0.
      RETURN
      END

      SUBROUTINE MULTIPLYARRAY(H,n,M,C,Y)
      REAL H(M),Y,C(M)
      Y=0.
      DO 100 K=n,M
 100  Y=Y+C(K)*H(K)
      RETURN
      END

      SUBROUTINE LMSUPDATE(H,n,M,X,E,K)
      REAL H(M),X(M),E,K
      DO 100 J=n,M
 100  H(J)=H(J)+K*E*X(J)
      RETURN

      END

      SUBROUTINE ARRAYFILL(A,m,N)
      REAL A(N)
```

```
      CHARACTER*10 AFILE
      WRITE(5,3)
      READ(9,4) AFILE
      OPEN (2,FILE=AFILE,STATUS='OLD')
      DO 1 I=m,N
1     READ(2,2) A(I)
      CLOSE(2)
2     FORMAT(1pe14.6)
3     FORMAT(1X,'Input data filename :-')
4     FORMAT(A)
      RETURN
      END

      SUBROUTINE SHIFT(A,N)
      REAL A(N)
      DO 100 J=N-1,1,-1
100   A(J+1)=A(J)
      RETURN
      END
```