CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r.86/06)

Canada

ON THE LMS ALGORITHM FOR HIGH-SPEED ADAPTIVE DIGITAL SIGNAL PROCESSING

by

Emil Savov

A thesis
presented to the School of
Graduate Studies and Research
of the University of Ottawa
in partial fulfillment of the requirements
for the degree of
Master of Applied Sciences
in Electrical Engineering

Ottawa, Canada, 1986

ISBN 0-315-36538-2

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

The University of Ottawa requires the signatures of all persons using or photocopying this theses. Please sign below and give address and date.

ABSTRACT

A modified version of the LMS adaptive algorithm, which uses a filtered value of the error signal, is studied. The effect of error filtering on the speed of convergence and on the misadjustment is evaluated.

The problem of sequential coefficient access in the LMS algorithm is investigated. Some efficient architectures, suitable for VLSI applications, are presented.

Finally a hardware implementation of a high-speed 16-tap FIR adaptive digital filter, based on TDC1028 integrated circuits, is presented.

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to his academic supervisor Dr. Willem Steenaart for his scientific guidance and moral support during the course of the program.

The author also wishes to thank all professors, colleagues and friends at the Department of Electrical Engineering, University of Ottawa for the good academic atmosphere, the advice and the encouragement during the period of his studies.

CONTENTS

LIST OF FIGURES

GENERAL INTRODUCTION

Adaptive signal processing has been a topic of intensive research during the last several decades. The reason for that attention is the considerable potential of adaptive processing methods. Without a claim for strict semantic definition, it can be stated that an adaptive system, or algorithm, is characterized by the ability to adjust a set of its parameters and its behavior in accordance with a specified criterion. This self-optimizing capacity is very attractive, since in practice quite often insufficient information is available about the desired characteristics of the system, or those characteristics change with time.

Adaptive methods are widely used in control applications, robotics, communications and signal processing. In recent years they have become even more attractive, because the technological progress in electronics and computer design makes it possible to implement sophisticated algorithms that require a great amount of computational power in real-time realizations.

Nowadays adaptive signal processing is grown into a mature discipline. Nevertheless new research results come out at an ever increasing rate. But, as pointed out in [8], the trend is now towards specific applications and case studies.

The author's interest in investigating the capacities of the LMS (Least Mean-Square) algorithm came as a result of research on the implementation of high-speed adaptive filtering. Since its introduction in 1960 by Widrow and Hoff [5], the LMS algorithm has been widely used for various applications. A considerable number of other adaptive algorithms have been proposed, like the Kalman and fast Kalman algorithms [33], lattice structures [12], IIR (Infinite Impulse Response) adaptive filters [18], etc.. Some of them are more powerful than the LMS. In spite of that the LMS algorithm remains an attractive and popular method, due to its proven efficiency, relative simplicity of implementation and numerical robustness. This is especially true for high-speed real-time applications.

A great amount of research has been done on the LMS algorithm, it is well understood, and has many modifications. Yet it is possible to further improve its performance and to find new applications.

Chapter one of this thesis is an introduction to adaptive filtering and the LMS algorithm. Its main goal is to introduce the notation and some basic principles that are used further in the text.

In chapter two a modified version of the LMS algorithm is introduced, which uses a filtered value of the error signal in the coefficient update equation. The effect of the error filtering on the performance of the algorithm is studied. It appears that no previous work has been done on this topic.

In chapter three the problem of sequential coefficient access is addressed. Some alternative architectures for VLSI-based † adaptive digital filters are proposed. Some original research can be found here too.

Finally, in chapter four an efficient hardware implementation of a high-speed (10 MHz sampling rate) adaptive transversal digital filter is proposed, which uses TDC1028 integrated circuits.

† VLSI — Very Large Scale Integration

CHAPTER I

INTRODUCTION TO ADAPTIVE FILTERING

1.1 Introduction

The field of adaptive signal processing is very broad. But the issues discussed in the thesis are mainly related to a class of methods known as closed-loop adaptation. The concept of closed-loop adaptation is illustrated in Fig.1.1. The input signal is denoted by x, y stands for the output, d for the *desired output* (also called the reference or training signal), and e is the error signal.

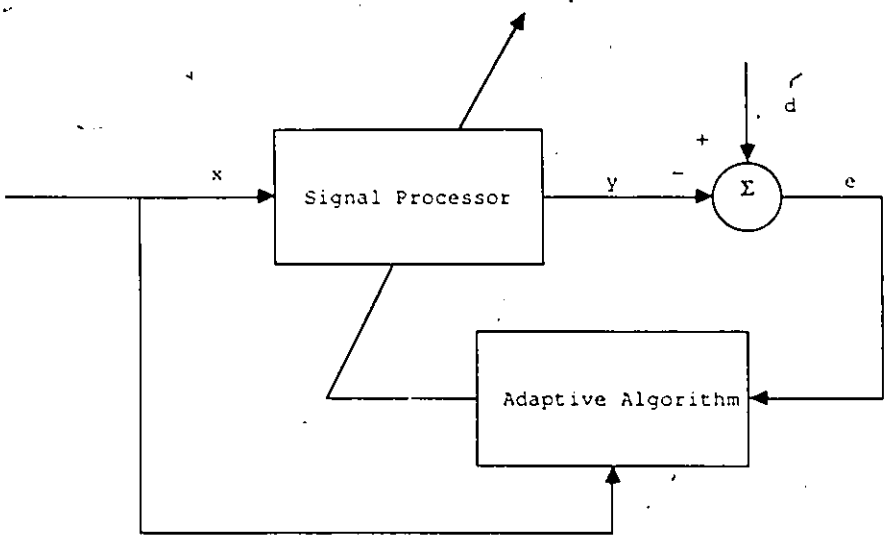


Fig.1.1 Closed-loop adaptation

The adaptive algorithm uses the information from the input signal and the error signal to adjust the variable parameters of the processor in order to optimize a certain performance criterion. Depending on where the signals come from, on what the

performance criterion and the type of signal processor's architecture are, there can be a large variety of algorithms and possible applications.

1.2 The Adaptive Linear Combiner

The adaptive linear combiner (or nonrecursive adaptive filter) is the simplest and most widely used structure for adaptive signal processing. The most general form of the adaptive linear combiner is shown in Fig.1.2. It consists of a set of adjustable weights $w_0, w_1, ..., w_L$, a corresponding signal vector with elements $x_0, x_1, ..., x_L$, a summing device and a single output signal y.

In the case when the input signals are samples of a process, the resulting structure is a single-input adaptive transversal filter. This particular structure is in the core of all further discussions throughout the text.It is shown in Fig.1.3 together with other components necessary for a complete adaptive system. The input and the weight vectors of length (L+1) are:

$$\mathbf{X_k} = [x_k \; x_{k-1} \ldots x_{k-L}]^T \tag{1.1}$$

$$\mathbf{W_k} = [w_{0k} \; w_{1k} \ldots w_{Lk}]^T \tag{1.2}$$

The index k is used to emphasize the fact that the vectors are variable in time. Normally k stands for *iteration number* as will be seen later. The superscript T means transposition.

The name *linear combiner* comes from the fact that for given values of the weight vector, the output signal y_k is a convolution of the input signal with the filter impulse response. The input-output relationship is:

$$y_k = \sum_{i=0}^{L} w_{ik} x_{k-i} \tag{1.3}$$

Or in vector form:

$$y_k = \mathbf{X_k^T W_k} = \mathbf{W_k^T X_k} \tag{1.4}$$

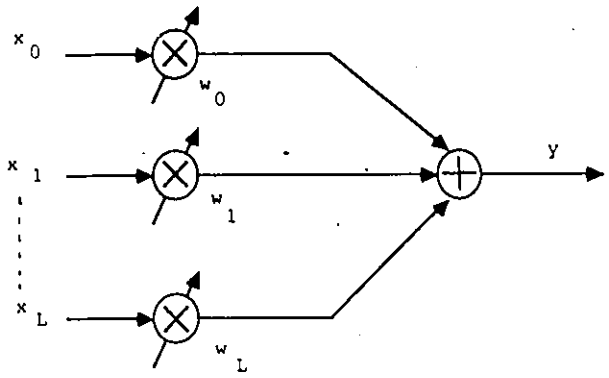


Fig.1.2 Adaptive linear combiner -- general form

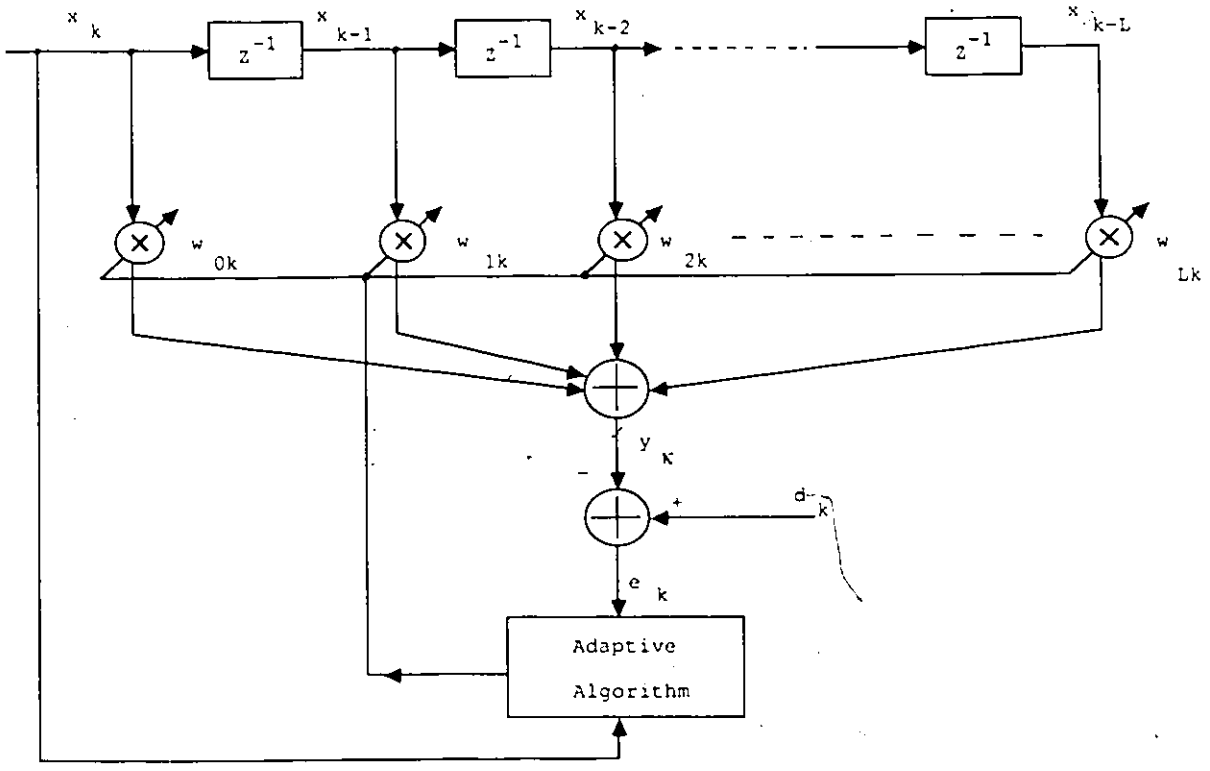


Fig.1.3 Adaptive transversal filter

1.3 Mean-Square Error Performance Criterion

A large class of adaptive algorithms is based on the minimization of the mean-square value, or the average power, of the error signal. By definition:

$$e_k = d_k - y_k = d_k - \mathbf{X_k^T W} = d_k - \mathbf{W^T X_k} \tag{1.5}$$

Here it is assumed that the coefficient vector $\mathbf{W}$ is held constant so the index k is dropped. Then:

$$e_k^2 = d_k^2 + \mathbf{W^T X_k X_k^T W} - 2d_k\mathbf{X_k^T W} \tag{1.6}$$

If e_k, d_k and $\mathbf{X_k}$ are statistically stationary, then taking the expected value of (1.6) over k yields the equation:

$$MSE \triangleq \xi = E[e_k^2] = E[d_k^2] + \mathbf{W^T R W} - 2\mathbf{P^T W} \tag{1.7}$$

Here $\mathbf{R}$ is the $(L+1) \times (L+1)$ square autocorrelation matrix of the input signal and $\mathbf{P}$ is a vector of cross-correlations between the desired response and the input vector.

$$\mathbf{R} = E[\mathbf{X_k X_k^T}] = E\begin{bmatrix} x_k^2 & x_k x_{k-1} & \cdots & x_k x_{k-L} \\ x_{k-1}x_k & x_{k-1}^2 & \cdots & x_{k-1}x_{k-L} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k-L}x_k & \cdots & \cdots & x_{k-L}^2 \end{bmatrix} \tag{1.8}$$

$$\mathbf{P} = E[d_k\mathbf{X_k}] = E[d_k x_k \;\; d_k x_{k-1} \ldots d_k x_{k-L}]^T \tag{1.9}$$

From equation (1.7) it can be seen that the mean-square error is a quadratic function of the elements of the weight vector. It represents a surface in the multidimensional space (or more precisely the (L+1)-dimensional space) of weights and is often called the *performance surface*. Since it is quadratic and non-negative then a single minimum exists, which corresponds to the optimum weight vector. This optimum weight vector can be found by taking the derivative of (1.7) with respect to $\mathbf{W}$ and setting it equal to zero. The derivative itself is the *gradient* and by definition is:

$$\nabla \triangleq \frac{\partial \xi}{\partial \mathbf{W}} = 2(\mathbf{RW} - \mathbf{P}) \tag{1.10}$$

Then solving the matrix equation

$$\nabla = 0 = 2(\mathbf{RW}^* - \mathbf{P}) \tag{1.11}$$

yields

$$\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P} \tag{1.12}$$

where $\mathbf{W}^*$ is the optimum weight vector (Wiener solution). †

Expression (1.12) is the Wiener-Hopf equation in matrix form.The minimum possible mean-square error can be obtained by substituting (1.12) into (1.7):

$$\xi_{min} = E[d_k^2] - \mathbf{P}^{\mathbf{T}}\mathbf{W}^* \tag{1.13}$$

The discussion so far shows that when the signals in the system are stationary and with known statistics, the optimum weights can be computed *a priori* and only once. This kind of situation is more like the matched-filter design problem and a truly adaptive system is not necessary. However, in many practical situations these conditions are not met and an alternative approach is necessary. Such an approach is provided by the methods of gradient search. They represent an iterative way of searching for the minimum of the performance surface, starting from arbitrary weight values. Widely used is the method of *steepest descent.* The iteration equation for the weight vector is:

$$\mathbf{W_{k+1}} = \mathbf{W_k} - \mu\widehat{\nabla}_k \tag{1.14}$$

where μ is a small constant and $\widehat{\nabla}_k$ is the estimate of the gradient at the k-th iteration.

It can be proved [5] that the algorithm is stable and converges to the optimum solution for values of μ in the interval

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{1.15}$$

† The symbol * is used to denote the optimum value, not complex conjugate.

where λ_{max} is the largest eigenvalue of $\mathbf{R}$.

From eqn. 1.5 the gradient can be written as:

$$\nabla = \frac{\partial E[e_k^2]}{\partial \mathbf{W}} = -E[2e_k\mathbf{X_k}] \tag{1.16}$$

Then the weight recursion becomes:

$$\mathbf{W_{k+1}} = \mathbf{W_k} + 2\mu E[e_k\mathbf{X_k}] \tag{1.17}$$

There exist a number of ways of estimating $E[e_k\mathbf{X_k}]$ which will not be discussed here. It is important to note that for the steepest descent algorithm iteration number does not correspond to sampling interval. Usually there are many sampling intervals between two iterations. This is necessary because of the need to estimate ∇_k .

1.4 The LMS Algorithm

The least mean-square (LMS) algorithm can be derived from the algorithm of steepest descent by assuming that the estimate of the mean-square error is the squared error itself. Thus the expected value in (1.17) is replaced by the instantaneous value and the weight vector recursion is:

$$\mathbf{W_{k+1}} = \mathbf{W_k} + 2\mu e_k\mathbf{X_k} \tag{1.18}$$

This is the basic equation of the LMS algorithm. A direct implementation, using one delay line, is shown in Fig.1.4.

The gradient estimate used is noisy but unbiased. The adaptation process will never stop, even after reaching steady-state. The coefficients will continue to oscillate near the optimum solution.

It is worth noting that in the great majority of cases where the LMS algorithm is used iteration number corresponds to sampling interval, i.e. all the filter coefficients are updated at every sampling interval.

For the LMS algorithm the condition for convergence is proved [5, p.103] to be:

$$0 < \mu < \frac{1}{tr[\mathbf{R}]} \tag{1.19}$$

and

$$tr[\mathbf{R}] = (L+1)(signal\ power) \tag{1.20}$$

However in other references [9] a more restrictive but safer bound is imposed:

$$0 < \mu < \frac{1}{3tr[\mathbf{R}]} \tag{1.21}$$

In practice it is normal to work with values of μ ten times smaller than the permissible maximum.

The two most important performance characteristics of the LMS algorithm are the speed of convergence and the residual (or excess) mean-square error after convergence.

The excess mean-square error is inherent in the LMS algorithm and is due to the coefficients' wandering around the optimum solution. A fairly good approximation is given [5, p.110] by:

$$\text{excess MSE} \approx \mu \xi_{min} tr[\mathbf{R}] \tag{1.22}$$

A convenient dimensionless measure can be defined, called *misadjustment*:

$$M = \frac{\text{excess MSE}}{\xi_{min}} \approx \mu tr[\mathbf{R}] \tag{1.23}$$

The convergence rate of each individual coefficient of the filter towards the steady-state value in general is different and is governed by the corresponding eigenvalue of the input correlation matrix $\mathbf{R}$. The curve of coefficient values versus time can be approximated by an exponent with time constant

$$\tau_n \approx \frac{1}{2\mu\lambda_n} \tag{1.24}$$

for $n = 0,\ldots,L$.

The relaxation of the mean-square error (called the *learning curve*) can also be expressed as an exponential function. In the case of equal eigenvalues a simple expression for the second time constant exists:

$$\tau_{mse} \approx \frac{L+1}{4\mu tr[\mathbf{R}]} = \frac{1}{4\mu\lambda} \tag{1.25}$$

since $\sum_{n=0}^{L} \lambda_n = tr[\mathbf{R}]$.

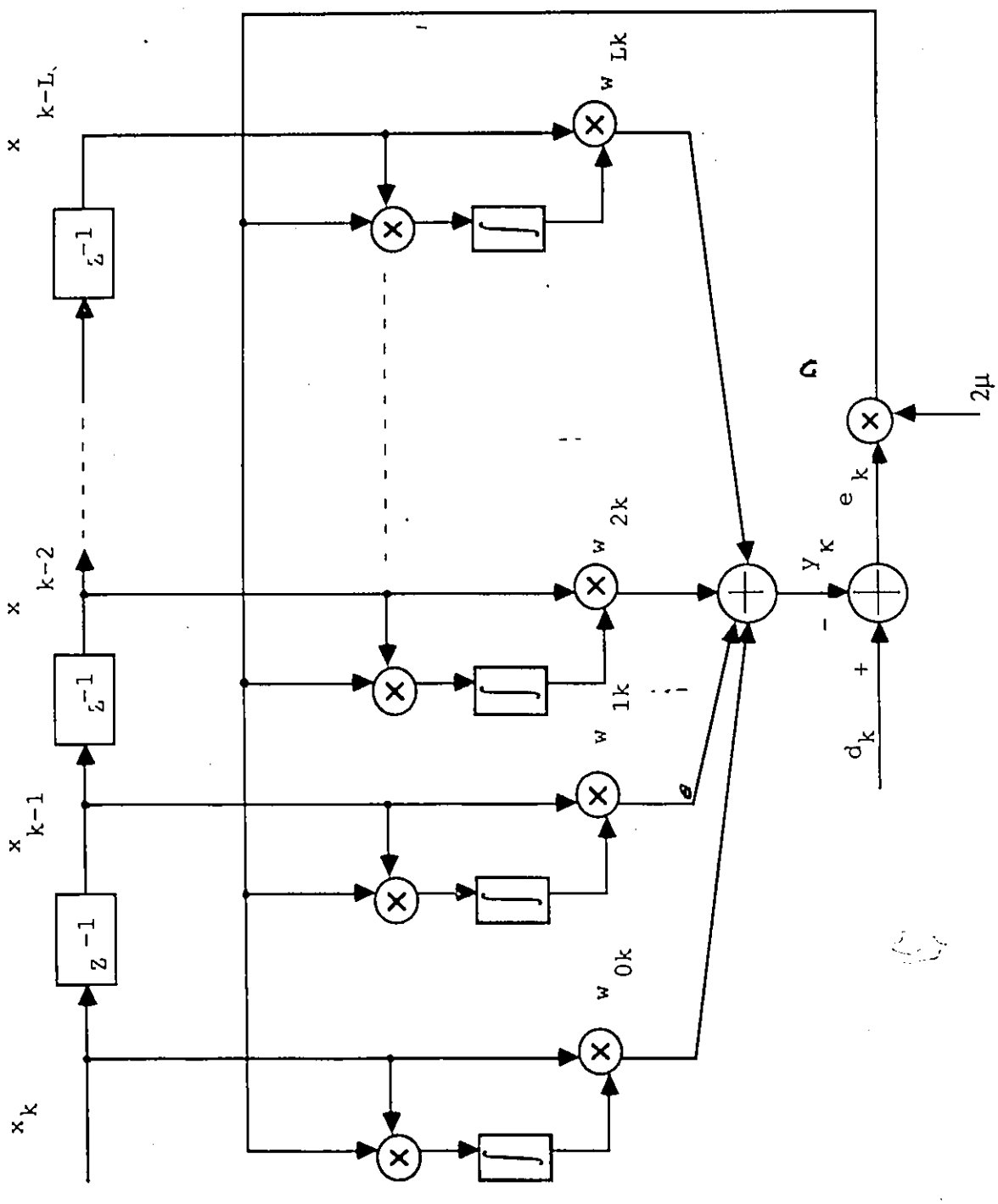


Fig.1.4 A direct implementation of the LMS algorithm

1.5 Other Adaptive Algorithms and Architectures

The main criteria for evaluating the performance of an adaptive system are the speed of convergence, the residual error after steady-state is reached, i.e. the precision with which the desired signal can be approximated and the computational complexity. Some additional requirements that must be taken into account are the numerical stability of the algorithm under various conditions, the effects of nonstationarity on the convergence etc.. As is often the case, no universal solution to all those problems exists. Each adaptive algorithm has its own merits, depending on the particular application.

First of all some alternative algorithms for adapting transversal architectures will be examined.

The LMS algorithm has proven to be very efficient when the eigenvalues of the input correlation matrix are equal, or at least close to each other. On the other hand, a large eigenvalue spread adversely affects the speed of convergence. Each tap coefficient converges with a different time constant, determined by the corresponding eigenvalue. The smaller the eigenvalue, the slower the convergence. A possible remedy is to premultiply the gradient estimate by the inverse of the estimated input autocorrelation matrix. This results in a modification of the LMS algorithm known as the *orthogonalized LMS* algorithm. The coefficient recursion becomes:

$$\mathbf{W_{k+1}} = \mathbf{W_k} + 2\mu e_k \widehat{\mathbf{R}}_\mathbf{k}^{-1} \mathbf{X_k} \tag{1.26}$$

This modification has the effect of making all time constants almost equal, thus improving the overall convergence. The matrix $\widehat{\mathbf{R}}_\mathbf{k}^{-1}$ is updated recursively but this requires a considerable amount of additional computations compared to the conventional LMS algorithm.

Another approach is suggested in a recent publication [22]. The algorithm, called *variable step LMS*, uses a diagonal feedback matrix $\mathbf{M_k}$ instead of the single adaptation coefficient μ.

$$\mathbf{M_k} = \begin{pmatrix} \mu_0(k) & 0 & \dots & 0 \\ 0 & \mu_1(k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mu_L(k) \end{pmatrix} \tag{1.27}$$

Moreover, the elements of the matrix $\mathbf{M_k}$ are allowed to vary with time according to a certain algorithm. The weight update equation is:

$$\mathbf{W_{k+1}} = \mathbf{W_k} + 2e_k\mathbf{M_k}\mathbf{X_k} \tag{1.28}$$

The authors claim that only about a 15 percent increase in computational complexity is entailed.

Yet another technique exists, which performs the filtering operation and the coefficient update in the frequency domain using the power of the FFT (Fast Fourier Transform). Moreover, the transformation decorrelates the coefficients, thus improving the convergence properties. Considerable computational savings are reported possible [26] for filter lengths greater than 16. A drawback of this approach is the block nature of the algorithm (because of the FFT's), which may not be appropriate for all applications. Also the hardware implementation could be much more complicated than the time-domain techniques.

The most powerful method, developed to date for transversal filter adaptation, is the *recursive least-squares (Kalman)* algorithm. This algorithm tries to minimize the sum of the exponentially weighted error samples, going back in time, i.e.:

$$J_n = \sum_{k=0}^{n} \alpha^{n-k} |d_k - \mathbf{X_k^T W_k}|^2 \tag{1.29}$$

for $0 < \alpha < 1$. The performance function J_n is based on time averages (unlike the LMS algorithm which is based on ensemble averages). The coefficients are updated according to the following formula:

$$\mathbf{W_{k+1}} = \mathbf{W_k} + e_k\mathbf{K_k} \tag{1.30}$$

where $\mathbf{K}_k$ is the Kalman gain vector, which must be updated in a continuous manner.

In terms of adaptation speed and accuracy the Kalman algorithm is much superior to all other approaches. But the price to pay is a considerable increase in complexity and computational burden. Even the fast version of the algorithm, introduced by D. Falconer and L. Ljung [33], cannot overcome this problem. Another disadvantage is the reported instability of the fast Kalman algorithm after a large number of iterations, especially while working with finite precision arithmetic [12].

Another class of adaptive algorithms are based on the *lattice structure*. The algorithms can be gradient or recursive least-squares. The advantages over the transversal architecture are as in the non-adaptive case — better numerical properties and a relatively easy way to expand the filter order. Because of the orthogonalizing properties of the lattice structure the speed of convergence is also improved. Unfortunately the computational complexity becomes greater.

For the sake of completeness IIR (*Infinite Impulse Response*) adaptive algorithms should be mentioned. Their use is still limited because the analysis is quite complicated and they are prone to numerical instabilities. The main advantage consists in the possibility of modeling transfer functions with both poles and zeros and also of possible computational and memory savings due to reduced filter length, which is characteristic of IIR systems in general.

There are also many other modifications of existing algorithms, like the one using delta-modulation techniques [1], or block least-squares [27]. It is not possible and not necessary to consider them all.

To summarize the discussion it can be pointed out that algorithms more powerful than the LMS exist but they are more complicated and sometimes not very well understood. As mentioned before the focus of this work is concentrated on high-speed adaptive filtering, where simplicity of implementation and reliability are great advantages. For that reason further investigation of the properties of the LMS algorithm will be done in search of possible improvements of its performance.

CHAPTER II

LMS ALGORITHM WITH FILTERING OF THE ERROR SIGNAL (FLMS)

2.1 The Idea of Error Filtering

In chapter one it is pointed out that the LMS algorithm uses an unbiased but noisy estimate of the gradient (see equation 1.18). Naturally the question arises whether it is possible to do something to improve the estimate. For convenience the expression for the gradient is given here again:

$$\nabla_k = -2E[e_k \mathbf{X_k}] \tag{2.1}$$

It is important to note the statistical expectation involved, i.e. the need for ensemble averaging. In the steepest descent algorithm the gradient is estimated by using small perturbations of the weight vector and averaging the gradient estimate over all samples between two iterations. But the perturbations contribute to an excess error. Thus the algorithm of steepest descent does not provide better results than the LMS algorithm (or more precisely it is proven to be inferior in performance).

Another consideration in choosing (for our goals) a suitable manner of improving the performance of the LMS algorithm is simplicity. That is the reason why relatively complicated methods like the orthogonalized LMS algorithm or the variable step approach, mentioned in chapter one, are put aside. However, the overly simplified methods for coefficient update using sign-bit multiplications are shown [29] to severely degrade the convergence rate and will not be considered here.

Taking into account all these considerations, the idea came to investigate the effect of using a filtered value of the error signal. The filtering operation itself should be very simple in order not to introduce additional computational complexity and delay into the update procedure. A good candidate is a simple exponential averaging

in time i.e.:

$$\tilde{e}_k = \sum_{i=0}^{k} \alpha^{i-k} e_i = \sum_{i=0}^{k} a^{k-i} e_i \qquad (2.2)$$

Obviously $a = 1/\alpha$.

Intuitively there is a feeling that the information about past values of the error signal could be useful for improving the convergence of the algorithm.

Such processing of the error signal is reported [18] to be useful in IIR adaptation algorithms. There it is done mainly to guarantee the stability of the algorithm (the derivation is based on the principle of hyperstability). Simulation results also show slight convergence speed improvement.

Duttweiler [29] has investigated the effects of nonlinear operations on the coefficient update term $2\mu e_k \mathbf{X_k}$, i.e. the situation where e_k is replaced by a function (possibly nonlinear) $f(e_k)$ and $\mathbf{X_k}$ is replaced by $g(\mathbf{X_k})$. He proves that linear functions give optimum results. However, the possibility of linear filtering of the update components is not addressed.

2.2 Linear Systems Approach to the FLMS algorithm

Trying to explain the effects of error signal filtering on the LMS algorithm is not a simple task. It must be made clear from the start that the introduction of the filtering operation does *not* change the performance criterion used in conventional LMS algorithms — the minimization of the mean-square error. For that reason the FLMS algorithm will be analyzed using the now classical and well developed theory of LMS adaptation but inserting modifications where necessary.

The exponential averaging operation in (2.2) can be thought of as a first-order IIR linear filter shown in Fig.2.1.

The impulse response of the filter is:

$$h(n) = a^n u(n) \qquad \text{for} \quad n = 0, ..., \infty \qquad (2.3)$$

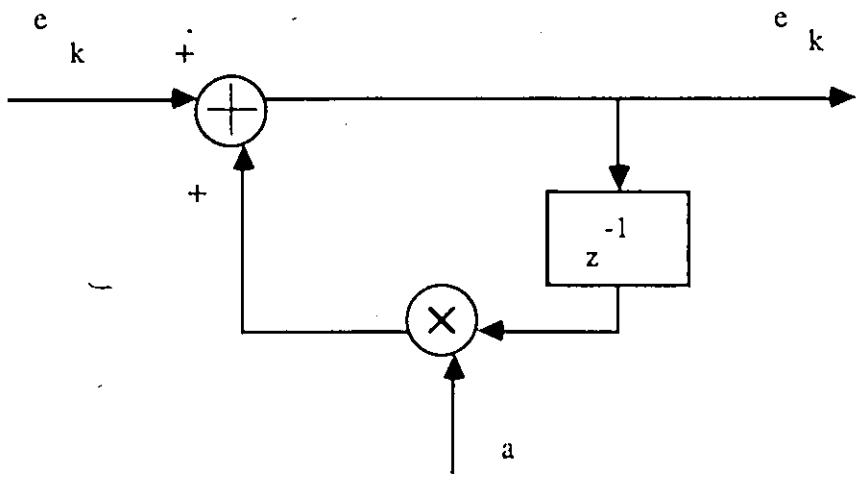


Fig.2.1 Error filtering element

where $u(n)$ is the unit step function. The transfer function is given by the equation:

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} = \sum_{n=0}^{\infty} (az^{-1})^n \tag{2.4}$$

To guarantee stability and convergence of the sum in (2.4) the following condition must be met:

$$|az^{-1}| < 1 \qquad \text{or} \qquad |z| > |a| \tag{2.5}$$

i.e. z must take values outside a circle of radius $|a|$ in the z-plane. Then (2.4) has a closed-form representation:

$$H(z) = \frac{1}{1 - az^{-1}} = \frac{z}{z - a} \tag{2.6}$$

For real signals it is necessary to include the unit circle in the convergence zone, which imposes limits on the possible values of a :

$$|a| < 1 \tag{2.7}$$

The frequency response can be written as:

$$|H(e^{j\omega})| = |\frac{e^{j\omega}}{e^{j\omega} - a}| = \frac{1}{\sqrt{1 - 2a\cos\omega + a^2}} \tag{2.8}$$

where $\omega = 2\pi f/f_s$ and f_s is the sampling frequency. The phase response is given by:

$$\varphi(\omega) = \tan^{-1}\frac{Im[H(e^{j\omega})]}{Re[H(e^{j\omega})]} = -\tan^{-1}\left[\frac{a\sin\omega}{1 - a\cos\omega}\right] \tag{2.9}$$

The Nyquist interval of frequencies is $0 \leq f \leq f_s/2$ so that $0 \leq \omega \leq \pi$. For $a > 0$ the resulting filter is lowpass, for $a < 0$ the filter is highpass. The frequency response and the phase response are shown in Fig.2.2 and Fig.2.3 respectively for three different values of a.

The only modification required to implement the FLMS algorithm is to replace the error e_k with the filtered error $\tilde{e}_k$ in the coefficient update procedure:

$$\mathbf{W_{k+1}} = \mathbf{W_k} - 2\mu\tilde{e}_k\mathbf{X_k} \tag{2.10}$$

As can be seen from Fig.2.1 and equation (2.4), the filtering operation does not introduce delay into the update procedure (actually in hardware implementation there will be initially one clock cycle delay for the summation but this is not significant). Also the additional computations are negligible compared to the rest of the algorithm. Values of α equal to powers of two are particularly attractive for hardware implementation because the multiplication can be replaced by shifting. Further considerations on how to choose a (or α) will be discussed in the following sections of this chapter.

2.3 Effects of Error Filtering on the Speed of Convergence

In order to be more specific in our discussions, a particular model of the adaptive filtering problem is assumed. The input signal x_k to the filter is composed of a sine wave and white gaussian noise with variance depending on the chosen signal-to-noise ratio. The desired output d_k is a pure sine wave with the same frequency.

When trying to minimize the mean-square error $\mathrm{E}[e_k^2]$ the adaptive algorithm attempts to predict and cancel the sine wave from the desired output and to suppress the noise contained in the input. So it will eventually converge to a narrow passband filter centered around the frequency of the sine wave. During adaptation it is the error resulting from the uncanceled sine wave that drives the algorithm towards the correct solution. The noise disturbs the adaptation process and creates statistical fluctuations. The filtering of the error signal will eventually have the effect of virtually increasing the adaptation constant μ for a specific frequency. In the steady-state, however, when the sinusoidal signal is almost cancelled out, the modification is expected not to affect the performance, i.e. not to increase the residual mean-square error. Whether these intuitive conclusions are correct or not remains to be seen.

From the observation of the error filter's frequency response (Fig.2.2) it is clear that for bigger absolute values of the coefficient a the lowpass (highpass) characteristics are more pronounced and a greater gain in convergence speed could be achieved for given frequency ω_i if $|H(e^{j\omega_i})| >> 1$. At the same time the area below the curve of the frequency response becomes greater with increasing values of $|a|$, which means that the power of the noise component also increases. A well known expression for the variance of a white-noise signal processed by a linear system is:

$$\sigma_y^2 = \sigma_x^2 \sum_{n=0}^{\infty} h^2(n) \tag{2.11}$$

where σ_x^2 is the variance of the input white noise, σ_y^2 is the variance of the processed noise (which can no longer be white) and $h(n)$ is the impulse response of the linear system.

In our specific case:

$$\sum_{n=0}^{\infty} h^2(n) = \frac{1}{1-a^2} \tag{2.12}$$

Improvement in convergence speed is expected to be present as long as more emphasis is put on the power of the deterministic signal than on the power of the noise. Using that fact and equations (2.12) and (2.8) the following relation can be established:

$$\left(\frac{1}{\sqrt{1-2a\cos\omega+a^2}}\right)^2 \geq \frac{1}{1-a^2} \tag{2.13}$$

or, since $(1-a^2)$ is always greater than 0:

$$\frac{1-a^2}{1-2a\cos\omega+a^2} \geq 1 \tag{2.14}$$

Solving that inequality for a gives:

$$\boxed{|a| \leq |\cos\omega|} \tag{2.15}$$

From the left side of inequality (2.14) a solution for the optimum value of a, maximizing this expression, can be found. Taking the derivative with respect to a and setting it equal to zero gives:

$$\frac{\partial}{\partial a}\left(\frac{1-a^2}{1-2a\cos\omega+a^2}\right) = 0 \tag{2.16}$$

After solving the equation (which involves rather elementary mathematics that will not be shown here) the following result for a is obtained:

$$\boxed{a_{opt} = \frac{\cos\omega}{1+\sin\omega}} \tag{2.17}$$

for $0 \leq \omega \leq \pi$

The value of a_{opt} can now be substituted in (2.14) to find the maximum achievable improvement in adaptation speed. The result is quite simple:

$$\boxed{G_{max} = \frac{1}{\sin\omega}} \tag{2.18}$$

for $0 < \omega < \pi$

From the above discussion it is clear that the criterion for choosing an appropriate value of a and the resulting performance improvement are frequency dependent. This is not desirable in general, because it limits the usefulness of the algorithm to a rather restricted number of applications. However, in some cases it can be used with success. For example in cancelling harmonic interference with a known frequency from a broadband signal, or retrieving a narrowband signal from broadband noise. It is certainly necessary to have a priori information about the approximate frequency range of the narrowband signal.

The results of the above discussion are summarized in Fig.2.4 and Fig.2.5. In Fig.2.4 are plotted expressions (2.15) and (2.17) . The shaded region is where the performance of the FLMS algorithm is better than that of the LMS algorithm. Also the curve of optimum values is given. From Fig.2.4 can conveniently be seen what value of a is optimum for a given frequency, and what is the limit beyond which the performance is worse than for the conventional LMS algorithm. On the other hand, if a is fixed, then it can be determined for what range of frequencies this coefficient gives better performance.

Fig.2.5 is a plot of the maximum achievable improvement G_{max} versus frequency. It is clear that bigger improvement is possible for frequencies near the edges of the working zone, i.e. $\omega = 0$ and $\omega = \pi$. Of course there is a practical limit , because the product $G_{max}\mu$ must be less than the maximum permissible value of the adaptation constant μ_{max}, otherwise the algorithm will not be stable.

Extensive computer simulations have been carried out to support the theoretical results and to look for possible surprises. The noise process is modelled by a pseudorandom sequence with zero mean and unit variance. The distribution is gaussian and the process can be assumed to be white with sufficient confidence. Tests were performed which showed that those assumptions were correct. The noise signal is

multiplied by an appropriate constant to achieve the desired signal-to-noise ratio in the input x of the filter. The sinusoidal signal used is with unit amplitude. Most of the simulations are done for filter length $L = 16$ taps. In the figure comments FS means the number of samples per period of the sine wave and gives information about its frequency.

Fig.2.6 shows the evolution of the mean-square error in time for three different values of the coefficient a (respectively α). The input signal-to-noise ratio is 0 dB ; the relative frequency of the sine wave is $0.1f_s$; the adaptation constant $\mu = 0.00055$. All curves are obtained by averaging 100 runs of the algorithm, every time using a different random sequence and different initial phase of the sine wave.

From Fig.2.6 it can be seen that the algorithm converges most rapidly for $\alpha = 2$ (or $a = 0.5$). The coefficient $\alpha = 1.5$ ($a = 0.66$) gives worse results, although the lowpass characteristics are more pronounced. From formula (2.17) the optimum value of a for frequency $0.1f_s$ is found to be 0.51 ($\alpha = 1.96$). For the maximum improvement formula (2.18) gives $G_{max} = 1.7$. The upper limit for a is found from (2.15) to be $a = 0.81(\alpha = 1.23)$. Fig.2.7 shows that there is almost no improvement for $\alpha = 1.25$ which is in accord with the theory.

Another observation that can be made from the graphs is the absence of visible difference between the three cases in terms of residual mean-square-error after convergence is completed for all of them. This fact comes to confirm the expectations, but needs more careful investigation, which will be done later in this chapter.

Fig.2.8 shows a different situation. Here the frequency of the sine wave is higher ($0.33f_s$) and the value $a = 0.5$ clearly gives worse performance than in the case without filtering. In this case $a_{opt} = -0.26$ (or $\alpha_{opt} = -3.9$), i.e. highpass filtering. But $G_{max} = 1.14$ which is quite small. This is the reason why in Fig.2.9 the difference between the two curves is so small.

In Fig.2.10 the input SNR is higher (20 dB). The curves have the same behavior as before, except that they are smoother and the residual MSE is very small, as

expected.

Fig.2.11 shows the error signal e_k itself (not averaged) for both LMS and FLMS algorithms. The error decreases faster for FLMS but after convergence both errors are the same.

It is interesting to observe how the coefficients of the filter change with time. Initially they are all set to zero. The behavior of one coefficient is characteristic for all of them. The coefficient, chosen arbitrarily for observation, is w_6. Fig.2.12 shows the coefficient values for one statistical sample and Fig.2.13 shows the mean value (i.e. averaged over all 100 runs).

Two things deserve attention. First, it is seen that in the case of filtering the curve exhibits a kind of "overshoot". This is not necessarily bad. The curves have no special physical meaning, moreover they show only *one* of the coefficients, so the final result can still be better in terms of minimizing the mean-square-error.

The second important observation is that the coefficient seems to converge to different values for different a. This fact is quite difficult to explain exactly. Some light can be brought on this by referring to a publication by R. J. Keeler [17]. Keeler points out that some of the assumptions used to simplify the derivation of the LMS algorithm and its performance are clearly not applicable to a large number of real situations. Those fundamental assumptions, on which the so-called *Independence theory of convergence* [13] is based, are:

1. All successive input vectors $\mathbf{X_i}$, $\mathbf{X_j}$ for $i \neq j$ are independent over time.

2. The weight vector $\mathbf{W_k}$ is independent of the data vector $\mathbf{X_k}$ for all k.

3. After convergence is completed, the error signal e_k is uncorrelated with the input vector $\mathbf{X_k}$.

Clearly in the presence of a deterministic signal in the input data neither of these conditions is met. As a result the coefficient vector will converge to a solution different from the Wiener solution.

In order to simplify the analysis Keeler assumes that the coefficient vector is updated once for every L sampling periods, where L is the number of filter taps. Thus only one of the coefficients is affected. For the mean weight vector recursion he gives the expression:

$$E[\mathbf{W_{k+1}}] = (\mathbf{I} - \mu\mathbf{R})E[\mathbf{W_k}] + \mu\mathbf{P} - \mu^2\sigma_n^2{\mathbf{S_k}}^T\mathbf{S_{k-L}}[0\ldots1]^T \tag{2.19}$$

where $\mathbf{S_k}$ is the deterministic signal vector and σ_n^2 is the variance of the noise. From here an expression for the steady-state expected weight vector can be obtained:

$$\mathbf{W}_\infty = \lim_{k\to\infty} E[\mathbf{W_k}] = \mathbf{R^{-1}P} - \mu\sigma_n^2\mathbf{R^{-1}S_k^T S_{k-L}}[0\ldots1]^T \tag{2.20}$$

But $\mathbf{R^{-1}P} = \mathbf{W^*}$ is the Wiener solution, so:

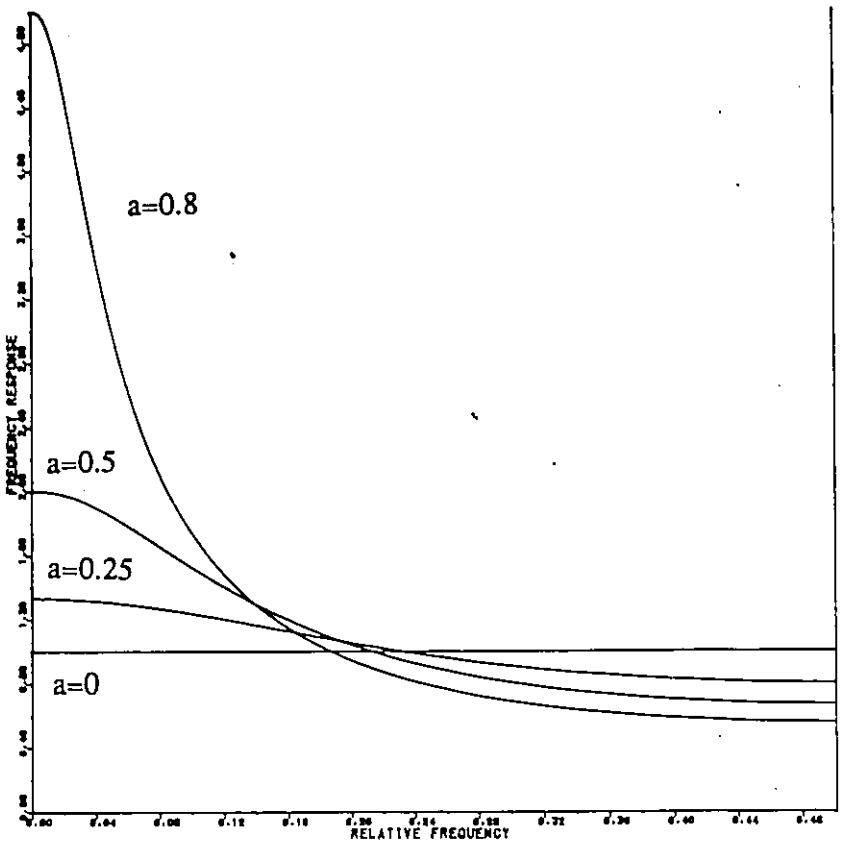
$$\mathbf{W}_\infty = \mathbf{W^*} - \mu\sigma_n^2\mathbf{R^{-1}S_k^T S_{k-L}}[0\ldots1]^T \tag{2.21}$$

Now the bias term is clearly seen in the right-hand side of equation (2.21). When updating at every sampling interval the expression will be much more complicated, involving more terms and affecting all coefficients. An exact analytical expression is extremely difficult, if not impossible, to find, but equation (2.21) gives enough insight into the problem, at least in qualitative sense.

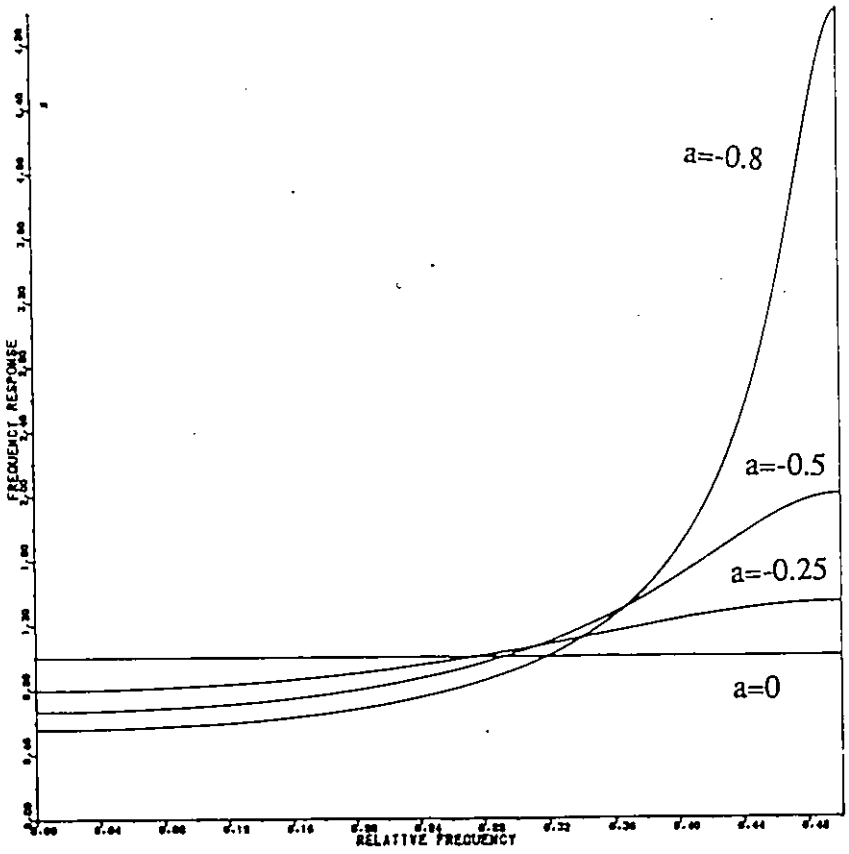
To account for the filtering of the error signal, σ_n^2 must be replaced by $\sigma_n^2/(1-a^2)$. That is where the additional bias comes from.

Equation (2.21) shows that when the noise power is small, the bias term can become insignificant. This is in accord with Fig.2.14; for SNR=20 dB in both cases the coefficient converges to the same value, which is in fact the optimum solution. Small values of μ also contribute to decrease the bias.

In Fig.2.15 the entire weight vector is plotted, i.e the impulse response of the filter, after convergence is completed and for three different values of α. The impulse response is sinusoidal with period equal to that of the input sine wave, as it should be. In the frequency domain the transfer function approximates a delta function, i.e a very narrow passband filter around the frequency of the input sinusoid.

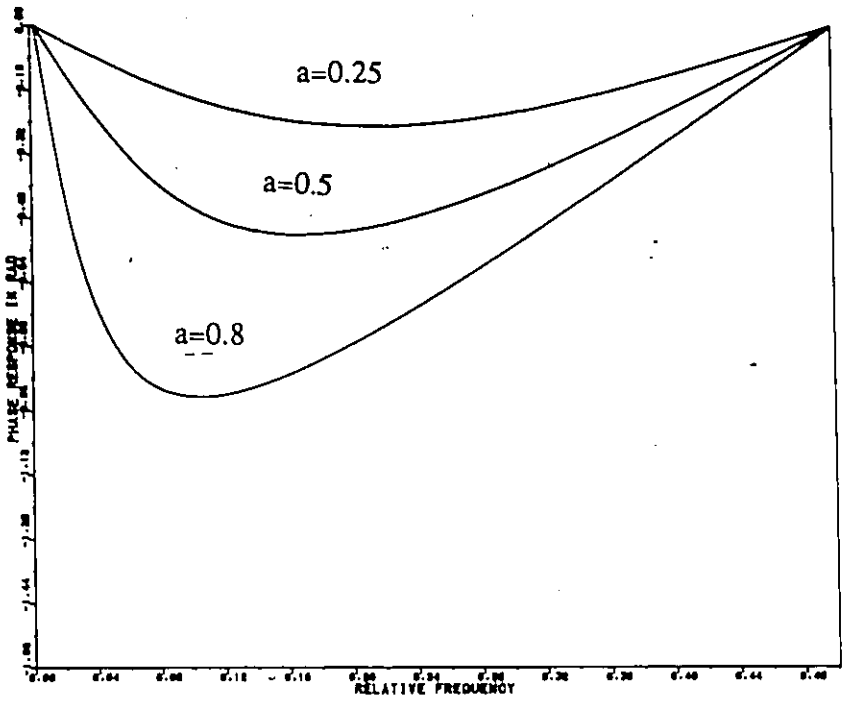


a)

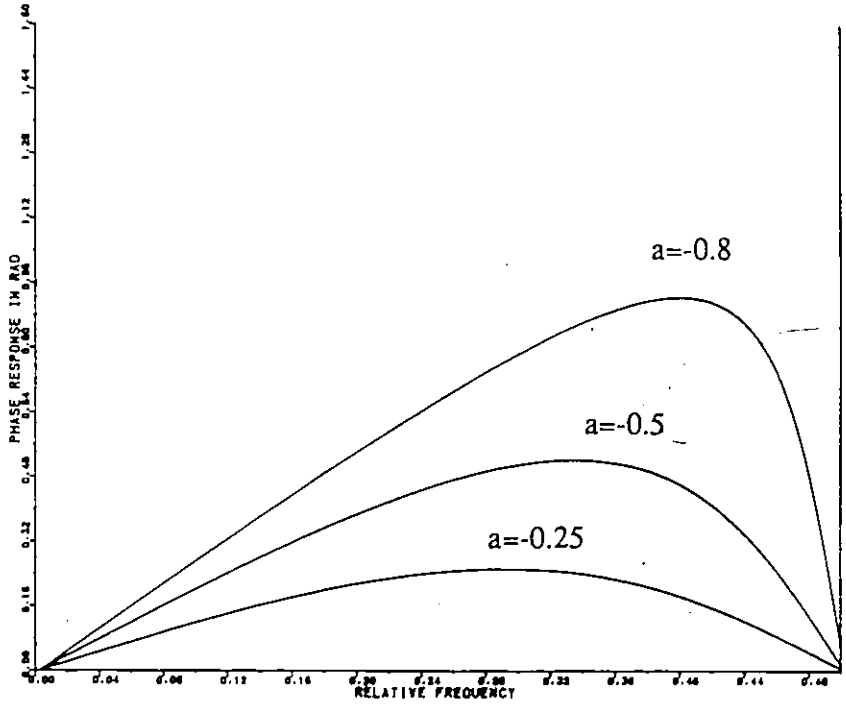


b)

Fig.2.2 Frequency response of the error filter, a) lowpass, b) highpass



a)



b)

Fig.2.3 Phase response of the error filter, a) lowpass, b) highpass

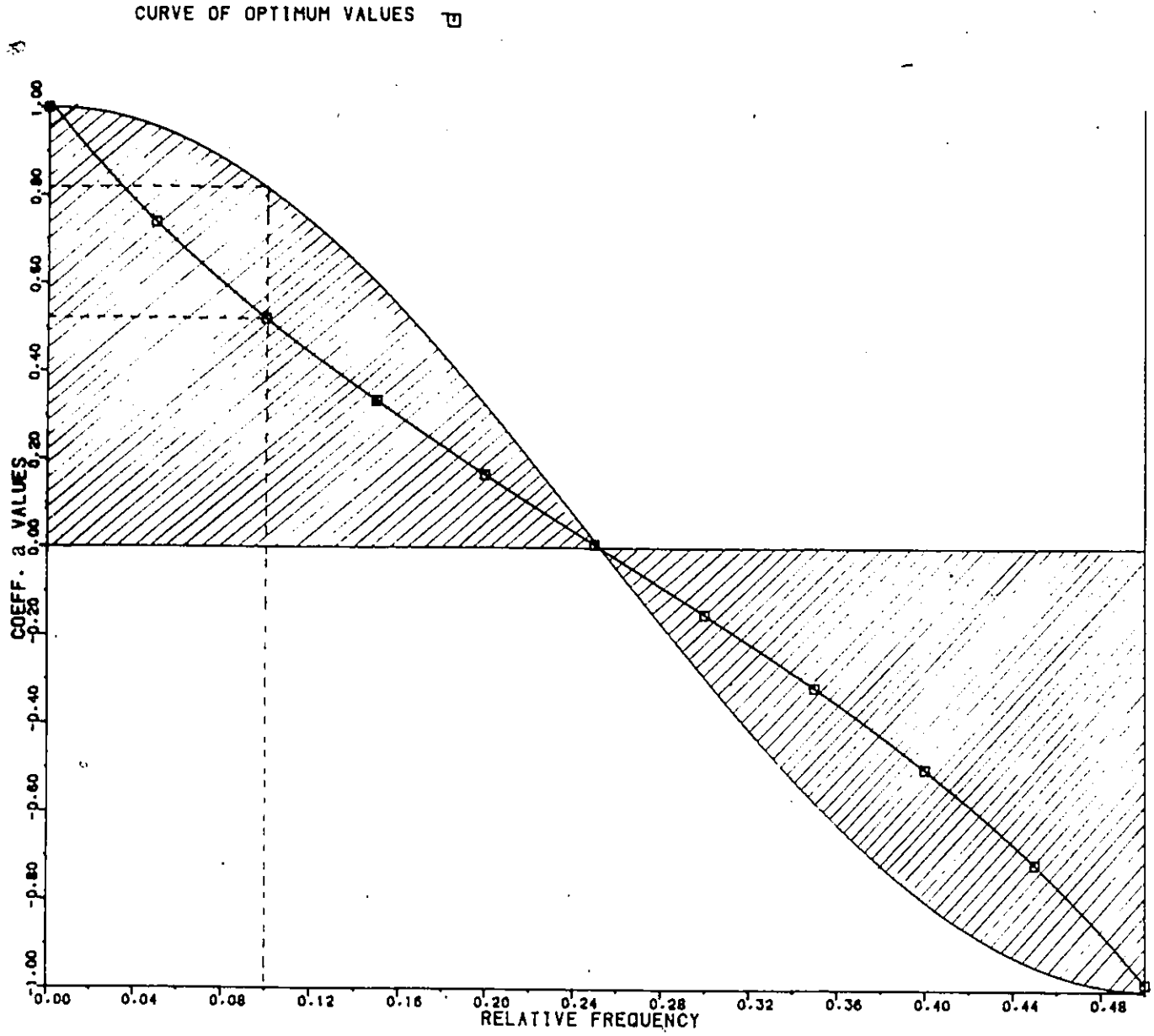


Fig.2.4 Optimum values of the coefficient a

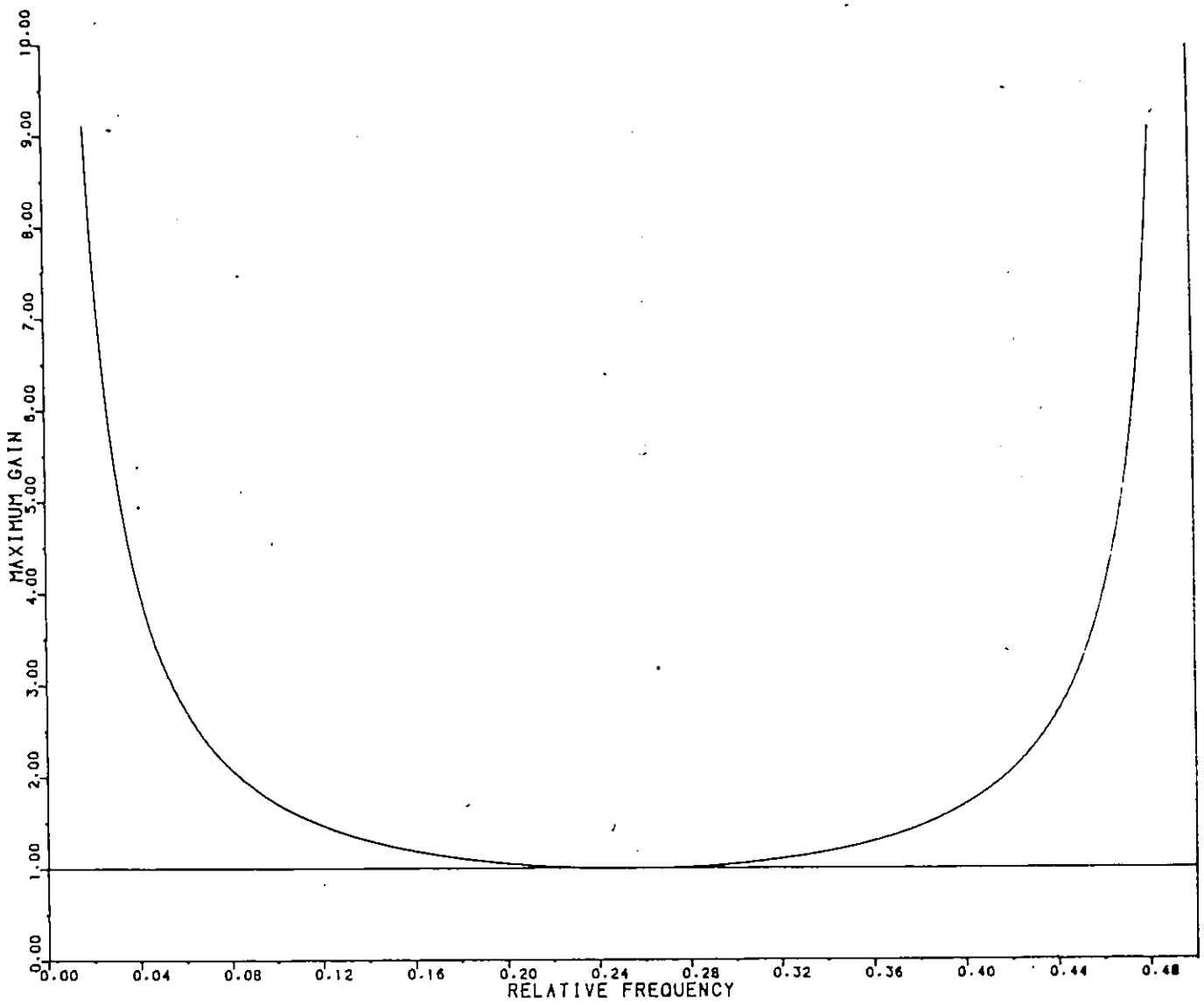


Fig.2.5 Maximum achievable gain, G_{max}

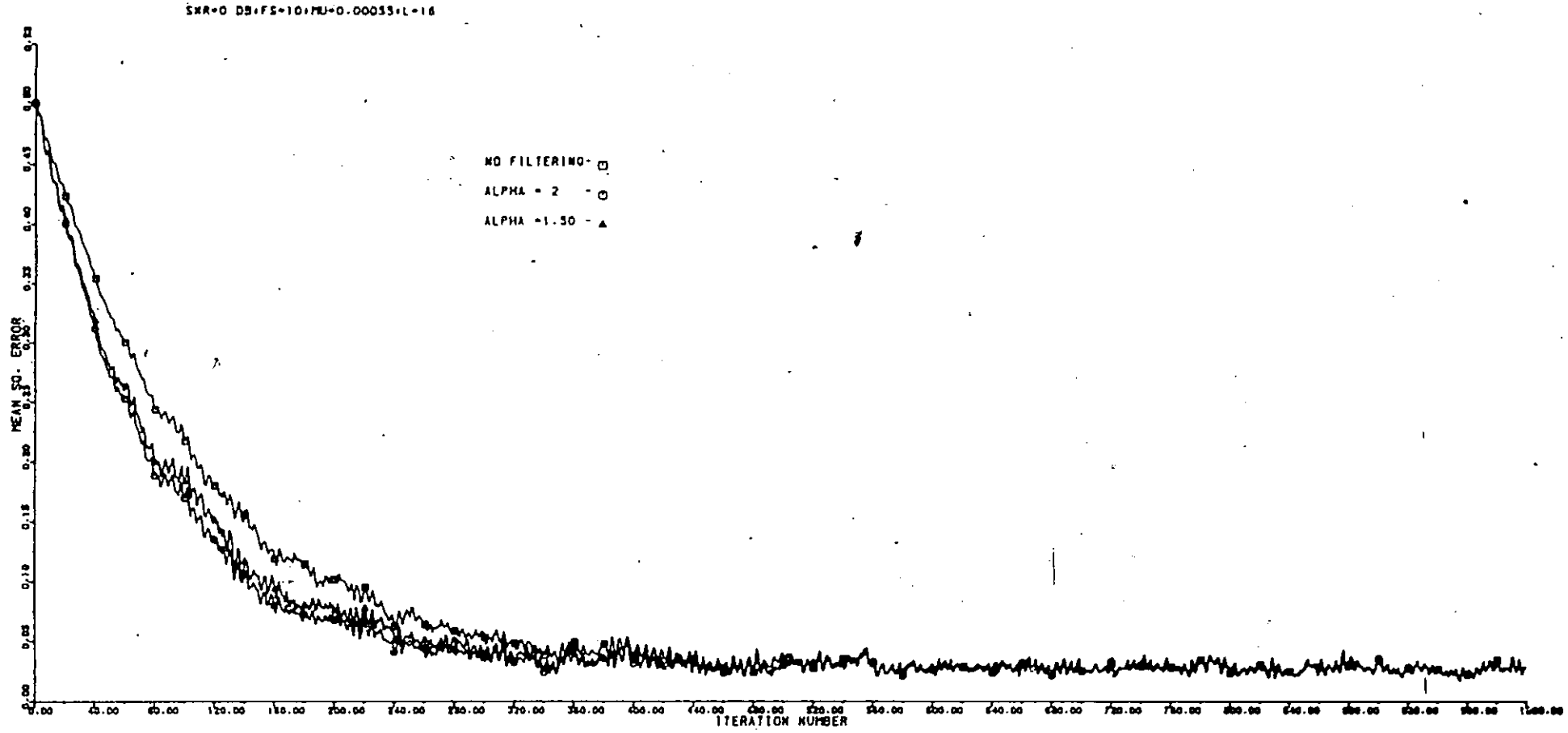


Fig.2.6 Learning curves for different values of α

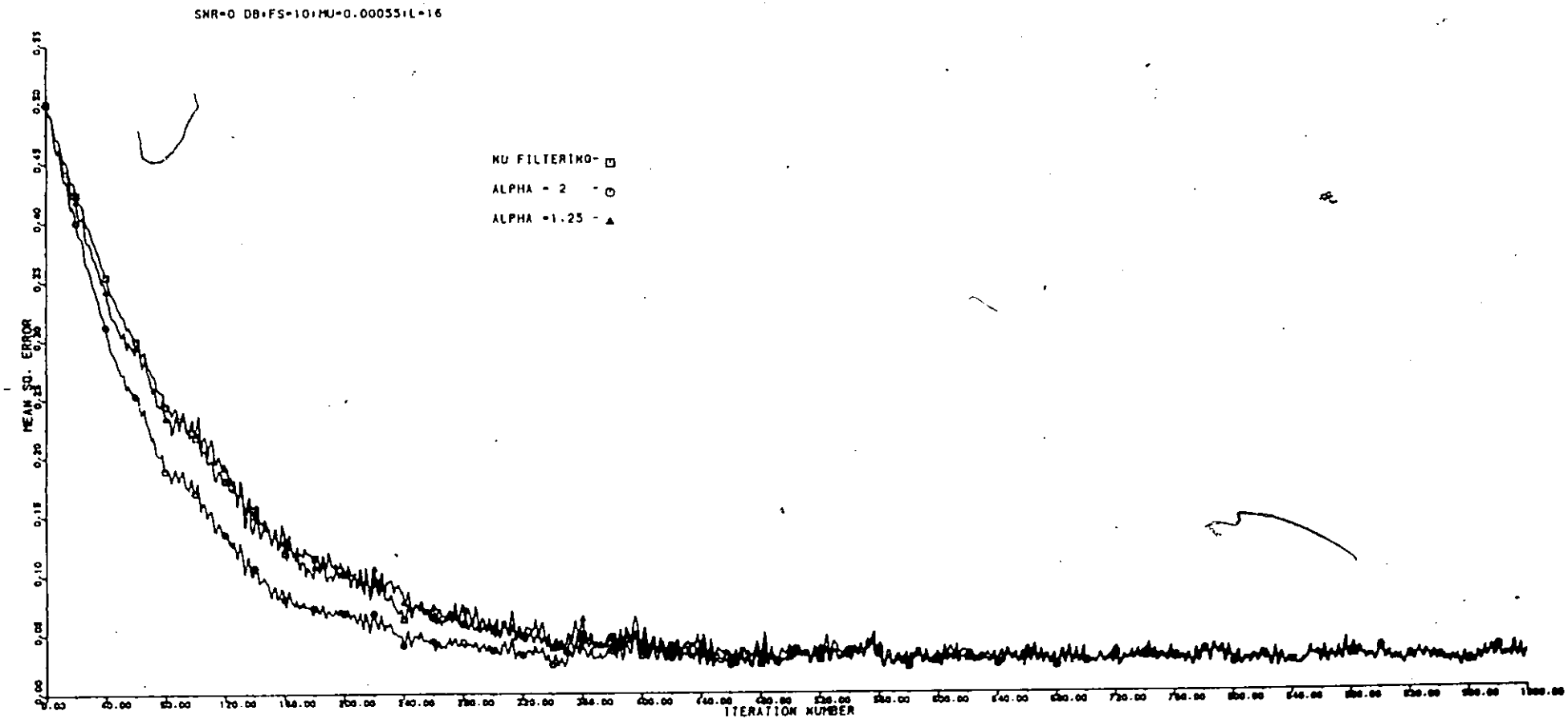


Fig.2.7 Learning curves

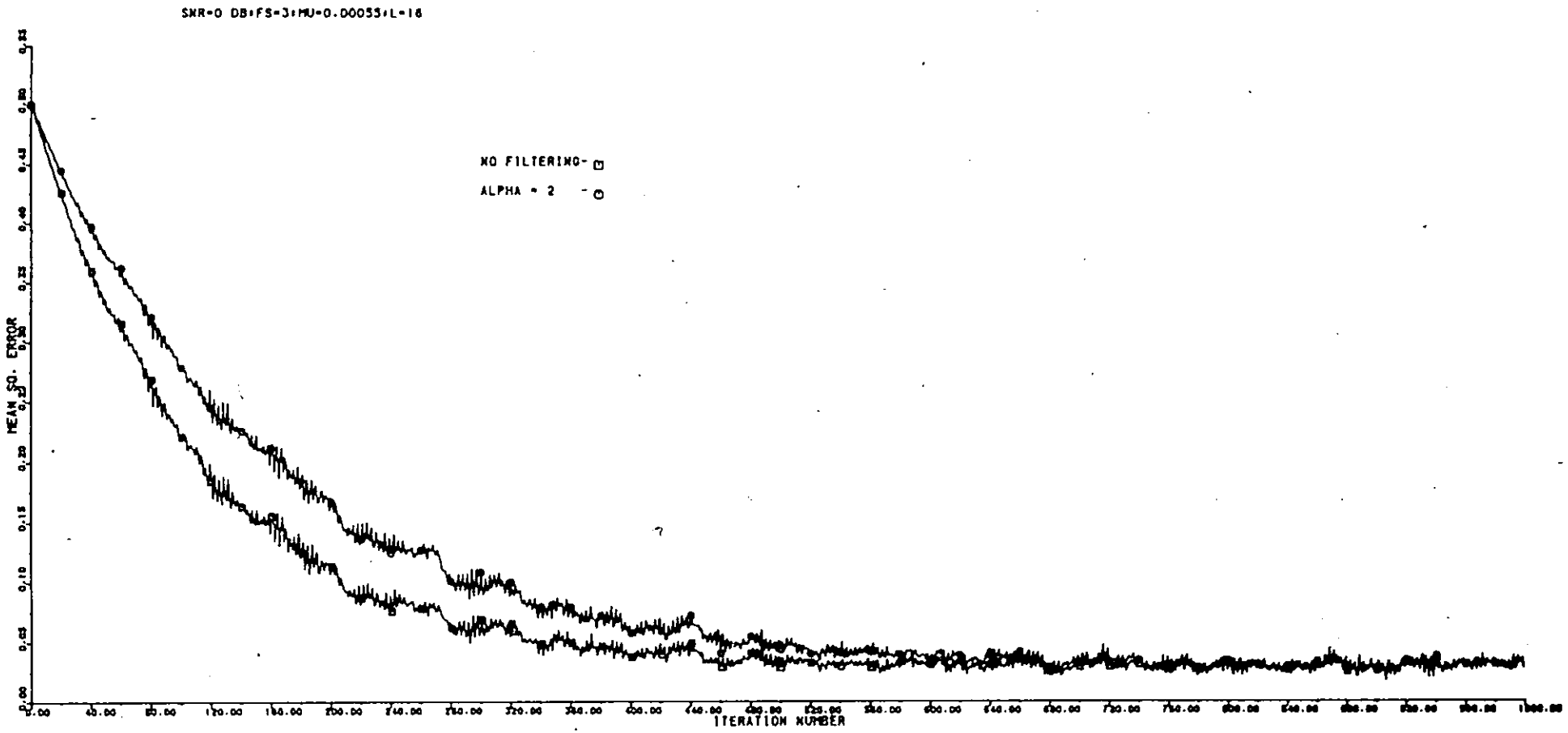


Fig.2.8 Learning curves for higher frequency, lowpass error filtering

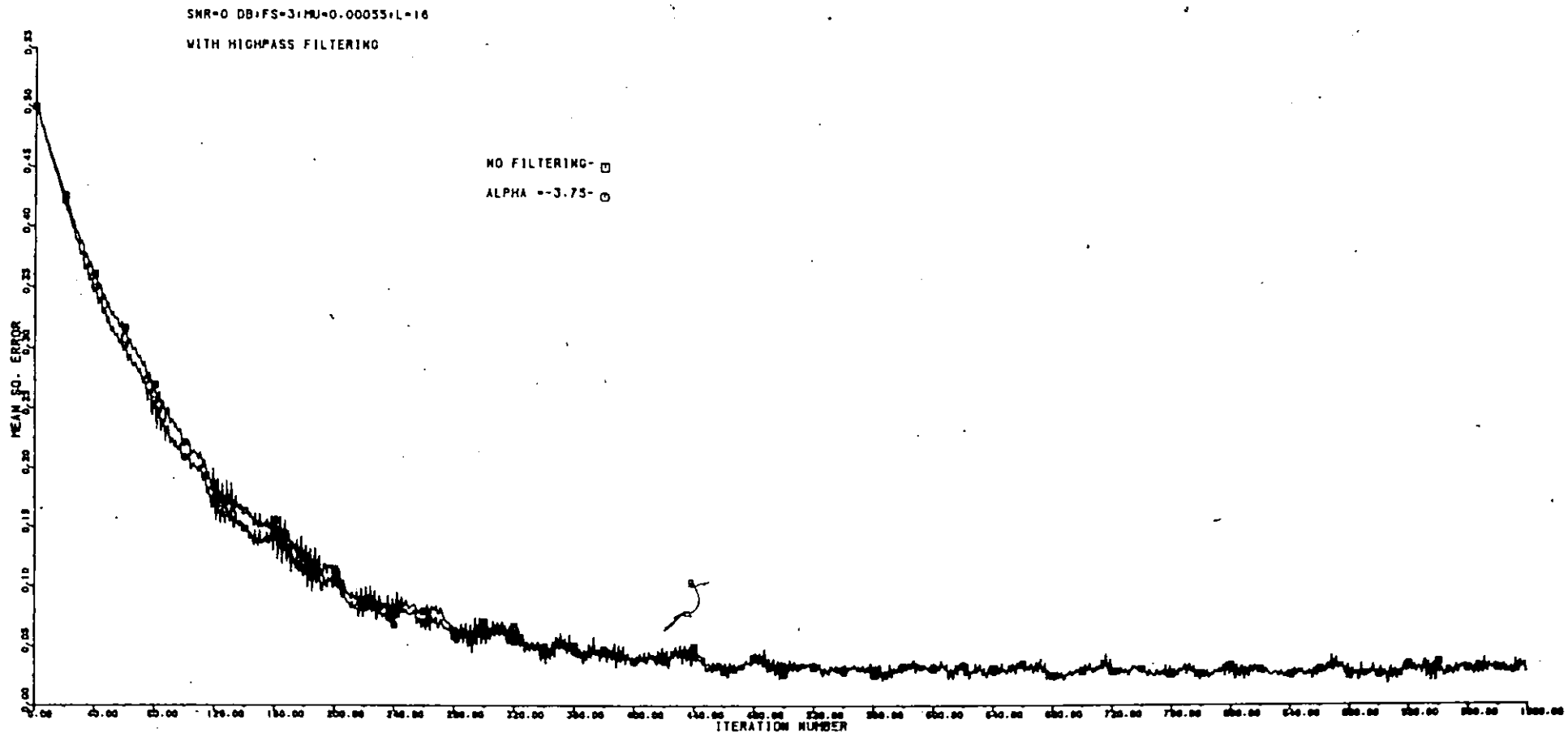


Fig.2.9 Learning curves for higher frequency and highpass error filtering

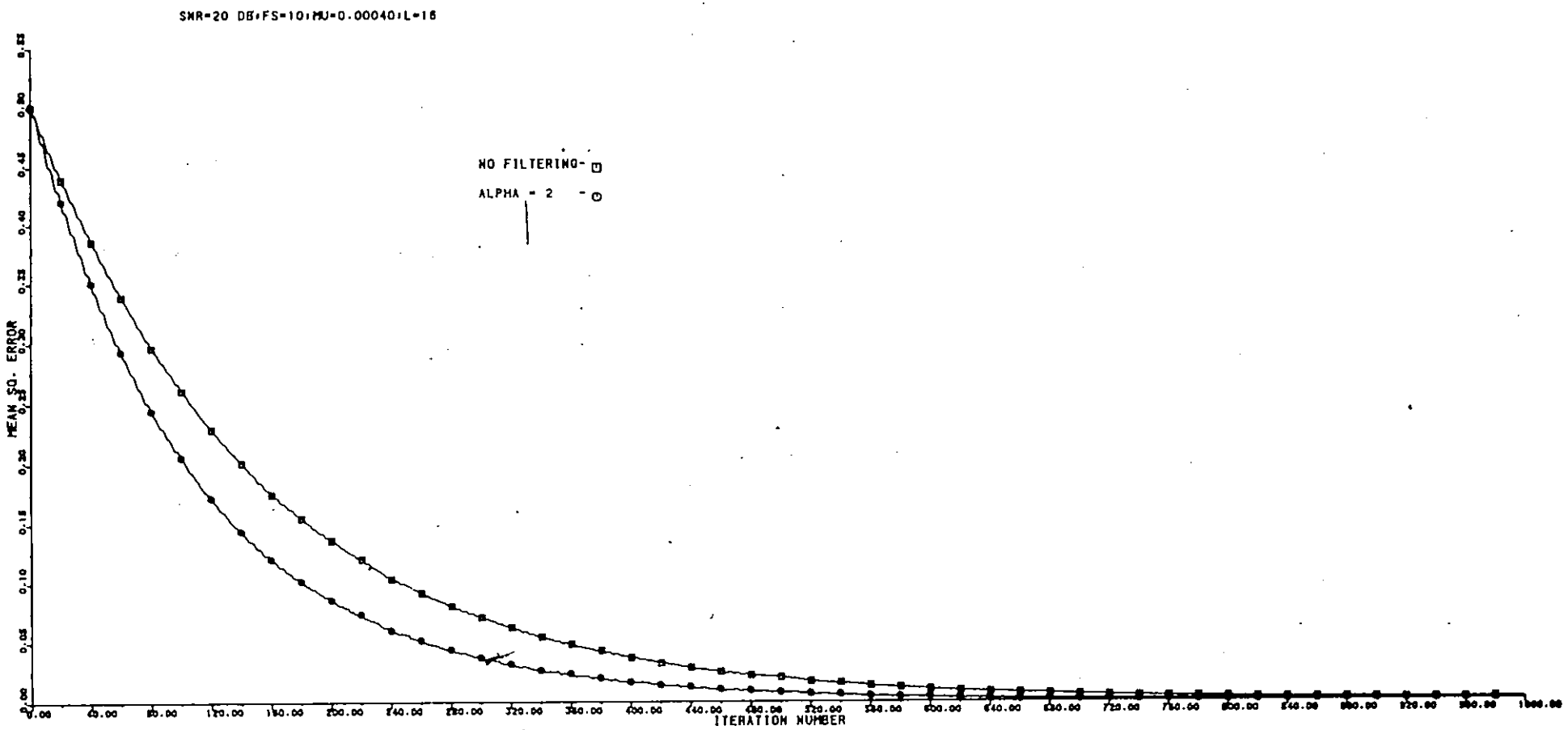


Fig.2.10 Learning curves for better input SNR

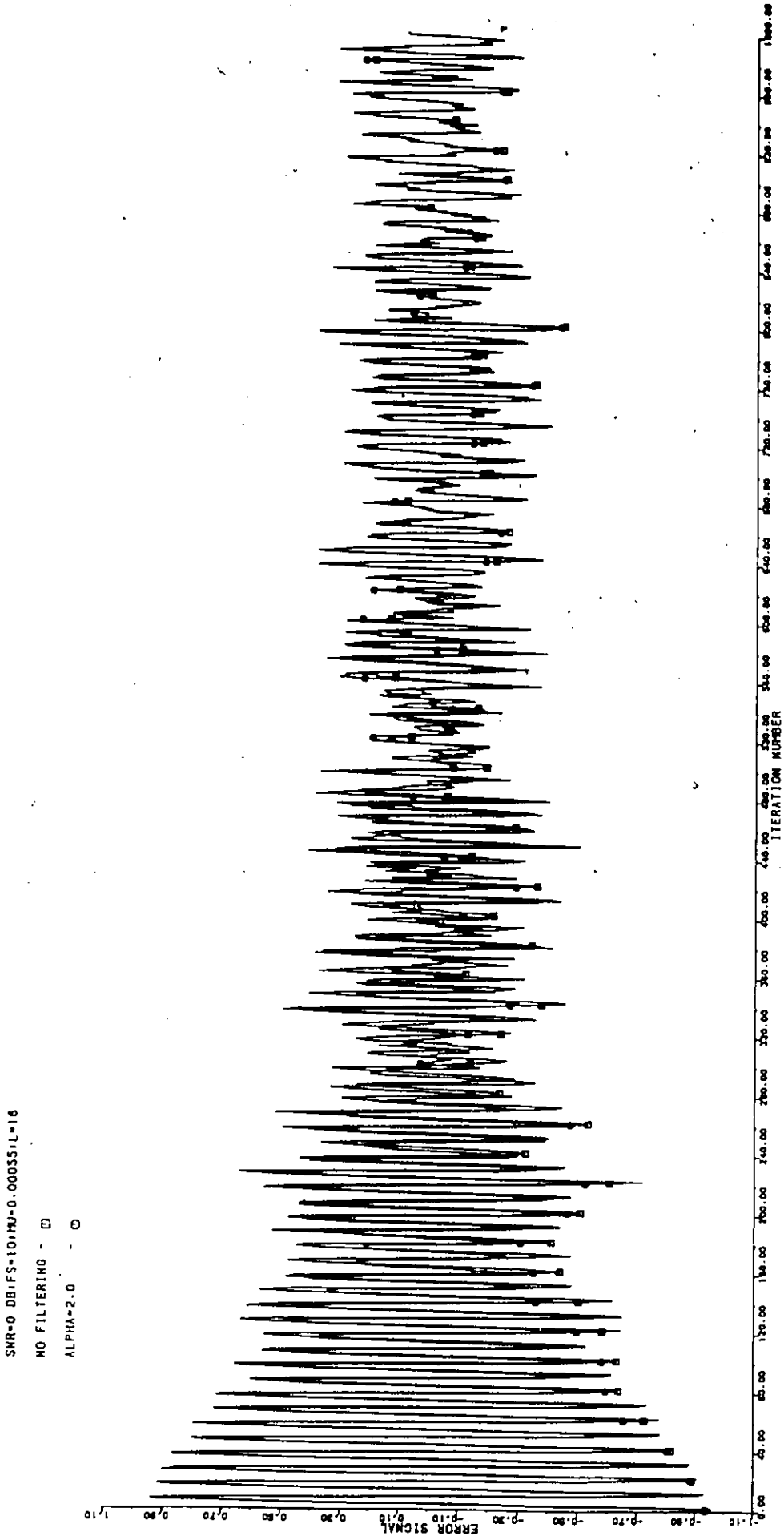


Fig.2.11 Comparison between the error signals in LMS and FLMS algorithms

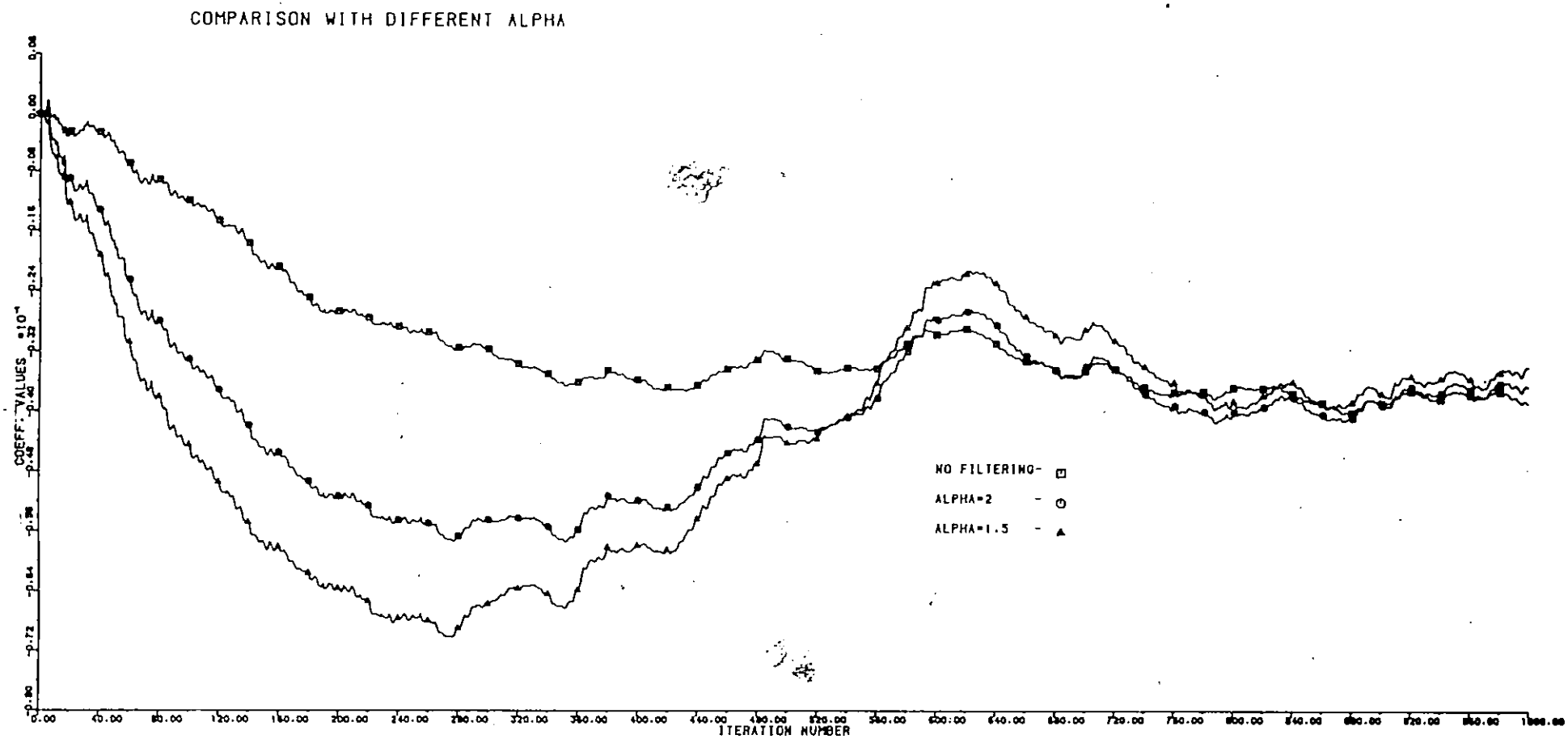


Fig.2.12 Coefficient behavior during adaptation -- individual run

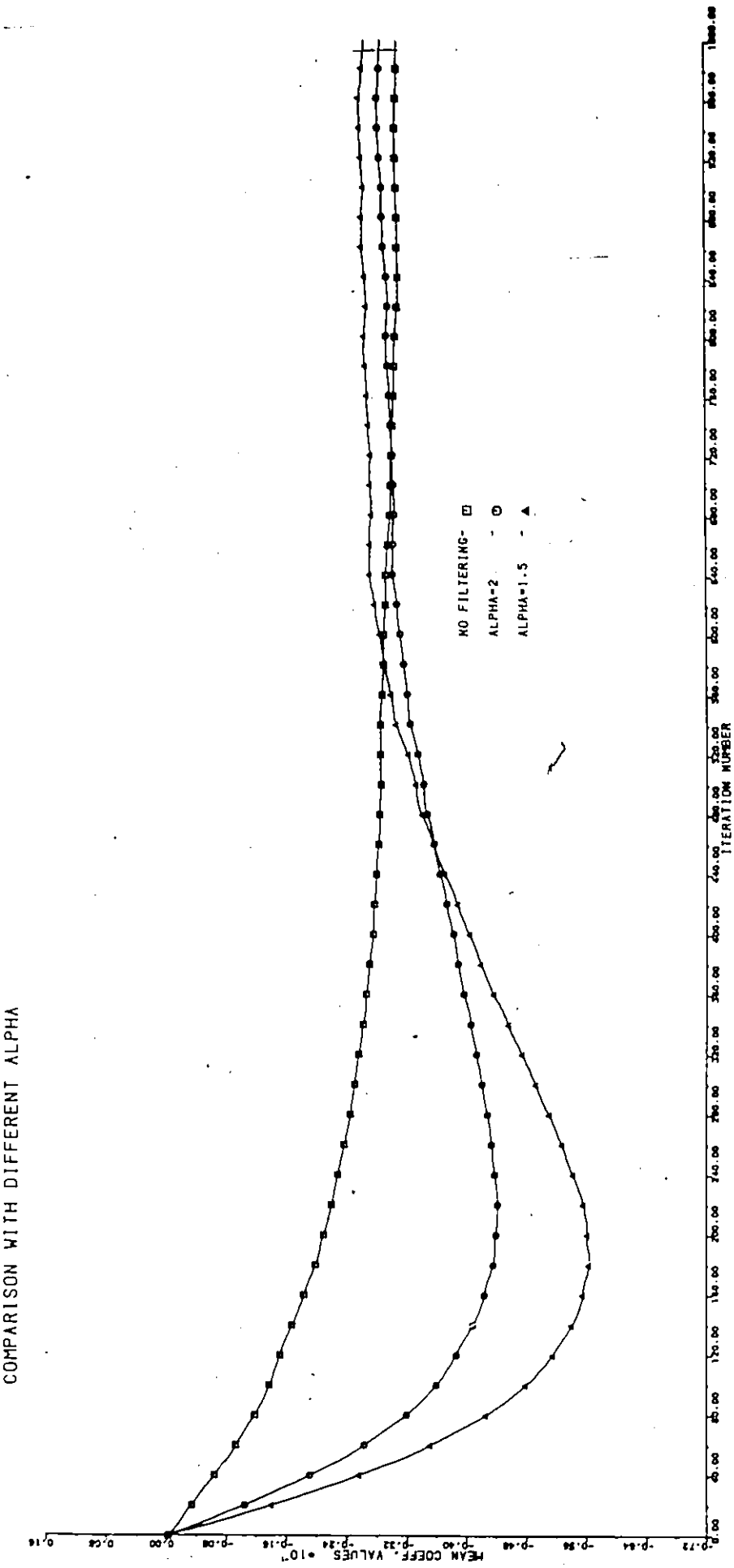


Fig.2.13 Behavior of the mean coefficient value during adaptation

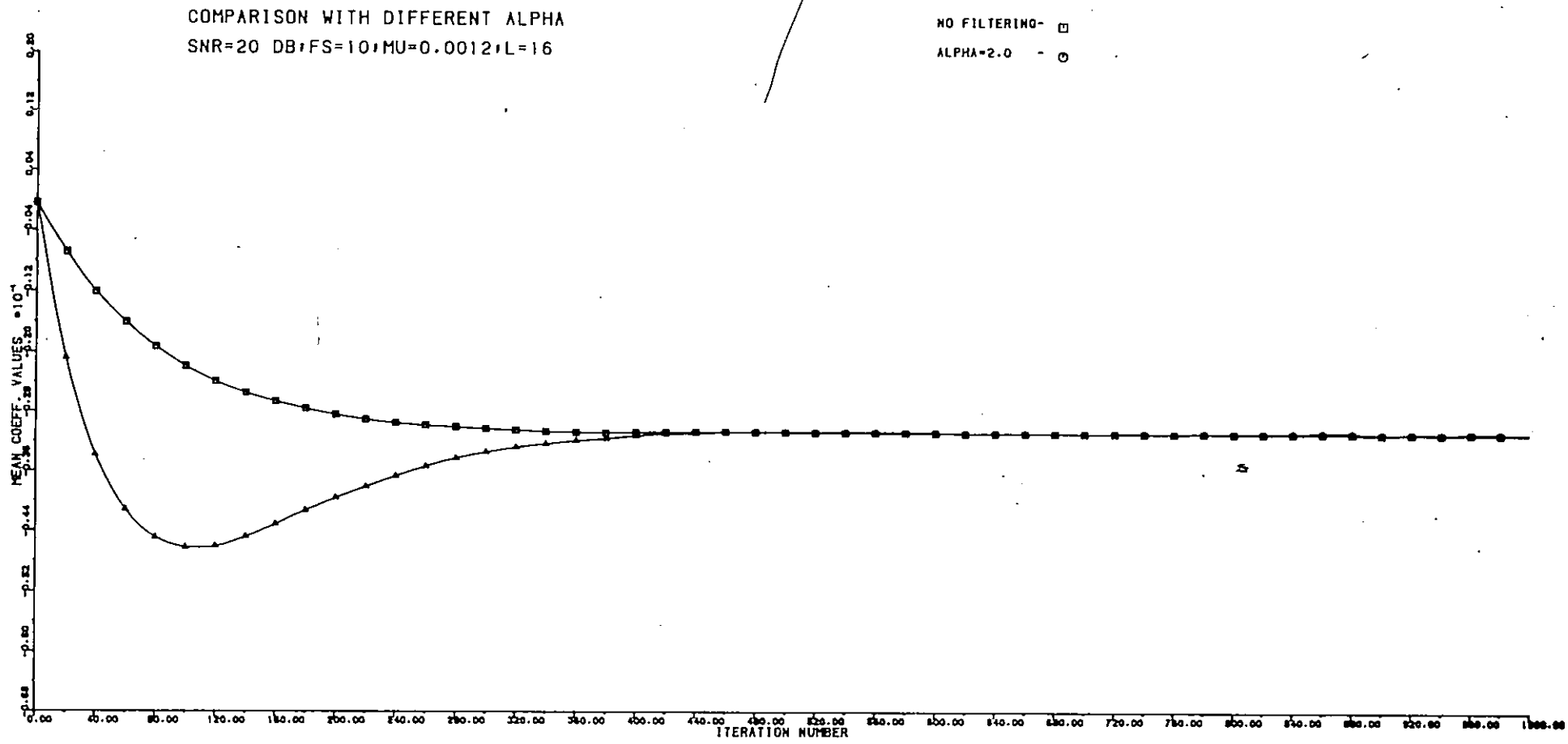


Fig.2.14 Mean coefficient values in the case of better input SNR

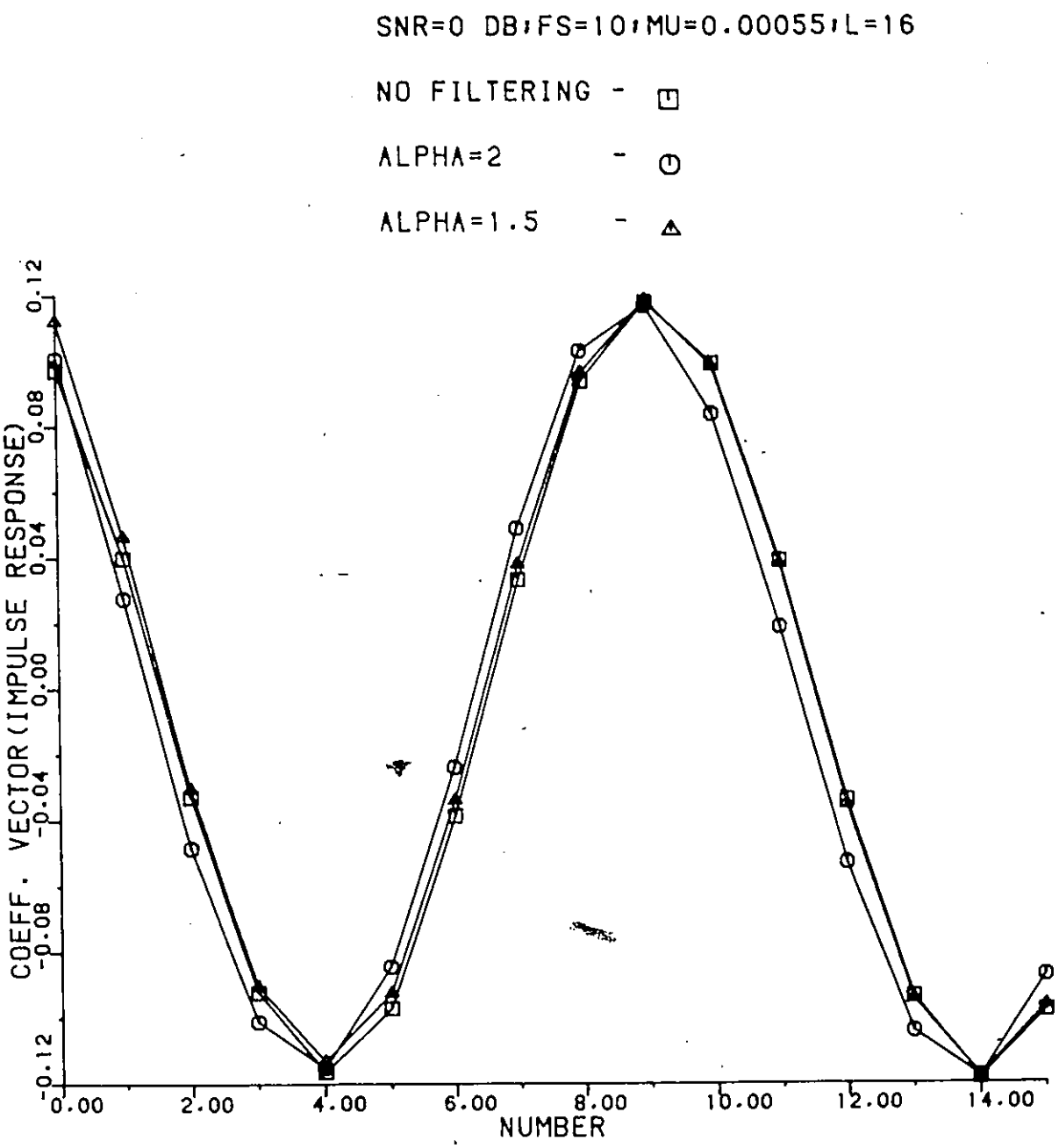


Fig.2.15 Impulse response of the adaptive filter in steady state

2.4 Error Filtering and Misadjustment

As was seen in the preceding section, the error filtering operation can provide improvement in convergence speed without increasing the misadjustment. This is of course desirable. If the error filtering causes bigger misadjustment, then it becomes useless, since the same result could be obtained by simply increasing the adaptation constant μ . However, as desirable as it is, this phenomenon is rather difficult to explain in the framework of conventional LMS algorithm analysis.

Let us define a vector $\mathbf{V_k}$ which is the difference between the optimum Wiener vector $\mathbf{W^*}$ and the actual weight vector $\mathbf{W_k}$:

$$\mathbf{V_k} = \mathbf{W_k} - \mathbf{W^*} \tag{2.22}$$

or for the steady-state:

$$\mathbf{V_\infty} = \mathbf{W_\infty} - \mathbf{W^*} \tag{2.23}$$

It can be proved that the excess mean-square error is given by:

$$\text{excess MSE} = \mathbf{V_\infty^T R V_\infty} \tag{2.24}$$

This equation indicates that any deviation from the optimum solution must eventually contribute to bigger misadjustment. In the preceding section it was seen how the error filtering process drives the coefficients further away from the optimum solution. Therefore the misadjustment should be greater.

Another way to come to the same conclusion is by considering the covariance of the gradient estimate. Let $\mathbf{N_k}$ denote the vector of noise components in the gradient estimate. Then:

$$\widehat{\nabla}_k = \nabla_k + \mathbf{N_k} \tag{2.25}$$

After convergence is completed $\nabla_k \approx 0$ and

$$\mathbf{N_k} \approx \widehat{\nabla}_k = -2e_k\mathbf{X_k} \tag{2.26}$$

Defined as in (2.25), $\mathbf{N_k}$ is zero mean, then:

$$\text{cov}[\mathbf{N_k}] = E[\mathbf{N_k}\mathbf{N_k^T}] = 4E[e_k^2\mathbf{X_k}\mathbf{X_k^T}] \tag{2.27}$$

Assuming e_k independent of $\mathbf{X_k}$, (2.27) becomes:

$$\text{cov}[\mathbf{N_k}] = 4E[e_k^2]\mathbf{R} \tag{2.28}$$

For the FLMS algorithm $E[e_k^2]$ is replaced by $\frac{1}{1-a^2}E[e_k^2]$, which increases the gradient noise covariance, which in turn will increase the misadjustment. Fig.2.16 shows the error signal before and after filtering. The filtered signal indeed has a bigger amplitude span and is slightly delayed in time, due to the negative phase response of the error filter.

Despite all theoretical indications, there is no sign of increased misadjustment in the simulation results. Fig.2.17 shows plots of three learning curves on a logarithmic scale. The logarithmic scaling emphasizes small differences but even this measure cannot distinguish between the curves after convergence is completed in all three cases. Moreover, Fig.2.18 and Fig.2.19 compare the autocorrelation functions and the power spectra respectively of the error signal for the LMS and the FLMS algorithms. The measurements are taken after a sufficient number of iterations (5000) to allow for complete convergence of both algorithms. The plots are seen to be almost identical.

It can be speculated that an explanation must be based on the nonvalidity of the independence assumptions given in section 2.3. Fig.2.18 shows the presence of relatively high amount of uncanceled sine wave in the error signal. It is clear then that the error signal e_k is correlated with the input vector $\mathbf{X_k}$ and the desired response d_k. Then the term $E[e_k^2\mathbf{X_k}\mathbf{X_k^T}]$ cannot be separated into a product of two expectations, as in equation (2.27). As mentioned before, consecutive input vectors are also highly correlated. Probably those complicated relationships are responsible for the results obtained.

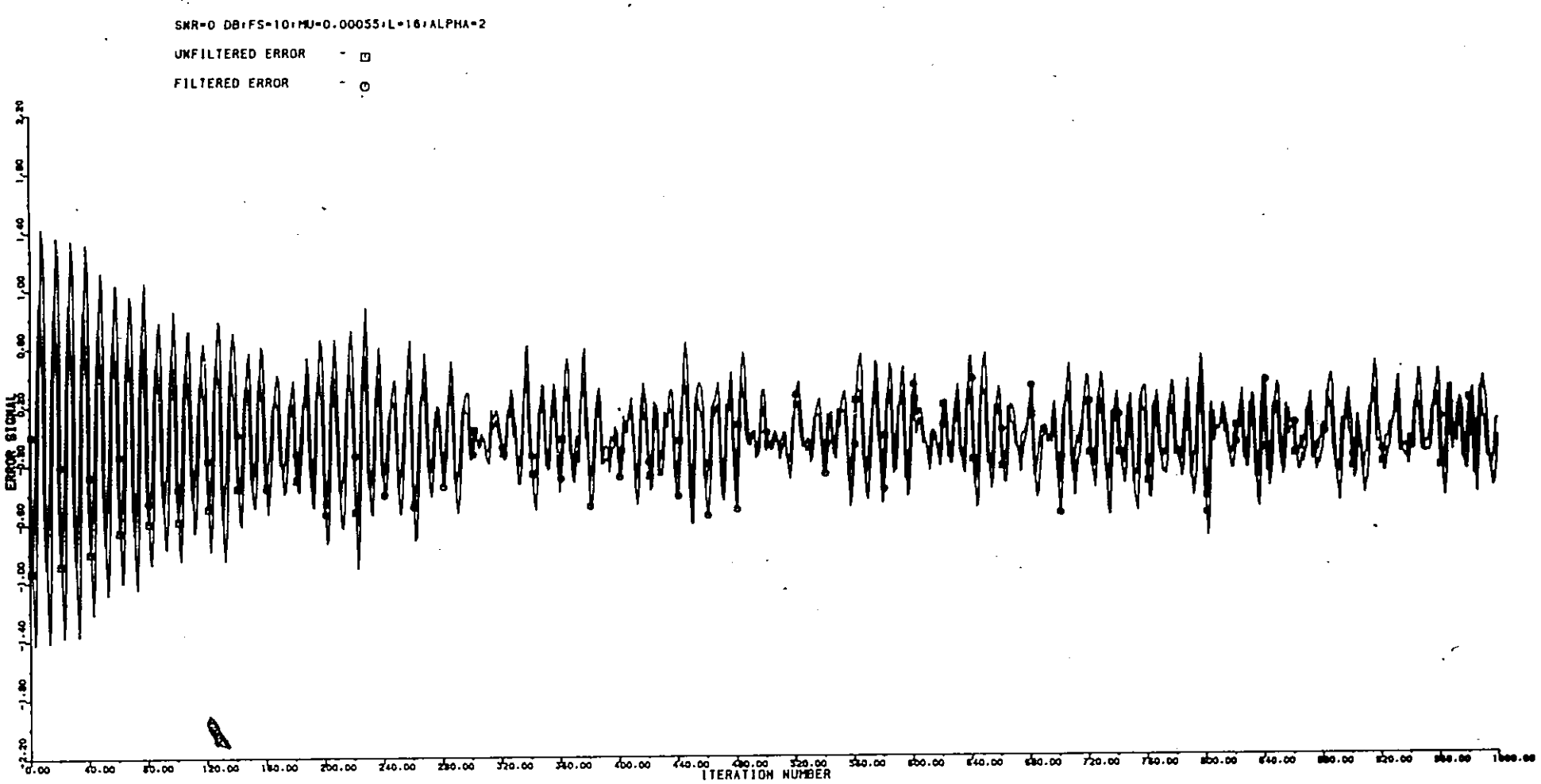


Fig.2.16 Comparison between the error signal and the filtered error signal for the FLMS algorithm

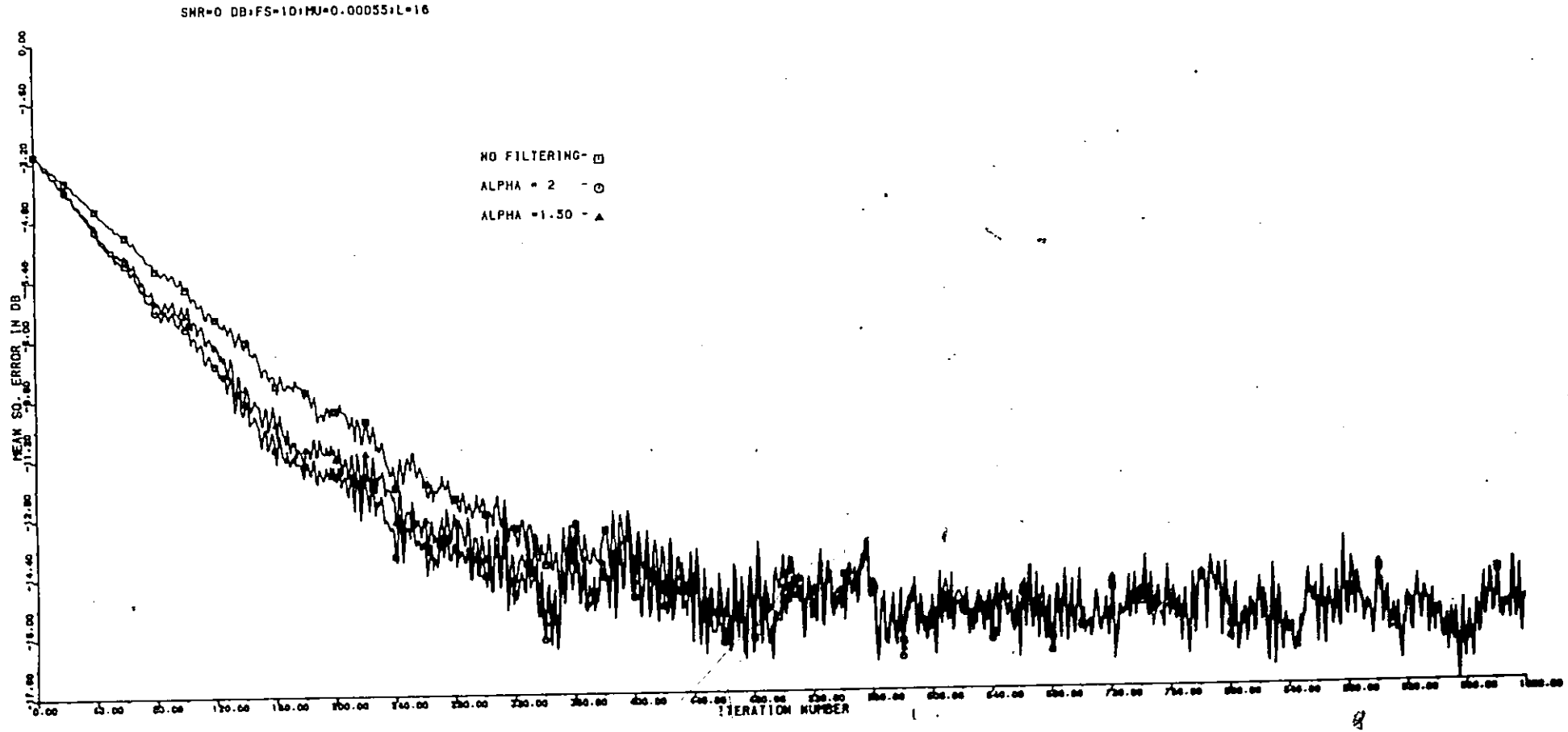
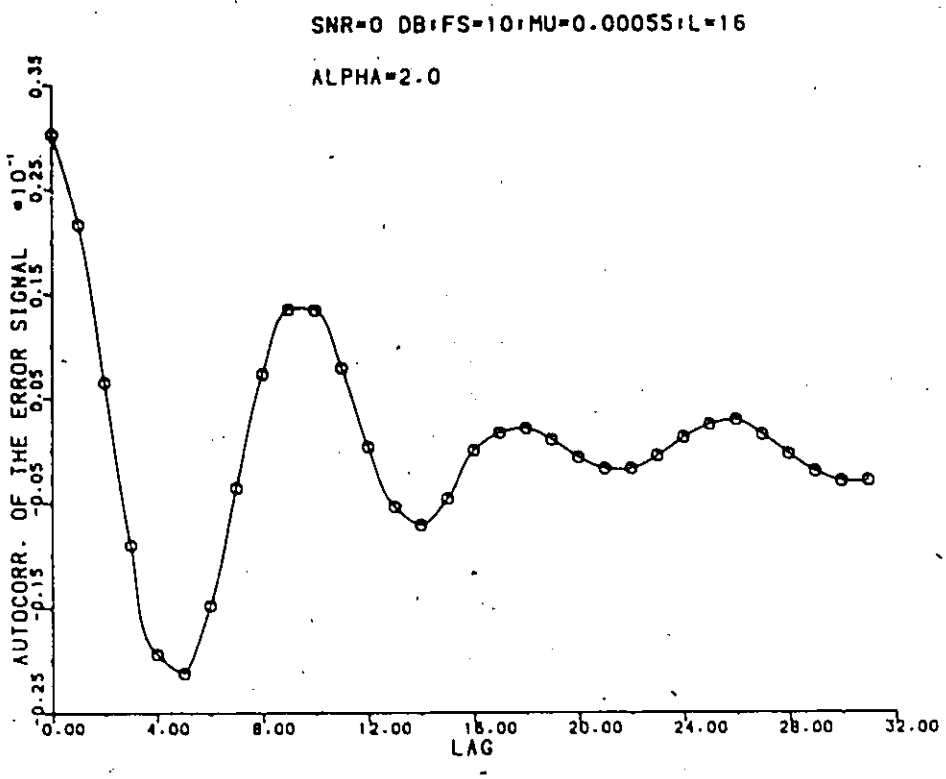
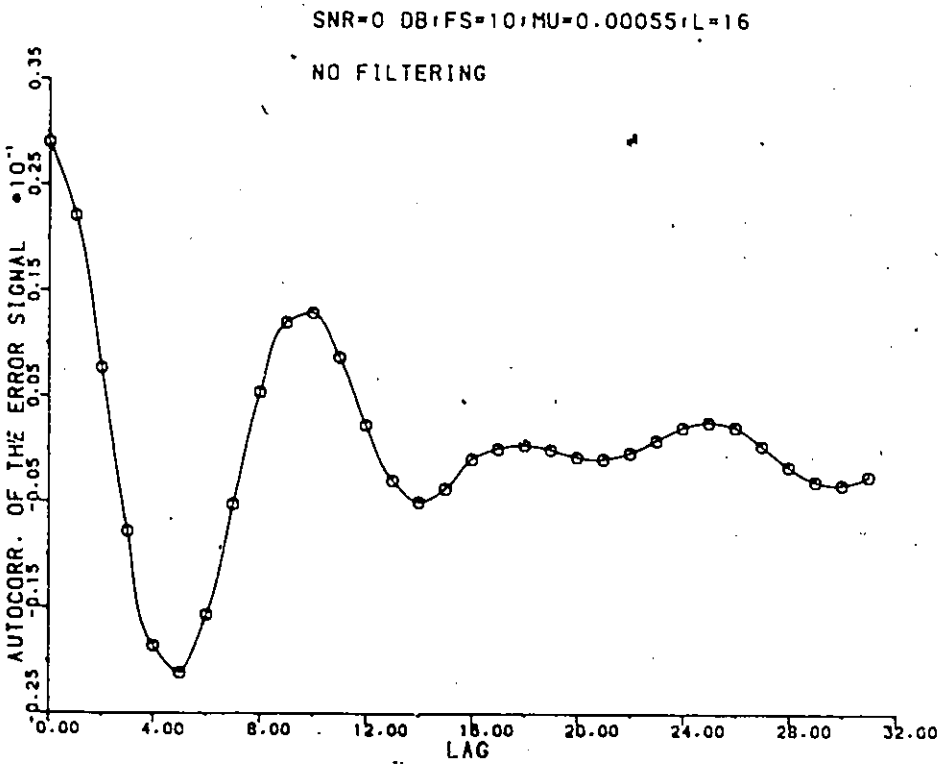


Fig.2.17 Learning curves on a logarithmic scale



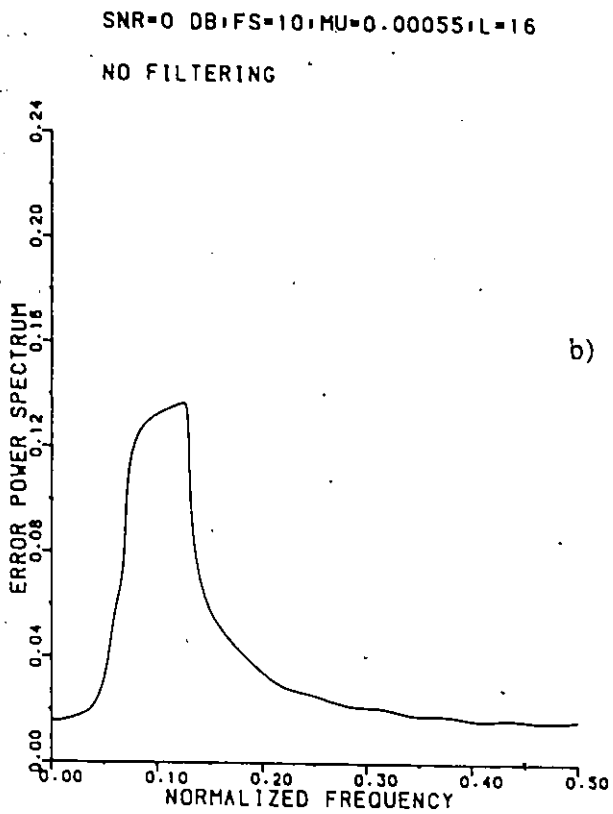
a)



b)

Fig.2.18 Autocorrelation function of the error signal after convergence:

a) FLMS algorithm, b) LMS algorithm



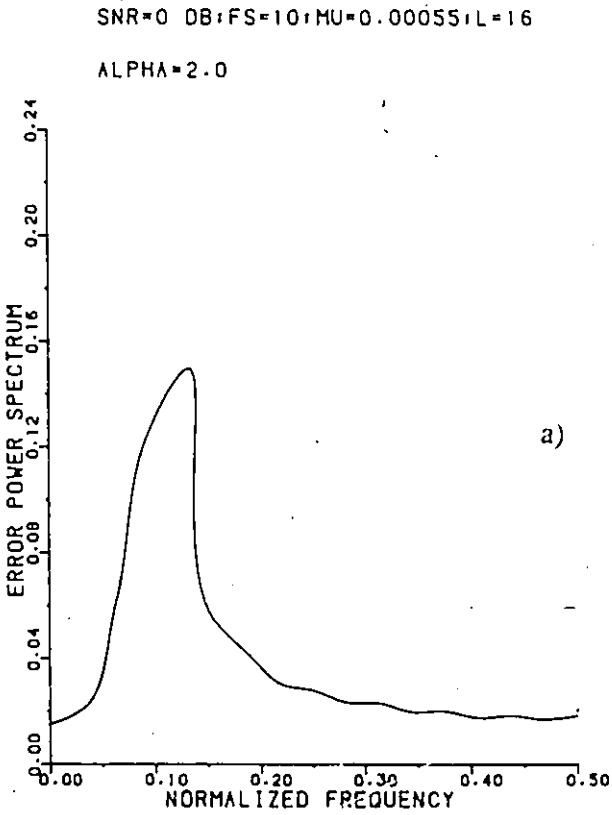


Fig.2.19 Power spectrum of the error signal after convergence:

a) FLMS algorithm, b) LMS algorithm

2.5 Possible Applications of the FLMS Algorithm

Good candidates for the FLMS algorithm are noise cancelling problems where a deterministic or narrowband signal is present either as desired for retrieval or for cancelling. Also, a priori information is needed about the approximate frequency range of this signal. The FLMS algorithm is more appealing for high speed real-time applications because it requires very little additional effort.

There is one more important use of the FLMS algorithm that has not been discussed yet. While doing simulations with the LMS algorithm, it has been found that under certain special conditions it could exhibit instability. The error filtering provides a kind of fading memory for the error values, thus helping to overcome the instability. If working with relatively small values of a (around 0.25), the performance of the FLMS algorithm will not be significantly different from that of the LMS algorithm over the entire frequency range. But this slight filtering can help in preventing instability. This problem will be addressed again and in more detail in chapter three.

A simple example, illustrating the use of the LMS and the FLMS algorithms, will be given. The chosen structure is the *self-tuning filter*, shown in Fig.2.20.

Let the input signal be composed of a periodic signal s and a broadband noise signal n. The delay, inserted before the adaptive filter, if sufficiently long, decorrelates the noise components in x and x_d but the periodic components are still correlated. Thus the adaptive filter, trying to minimize the output mean-square error, is able to predict and cancel the periodic component only. Depending on what is considered to be the useful signal, the output of the system can be taken from y, or from e. The signal y after convergence is the periodic component only, filtered from noise, and the signal e gives the noise only, without periodic interference (all this of course within the performance limits of the system).

Fig.2.21 shows the input signal to the filter. The signal-to-noise ratio is 3 dB; the periodic signal is a sine wave with unit amplitude and with frequency $0.0625 f_s$

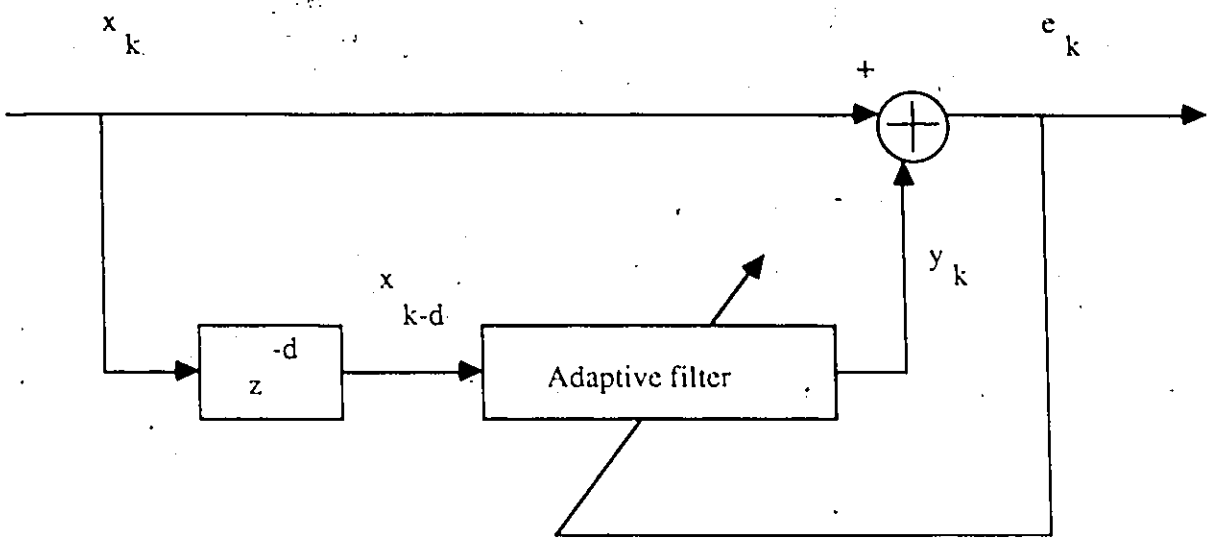


Fig.2.20 The self-tuning filter

(16 samples per period); the noise is white gaussian; the filter length is 32 taps.

In Fig.2.22 and Fig.2.23 is plotted the filter output y for the LMS and FLMS cases. The value of the coefficient a is 0.5. The results of the filtering are seen to be the same for both cases, except that convergence is faster with the FLMS algorithm. In Fig.2.24 the input and output signals are shown in extended scale in order to observe the results more conveniently. The improvement is obvious. The signal-to-noise ratio in the output is found to be 10 dB, i.e. 7 dB improvement is achieved.

Details concerning the self-tuning filter are not a subject of special interest in this thesis and will not be discussed any further.

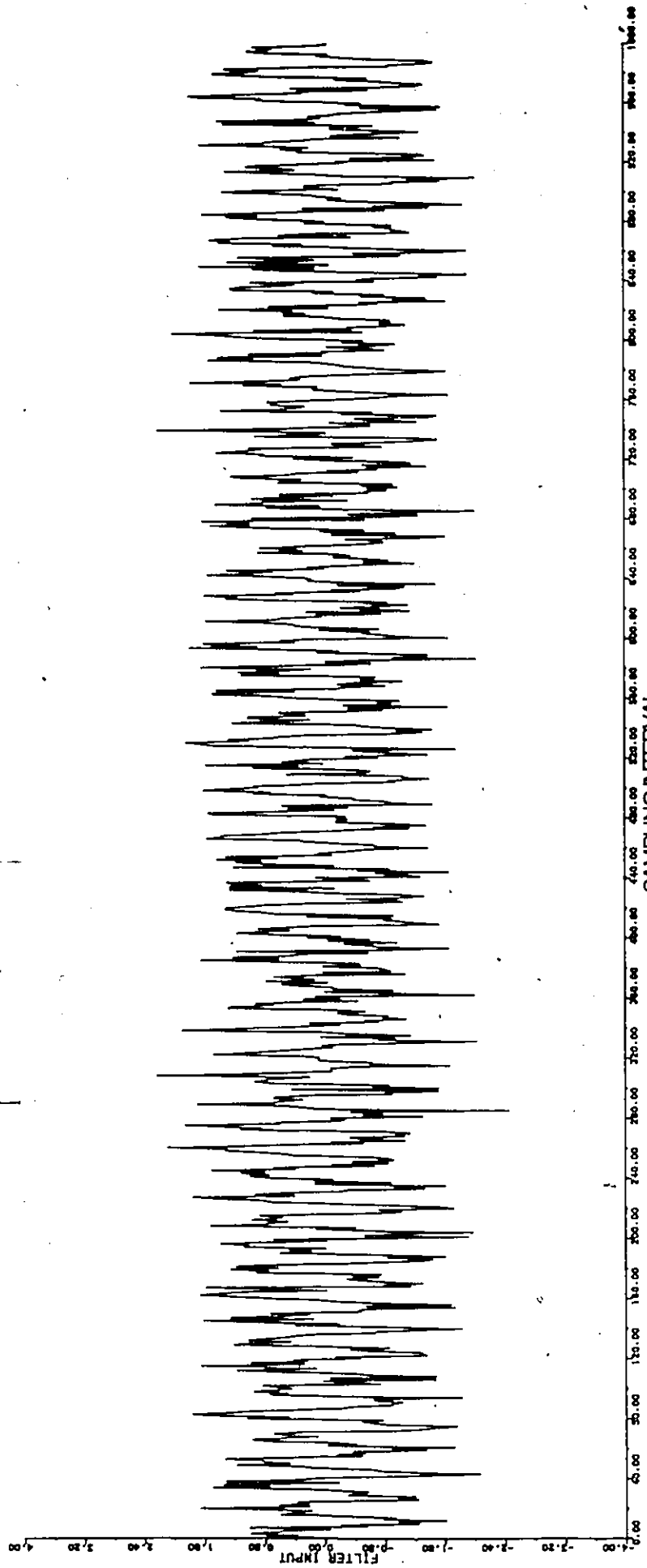


Fig.2.21 Input signal

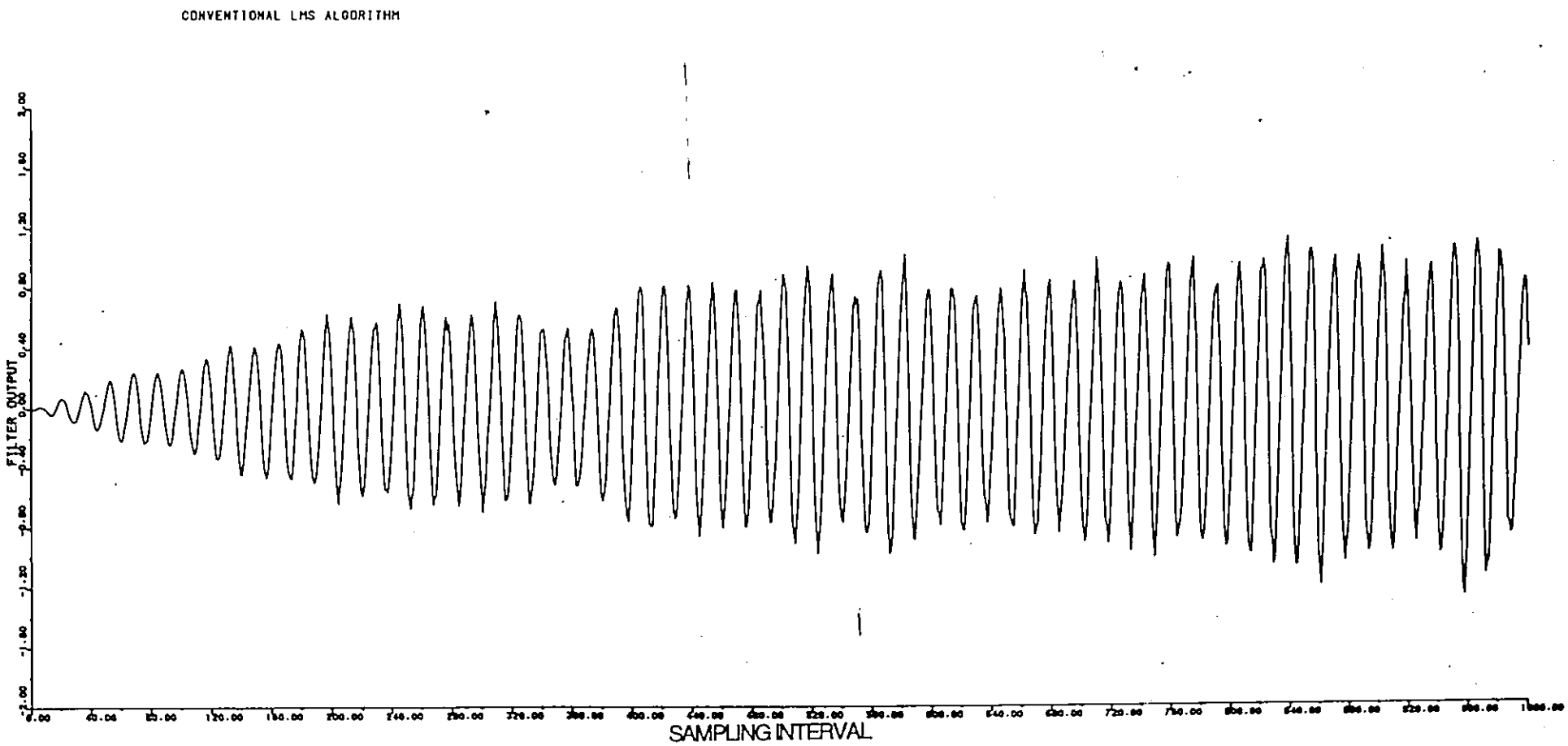


Fig.2.22 Output signal – LMS algorithm

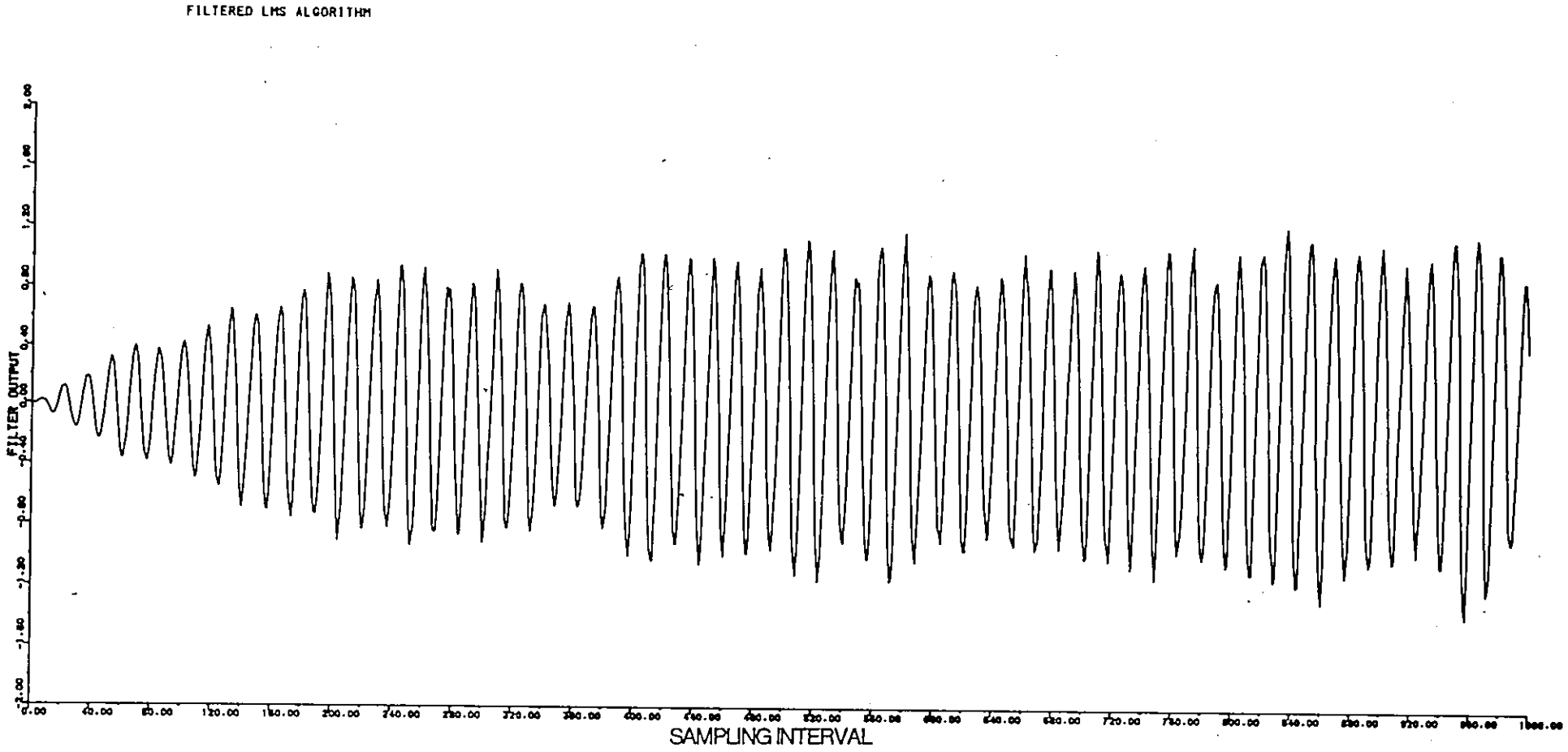
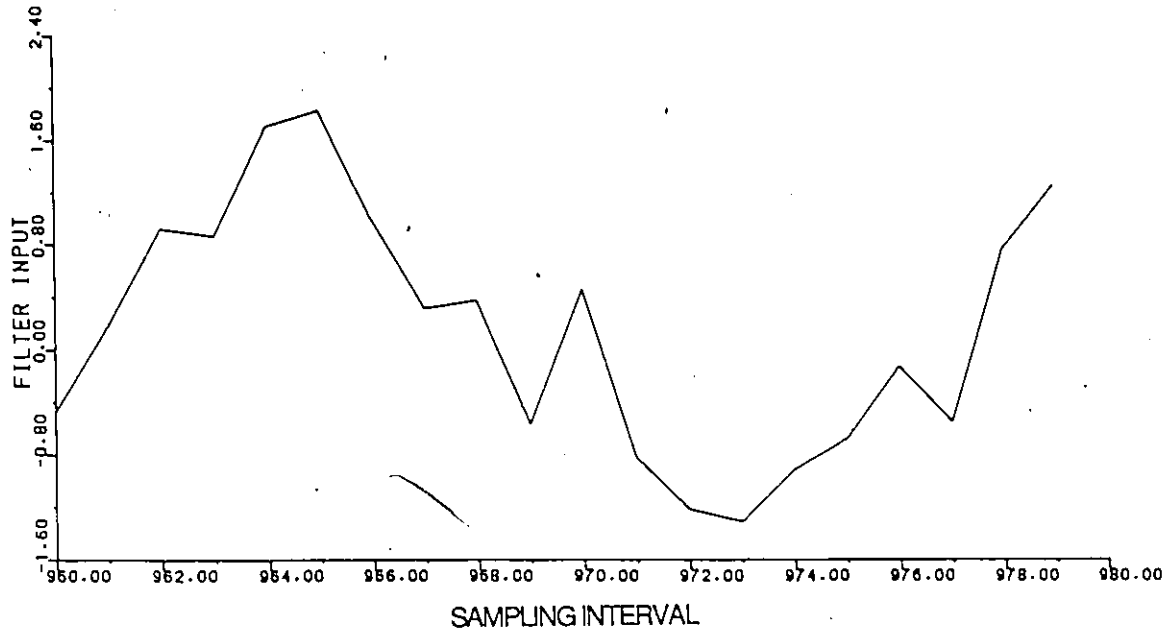
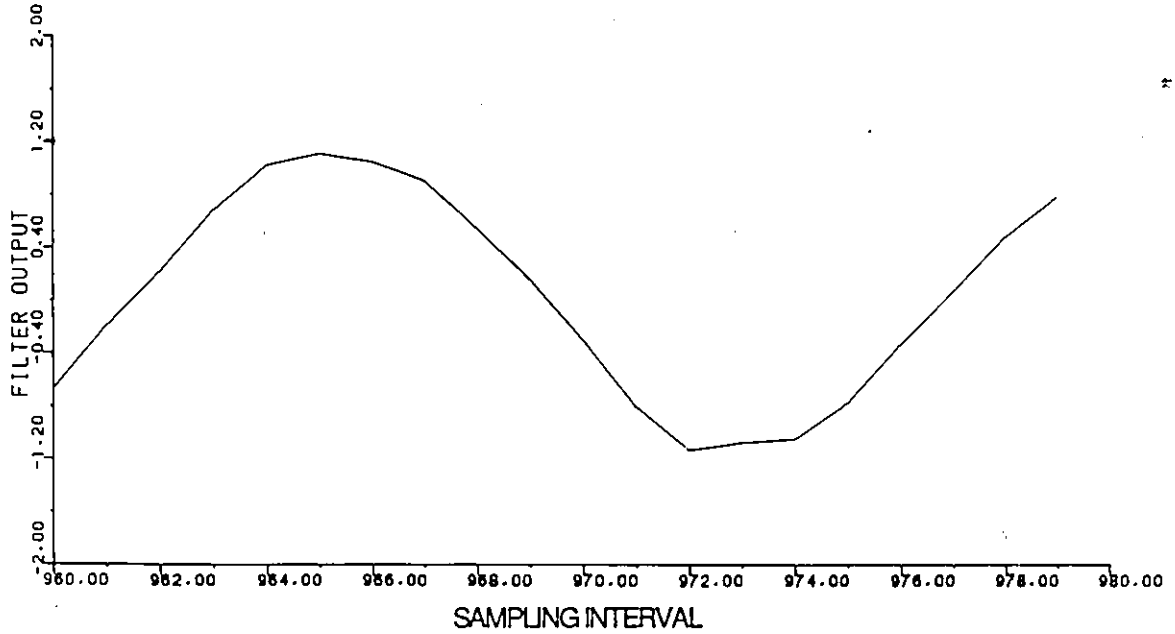


Fig.2.23 Output signal -- FLMS algorithm



a)



b)

Fig.2.24 Extended time base plots:

a) input signal, b) output signal from the LMS algorithm

CHAPTER III

MODIFIED LMS ALGORITHMS FOR VLSI APPLICATIONS

3.1 The Problem of Sequential Coefficient Access

In our discussions so far the attention was focused on the most popular form of LMS adaptive algorithms, where all filter coefficients are accessible in parallel and can be updated every sampling interval. This is the most straightforward form of the algorithm, for which the mathematical derivation was shown in chapter one. It is also the most efficient one in terms of adaptation speed and performance under nonstationary conditions [4,12].

In some cases though, it is not possible, or not necessary, to update all the filter coefficients at the same time and for every sampling period. This is the case especially for hardware implementation, where simplicity and minimization of the number of components are always basic aims.

As can be seen from Fig.1.4, a direct hardware implementation of the "standard" LMS algorithm requires at least one delay line, L+1 multipliers for the transversal part of the filter, L+1 multiplier-accumulators for the adaptive part, two summing elements and, of course, some additional hardware. It is a well known fact that the cost of a multiplier is relatively high compared to other hardware components, especially for high-speed devices. For that reason it is important to keep the number of multipliers as low as possible without degrading too much the system performance.

One solution to this problem is to use a few multipliers on a time-shared basis in both the FIR and adaptive parts, or just in one of them. Thus the algorithm will perform in virtually the same way but the time-shared elements must do many operations between two iterations in *burst mode*. This approach is suitable if the sampling rate is relatively low compared to the maximum speed of the components

but it is clearly not applicable when very high speed is needed and all components must work almost at the limit of their capacities.

With the advent of the VLSI (Very Large Scale Integration) era tremendous progress has been made in reducing the number of hardware components in electronics design in general and in signal processing applications in particular. There are now commercially available single-chip integrated circuits that incorporate an entire transversal (FIR) filter and can operate at clock speeds of 10 MHz and up. A schematic diagram of such a device is shown in Fig.3.1. There is a port for the input signal x, an output port y, a port for changing the coefficients and an address bus for selecting the coefficients.

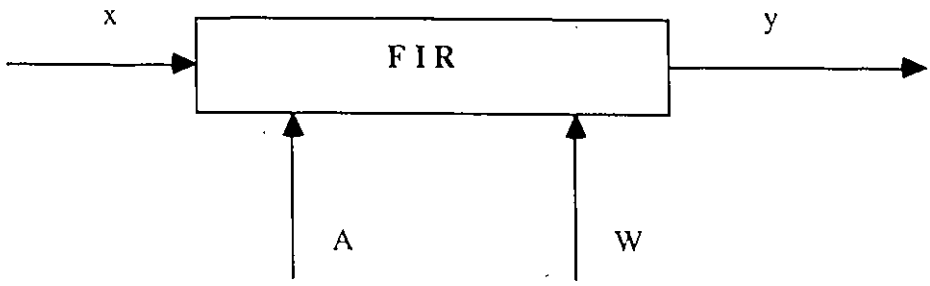


Fig. 3.1 Single-chip programmable FIR filter

In this form the filter should be called *programmable*, instead of adaptive. It must be emphasized here that only *one* coefficient can be accessed at every clock cycle via the W-port and an appropriate address to select the coefficient.

An obvious bottleneck exists here, but it is not likely to be overcome soon because the number of pins on an integrated circuit is the limiting factor. It is physically not possible to provide a separate port for every coefficient, especially when this number is high (and the trend is towards more and more elements on a single chip).

The above-mentioned programmable filter poses no problems for non-adaptive applications. But when used for adaptive filtering, it can create significant loss in convergence speed.

The "ideal" in that case would be to incorporate both the transversal and the adaptive parts into a single chip, as in Fig.3.2 and provide input/output pins for x, y, d (desired output), e (error signal) and w for eventually initializing the coefficients to certain values. The problem now is that the device becomes too specialized and with quite narrow application. The author is not aware of any commercially available device of that kind. Some experimental custom-designed chips exist [6, 7].

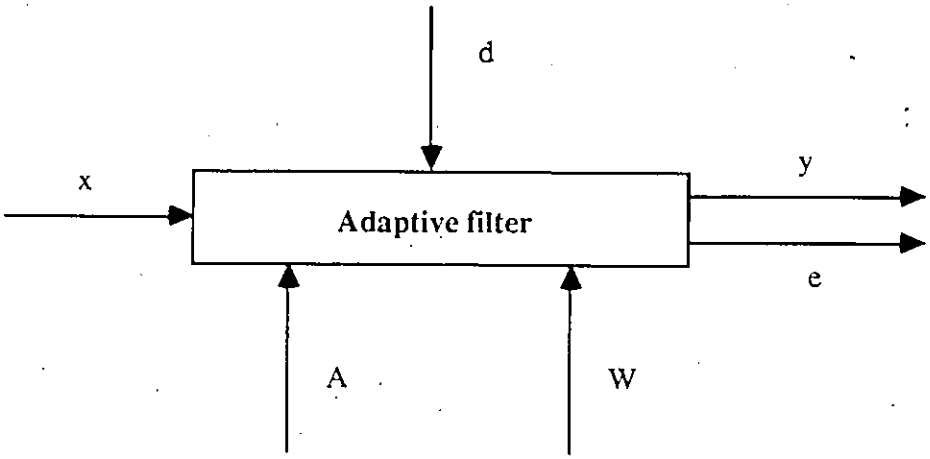


Fig.3.2 Single-chip fully adaptive filter

The seriousness of the sequential coefficient access problem is clear from the above discussion. Since there is no evidence that the problem will be solved soon and the use of VLSI circuits is growing very rapidly, it is important to investigate new (or modified) adaptive algorithms suitable for this kind of application. In the following sections of chapter three the LMS algorithm will be the object of such an investigation.

3.2 Sequential Coefficient Access LMS Algorithms

The most straightforward way of modifying the LMS algorithm into a serial version is by performing the update operation only once every L+1 sampling periods. Then in the basic equation for weight vector recursion

$$\mathbf{W_{k+1}} = \mathbf{W_k} + 2\mu e_k \mathbf{X_k} \tag{3.1}$$

the index k stands again for the iteration number, but there are L sampling periods between two iterations. A more relevant notation can be introduced and the update equation can be written as:

$$w_{i,k+1} = w_{i,k} - 2\mu e(k(L+1))x(k(L+1)-i) \tag{3.2}$$

$$\text{for} \quad i = 0, ..., L$$

Note that here w and x are not vectors but single values. The subscript i stands for coefficient number (or position in the transversal structure) and k is the iteration index. The values of e and x correspond to a specified *sampling interval*. It is seen that e is not a function of the coefficient number i, which means that it remains constant between two iterations. Equation (3.2) suggests the following way of performing the adaptation process:

1. At iteration moment k all new values of the filter coefficients are computed at the same time, according to equation (3.2).

2. Between iterations k and $k+1$ one filter coefficient at a time is replaced by its new value at every sampling period.

3. Between iterations k and $k+1$ no intermediate modification of the computed coefficient values is done.

4. The transversal part of the filter generates an output every sampling interval.

Here it is important to clarify the use of the term *update* in the present discussion. To update the coefficient vector means to *compute* the new set of coefficients in the

adaptive part of the system. It does not mean at this point that the transversal part automatically accepts the new coefficients, as is the case with the conventional LMS algorithm. The newly computed coefficients must *replace* the old values. This is a separate operation, which is done independently, as described before.

Since the coefficients are changed once every L+1 sampling intervals, then the speed of convergence is on the order of L+1 times slower than the standard LMS algorithm. The only positive effect is that the delay between consecutive iterations decreases the correlation between two input vectors, so the algorithm will perform in accordance with the Independence Theory [14].

Fig.3.3 shows a possible implementation of this algorithm. The need for L+1 multiplier-accumulators, working in parallel, is obvious. The adaptive part is clocked by a signal with L+1 times lower frequency. This particular modification will be referred to as case No.1.

The architecture proposed in Fig.3.3 is highly inefficient, since it makes use of all those multipliers only during a small portion of the operating time. However, the same architecture can be used to perform in a much better way simply by allowing the coefficients to be constantly updated (computed) between two iterations. Thus a new set of coefficients is computed every sampling interval, as in the conventional LMS algorithm, but then those values are inserted into the transversal filter one by one. The update equation is exactly like (3.1) and k is the sampling interval. The difference is that the value of the error signal e_k is not exactly the same as in (3.1), because the output signal y_k results from convolution with a weight vector different from the computed one (from one sampling period to the other only one coefficient is changed). The decrease in error values will tend to be slower than in the conventional LMS algorithm but on the other hand this will result in bigger steps in the coefficient updates, which in turn will drive the filter towards convergence more rapidly. As a result of these two opposite processes, the modified algorithm performs virtually as efficiently as the conventional LMS algorithm. This modification will be

called parallel coefficient computation with intermediate update and will be denoted as case No.2.

In equation (3.1) the error e_k can be replaced by its full expression, and taking the expected values from both sides gives:

$$E[\mathbf{W_{k+1}}] = E[\mathbf{W_k}] + E[2\mu(d_k - \mathbf{X_k^T W'_k})\mathbf{X_k}] \tag{3.3}$$

Here a prime is used to denote the coefficient vector that is loaded in the transversal part to produce the actual output, in contrast with the unprimed values, which are stored in the adaptive part. Knowing that $E[\mathbf{X_k X_k^T}] = \mathbf{R}$ and $E[d_k \mathbf{X_k}] = \mathbf{P}$ (3.3) becomes†:

$$E[\mathbf{W_{k+1}}] = (\mathbf{I} - 2\mu E[\mathbf{W'_k}]E[\mathbf{W_k^{-1}}]\mathbf{R})E[\mathbf{W_k}] + 2\mu\mathbf{P} \tag{3.4}$$

This expression differs from the similar one for the "true" LMS algorithm only in the presence of the term $E[\mathbf{W'_k}]E[\mathbf{W_k^{-1}}]$ which in general must be different from the identity matrix **I**. The effect of this term on the convergence speed is difficult to assess. But for relatively small filter lengths $\mathbf{W'_k} \approx \mathbf{W_k}$, $E[\mathbf{W'_k}]E[\mathbf{W_k^{-1}}] \approx \mathbf{I}$ and the term can be neglected. Simulation results which will be given later seem to confirm this assumption.

Both architectures proposed so far in this section are quite controversial. First they use an integrated single-chip FIR filter in order to reduce the hardware complexity. Then it turns out that a separate delay line and L+1 multiplier-accumulators are necessary for the adaptive part, which increases the complexity. For that reason an alternative approach is more suitable. Only one multiplier and one adder can be used to compute all coefficient updates. Such a realization is possible if only one coefficient is updated every sampling interval. This approach is suggested in [30,31]. A simplified block diagram is shown in Fig.3.4. The sample/hold device is clocked at every L+1 sampling intervals to allow a new value of e to be used. The memory unit

† A detailed proof is given in appendix A

plays the role of an accumulator, i.e. it is used to store previous coefficient values and can be implemented by a simple barrel shifter. The delay unit z^{-L} can also be implemented using a shifter (FIFO).

Note that the coefficients are updated backwards, i.e. starting from coefficient w_L to coefficient w_0. The update equation is the same as for case No.1 but the computation is not done in parallel and the index i goes from L to 0. The performance is still the same as in case No.1.

A slight modification of this latest version is possible which consists in not keeping the value of e constant between two iterations. Instead, a different value of e is used for every coefficient update and there is no need for a sample/hold device. Equation (3.2) then becomes:

$$w_{i,k+1} = w_{i,k} + 2\mu e(k(L+1) + L - i)x(k(l+1) - L + i) \qquad (3.5)$$

$$\text{for} \quad i = L, ..., 0$$

and k means iteration number.

This modification is expected not to affect seriously the performance of the latest version. It is probably even desirable, because in that way the algorithm uses more statistical information and will eventually find the way to the optimum solution with fewer fluctuations. This latest modified version will be referred to as case No.3.

The slight modification described above can prevent start-up problems that are possible with the algorithm proposed in [30]. Suppose that initially all coefficients are set to zero, which is common practice. Suppose also that the desired signal is a sine wave with period that fits an exact integer number of times into the filter length. If the value of the error signal happens to be taken at a moment when the desired signal crosses zero, then the update correction will be zero, i.e., $e_k = y_k - d_k = 0$ and $2\mu e_k \mathbf{X_k} = 0$. Thus all coefficients will remain unchanged, since e_k is kept the same for all updates. At the next iteration the algorithm will hit again at the same

point of the sine wave with the same result. Theoretically this can continue forever and no convergence is possible. On the other hand, if the error signal e, used in the update equation, is allowed to change at every sampling interval, such a problem will not occur.

But another problem is present — some of the coefficients will be updated in a completely random manner. When the input sinusoid and the "desired" sinusoid are near zero at the same moment, the error signal contains only a noise component. In this case slight error filtering, as described in chapter II, can help. The filtered error provides information about past values of the error signal which contain a deterministic component. Although much more slowly than normally all filter coefficients will converge to the optimum solution.

The kinds of situations described above have been observed during computer simulations. They are very unlikely to happen in real physical applications but are theoretically possible and should be considered seriously. Moreover, the proposed modifications do not require any significant design effort — just the opposite, they are simpler and avoid the use of a frequency divider and sample/hold device, and the error filtering element is very simple to implement.

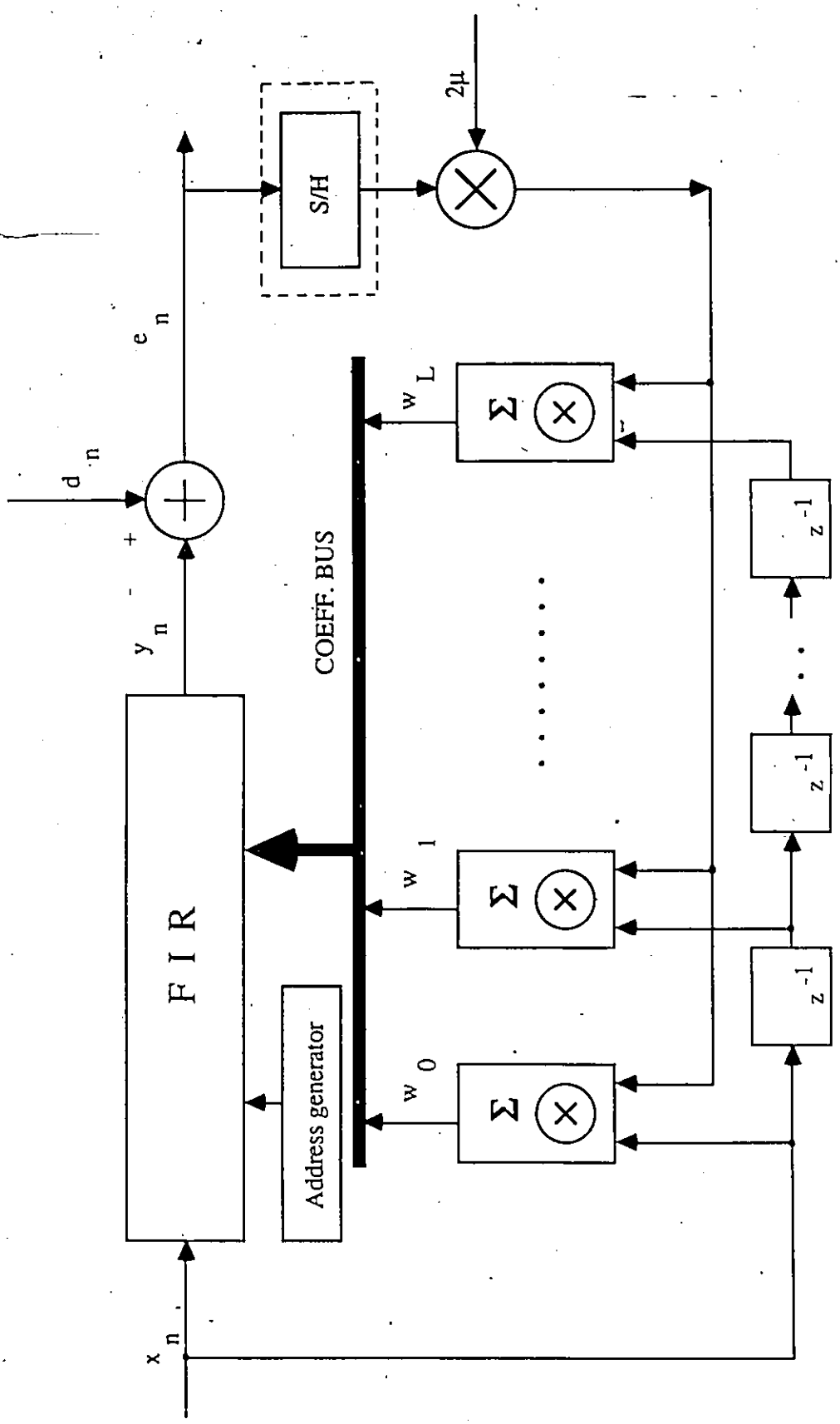


Fig.3.3 LMS adaptive filter with sequential coefficient access

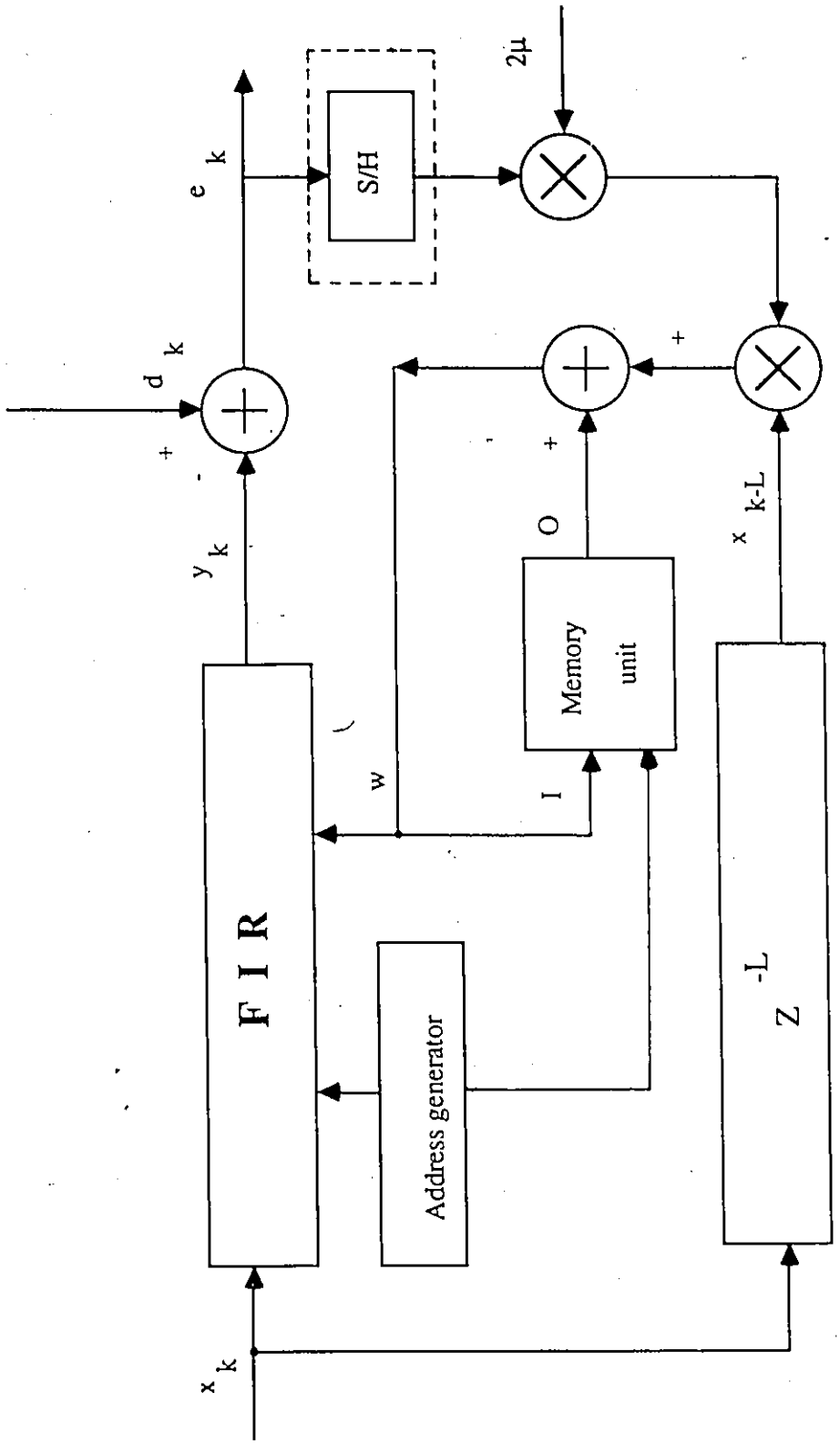


Fig. 3.4 Transversal adaptive filter with serial update

3.3 Simulation Results

In order to support the results discussed in this chapter, software simulations have been carried out similar to those in chapter two. The input and the desired signals are generated in the same way. The filter length is 16 taps and the rest of the simulation conditions, like input SNR, sampling frequency etc., can be seen on the figures.

In Fig.3.5 are shown learning curves for cases No. 1, 2 and 3. It is clear that case No.2 has much better convergence properties than the other two. The performance of case No.1 and case No.3 is virtually the same in terms of convergence speed but for case No.3 the behavior of the mean-square error is less erratic, as was suggested in the previous section.

In Fig.3.6 are shown the mean values of one of the filter coefficients (w_6) in time for the three cases. The staircase form reflects the fact that the coefficient is replaced by a new value every $L+1$ samples (i.e. 16 samples). Case No.3 demonstrates slightly faster convergence compared to case No.1.

In Fig.3.7 a comparison is done between the learning curves of the conventional LMS algorithm with parallel coefficient access and case No.2 with parallel update but sequential coefficient access. Both algorithms give practically the same results. But the price to pay for that was shown to be costly in hardware.

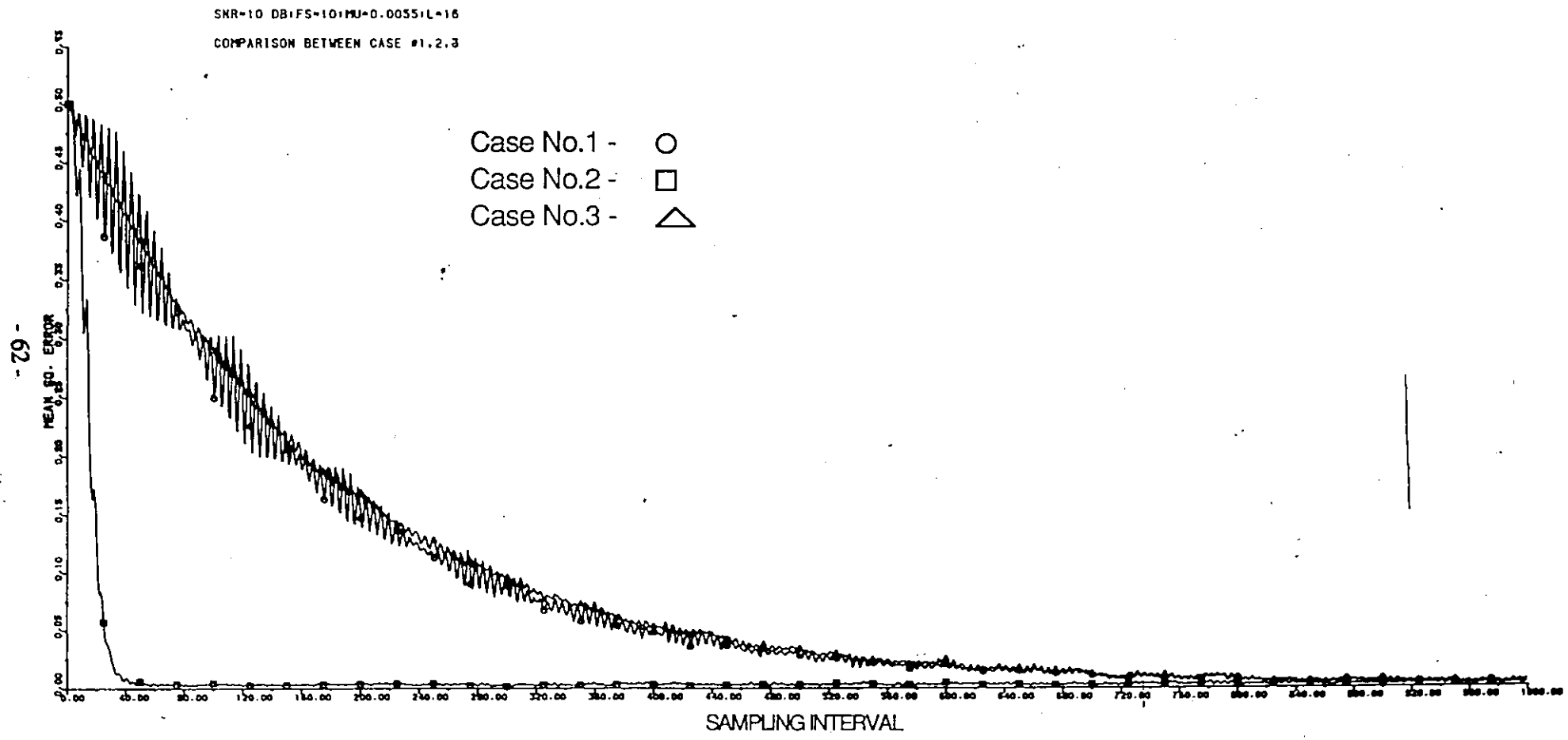


Fig.3.5 Learning curves for case No.'s 1, 2 and 3

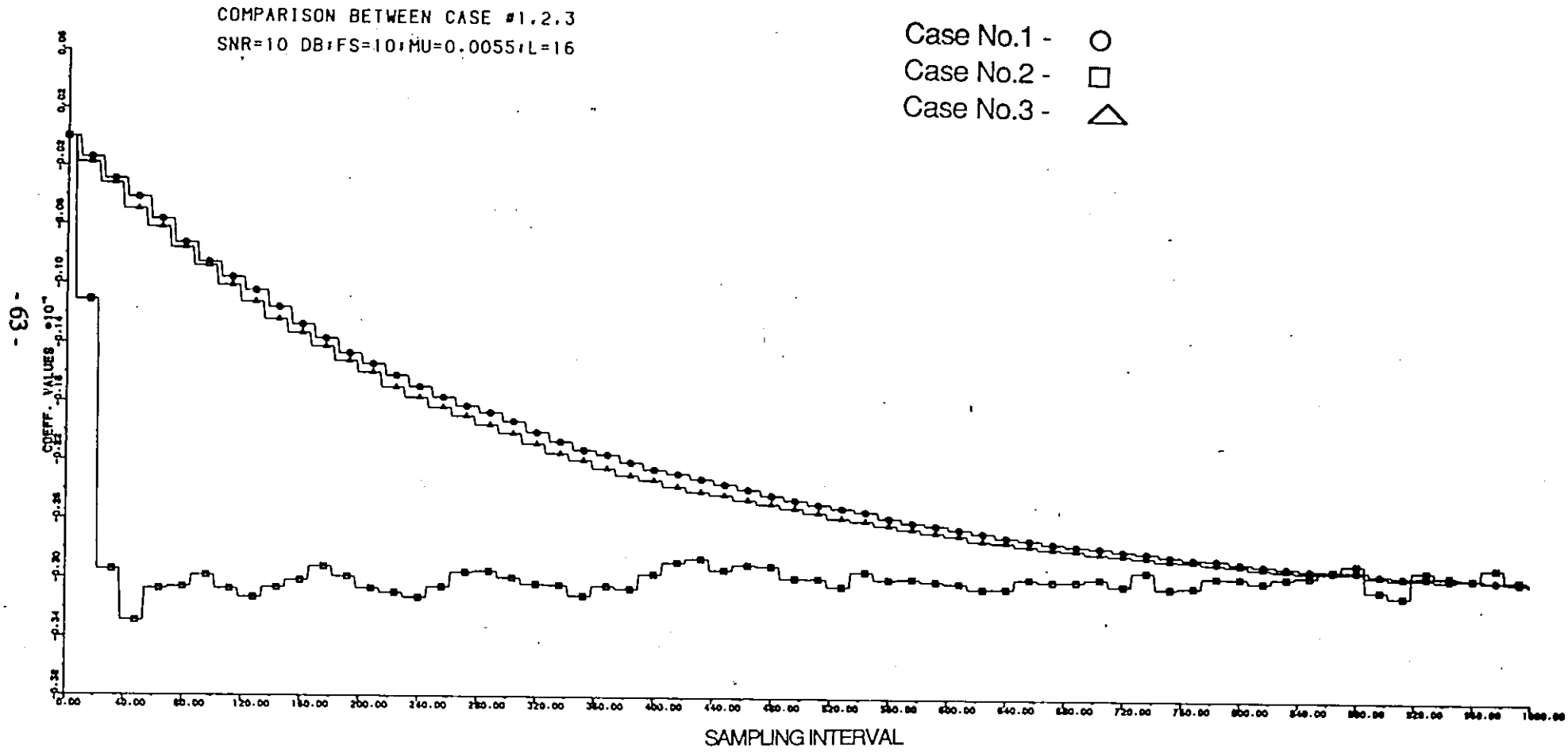


Fig.3.6 Mean values of coefficient w_6 for case No.'s 1, 2 and 3

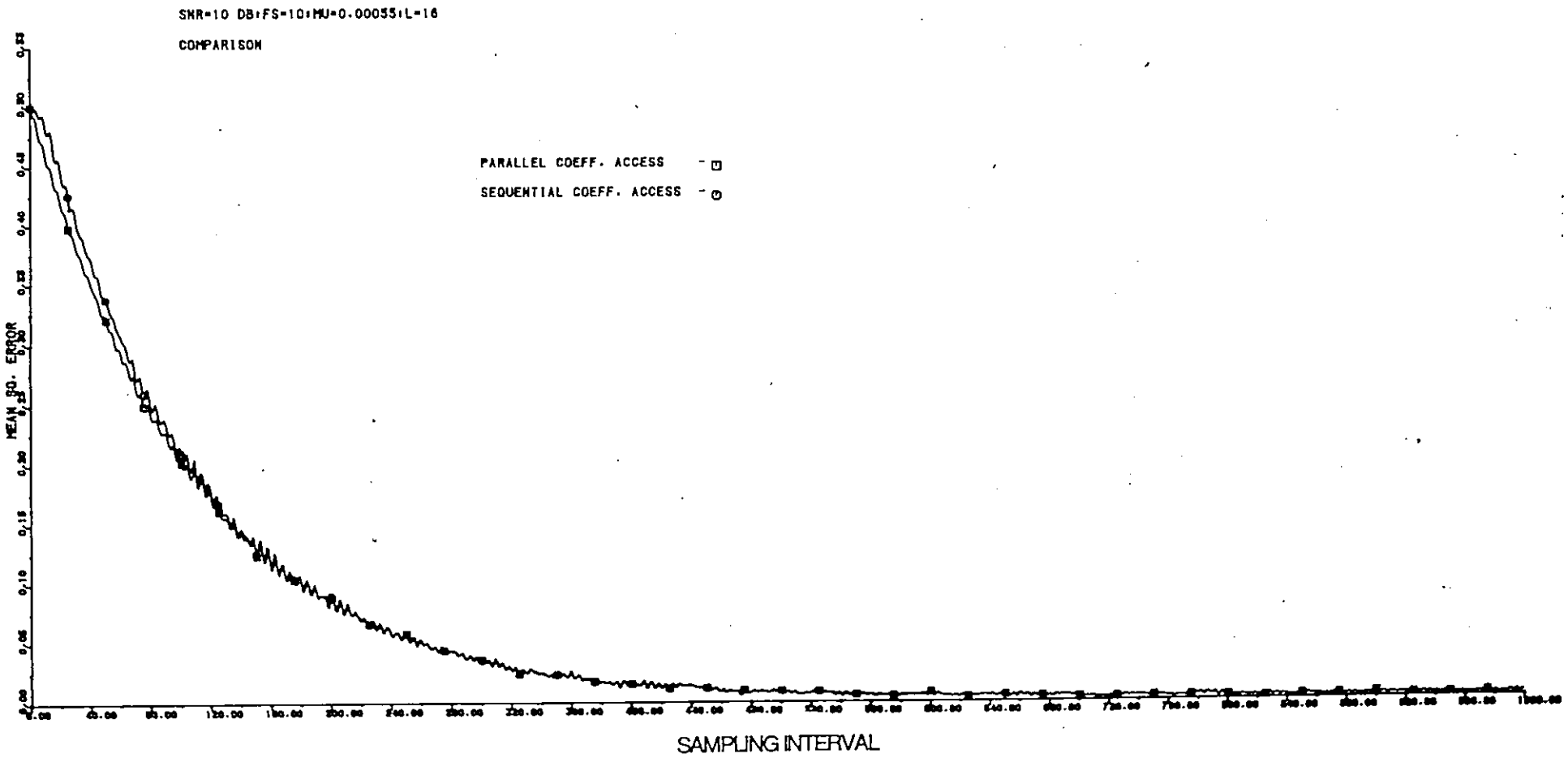


Fig.3.7 Learning curves for conventional LMS algorithm and case No.2

3.4 Digital Precision Considerations

In all discussions so far there was no concern about the precision with which computations were carried out. It was assumed that infinite precision was available. Now it is time to address this important question because, as will be seen, significant differences may result when working with limited precision, especially with fixed-point arithmetic.

The theory of quantization error analysis in non-adaptive systems is well developed and relatively easy to understand and apply [24]. The errors due to finite word length are interpreted as additive white noise with uniform distribution and variance $\sigma^2 = 2^{-2B}/12$ (assuming rounding), where B is the number of significant bits in the signal representation. In adaptive systems, however, the finite word length affects the system performance in a more complicated way and can be more damaging.

As was shown in chapter 1 (1.23), the misadjustment after convergence depends (except for external conditions) on the value of the adaptation constant μ and the filter length $(L+1)$:

$$M \approx \mu \xi_{min} (L+1)(signal\ power) \tag{3.6}$$

Thus one of the design parameters that can control the misadjustment is the step size μ. In fixed-point implementations a very small value of μ can actually degrade performance. This happens because the correction term $2\mu e_k x_{k,i}$, $(i = 0, ..., L)$ can become smaller than the value of the least significant bit in some or in all filter coefficients. Then the corresponding coefficients are not updated, the adaptation process stops before it is completed and the output mean-square error remains much higher than the minimum possible. It may turn out that additional tap weight precision is necessary in order to achieve the desired level of misadjustment. This higher precision was not necessary from the point of view of quantization errors.

For the discussion of the present topic results will be used from the work of Caraiscos and Liu [10] and Gitlin and Weinstein [11]. A simplified diagram of an adaptive system, using fixed-point arithmetic, is shown in Fig.3.8 . The primes denote quantized values. The desired signal d_k is scaled by multiplying by a constant b. This is often just a right shift, and $b = 1/2$ in most of the cases.

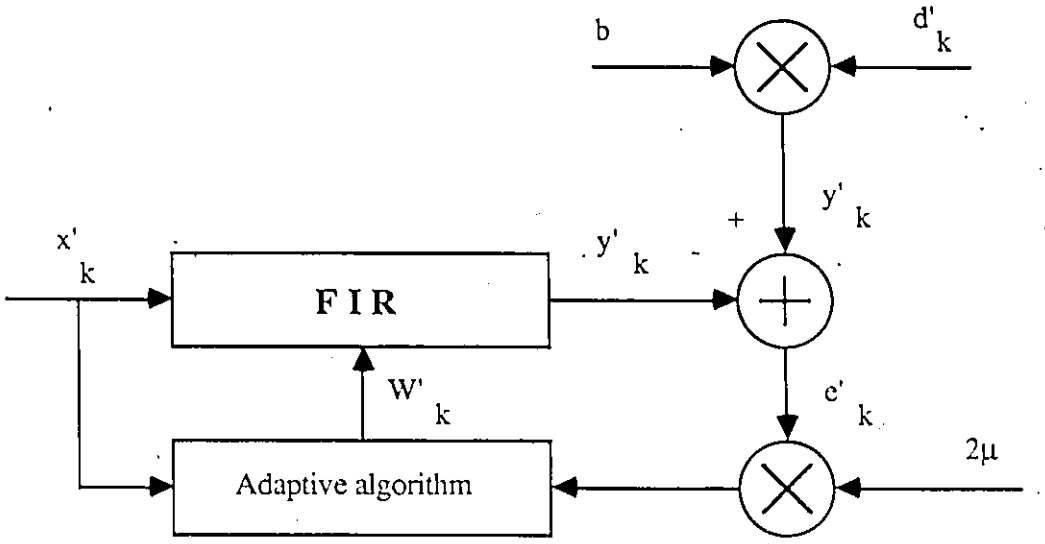


Fig.3.8 Adaptive system with finite precision arithmetic

It is assumed that the input signals are properly scaled and that their values lie between -1 and +1 . The data values are represented by B_d bits plus sign, and each filter coefficient by B_w bits plus sign. It is assumed also that no overflow occurs.

The condition for the i-th element of the weight vector to continue being updated is:

$$|2\mu e'_k x'_{k,i}| \geq 2^{-B_w} \tag{3.7}$$

This condition can be approximated with sufficient confidence [11] by replacing the magnitudes by $\sqrt{2}$ times their RMS (Root Mean Square) values and also assuming

the RMS values of all components $x'_{k,i}$ to be equal. Then:

$$\mu\sqrt{2}\sqrt{E[e'^2_k]E[x'^2_{k,i}]} \geq 2^{-B_w-1} \tag{3.8}$$

and also $E[e'^2_k] = \xi$ = MSE and $E[x'^2_k]$ is the input signal power. For inequality (3.8) to be kept ξ must increase when μ becomes too small.

Caraiscos and Liu [10] have found a simplified expression for the optimum value of μ that minimizes the mean-square error, provided that the algorithm has converged and for the case $B_w = B_d = B$. Adapted to our notation it is:

$$\mu^* = \frac{2^{-B}}{2b}\sqrt{\frac{(L+1)}{3\xi_{min}tr[\mathbf{R}]}} \tag{3.9}$$

This is the value of μ for which the error due to misadjustment is equal to the roundoff error. However, they prove that this value is too small to allow the algorithm to converge completely. In practice one should work with values of μ bigger than μ^* and (3.9) turns into an inequality. It can be simplified a bit more. As was mentioned before, b is usually 0.5. Also $tr[\mathbf{R}] = (L+1)(\textit{signal power})$ and the input signal power can be assumed to be 0.5, which is the power of a sine wave with unit amplitude. Thus for the adaptation constant μ we obtain:

$$\mu > 2^{-B}\sqrt{\frac{2}{3\xi_{min}}} \tag{3.10}$$

Remark: In theory ξ_{min} can become zero and $\mu \to \infty$, which obviously does not make sense. But in practice the output error power cannot get smaller than the roundoff error power, which is:

$$\sigma_e^2 = (L+1)\sigma_w^2 + \sigma_d^2 = (L+1)\frac{2^{-2B_w}}{12} + \frac{2^{-2B_d}}{12} \tag{3.11}$$

Expression (3.10) is a simple formula for fast initial design of a fixed-point system. When some a priori information is available (e.g. channel characteristics, initial input statistics etc.) ξ_{min} can always be computed from (1.13).

It must be stressed here that (3.10) provides the lower bound for μ in the steady state. During initial adaptation μ can be given larger values for fast start-up. It is well known [11] that the fastest initial adaptation is achieved with $\mu = 0.5\mu_{max}$ and μ_{max} is the maximum permissible value, given by (1.19).

Once a value of μ is chosen, the steady state mean-square error can be determined. When the same word length is used for both data and coefficients, i.e. $B_d = B_w = B$ and μ is not very large, the following expression is obtained:

$$\xi_\infty = \xi_{min} + \frac{1}{2}\mu\xi_{min}tr[\mathbf{R}] + \frac{(L+1)\sigma_w^2}{2b^2\mu} + \frac{1}{b^2}(\mathbf{W}^{*\mathbf{T}}\mathbf{W}^* + L + 1)\sigma_d^2 \qquad (3.12)$$

where $\sigma_d^2 = \sigma_w^2 = \frac{2^{-2B}}{12}$.

3.5 Concluding Remarks

The main objective of this chapter was to address the issue of sequential coefficient access in the LMS algorithm. It was shown that even in the case of sequential coefficient access virtually the same performance as with the conventional LMS algorithm could be achieved but with significant hardware complexity. A simple modification using a sequential update was proposed which is shown in Fig.3.4. This particular architecture is the most efficient and is recommended for high-speed VLSI applications.

CHAPTER IV

HARDWARE IMPLEMENTATION OF A HIGH-SPEED ADAPTIVE FIR DIGITAL FILTER

4.1 The TRW TDC1028 Chip

In this chapter a hardware architecture for a high-speed adaptive FIR digital filter is examined. The main building blocks are the TDC1028 integrated circuits, manufactured by TRW Inc.

In Fig.4.1 the functional block diagram of a TDC1028 chip is shown. In a 48-pin package it incorporates an 8-tap FIR programmable filter. The word length of the input signal is four bits and the coefficient word length is also four bits. The filter can operate with either two's complement or unsigned magnitude values for both the input data and the coefficients, together or separately. The maximum clock speed is 10 MHz. Every clock cycle one coefficient can be replaced by a new value via a 4-bit coefficient port and 3-bit address. For more details the interested reader can refer to [32].

In the TDC1028 chip an alternative to the canonical FIR architecture is used in order to permit pipelining and expansion of the filter length. The equivalent architecture is shown in Fig.4.2 .

In Fig.4.1 are shown the input and output provided for pipelining. The output of a given chip, denoted SUM OUT, can be connected to the input SUM IN of the next chip, thus providing a transfer of the partial sum from section to section. The ports DATA OUT and DATA IN can be connected in the same way.

TDC1028 can also work in bit-slice parallel mode, giving the possibility for increased resolution in the coefficient and the input data, separately or together.

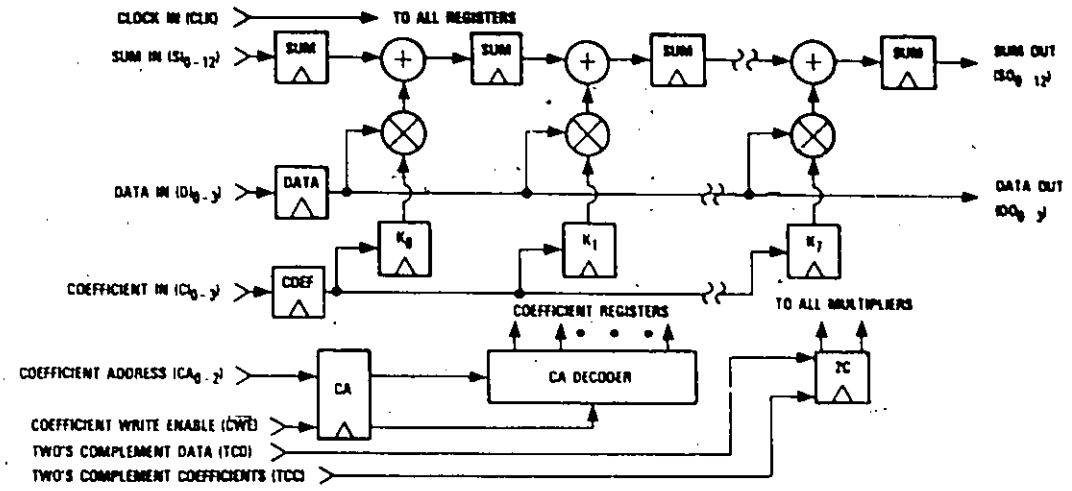


Fig.4.1 Functional block diagram of TDC1028 chip
(Reprinted from VLSI Data Book, TRW Inc., 1986)

4.2 Filter Architecture

The objective of this section is to suggest an efficient hardware structure for a 16-tap 8-bit word length adaptive digital filter, based on TDC1028.

The configuration of the transversal part is suggested by the manufacturer and is shown in Fig.4.3. Both data and coefficients are represented with 8-bit precision (7 plus 1 sign). Word length expansion is done in multiples of four. A separate section is needed for every possible combination of 4-bit groups from the coefficient and data words. More specifically, since both words contain two 4-bit groups, four possible combinations exist, as illustrated in Fig.4.4. The numbers in each square indicate the weight of each product.

In order to expand the filter length to 16 taps, it is necessary to connect another four sections in series with the previous four. This brings the total number of TDC1028 chips to eight. Also additional hardware components are needed for the scaling and the summation.

Basically two solutions come to mind for the implementation of the adaptive

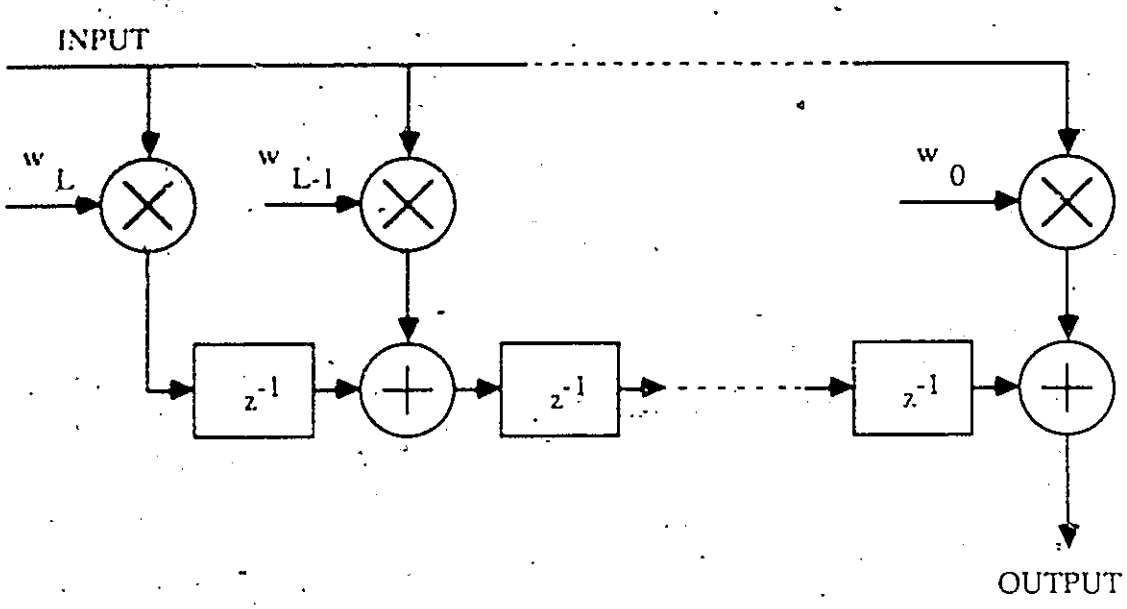


Fig.4.2 Equivalent pipelined FIR architecture

part of the system. The first one is to use a microprocessor and the second one is to use dedicated hardware. The choice of the first solution will result in a very big loss in adaptation speed. The microprocessor must execute many instructions for every coefficient update, including memory fetch and store, multiplications and additions. Thus many clock cycles are necessary to update only one coefficient. Even fast signal processors, like the TMS320 series, cannot successfully cope with the problem. In order to exploit the full capacities of the FIR part, a dedicated hardware implementation of the adaptive part is needed. Moreover, it is even a simpler solution in terms of hardware components and programming. The only advantage of the microprocessor-based realization is the possibility of modifying the adaptive algorithm without hardware modifications by supplying a new program for the microprocessor. But our choice is still the dedicated hardware implementation of the adaptive part.

The basic idea for a complete adaptive system comes from Fig.3.4 in chapter three. In our case, however, advantage can be taken from the fact that the filter sections are 8-taps long, thus giving access to two coefficients at a time every clock cycle. A functional diagram of the proposed system is given in Fig.4.5 .

The delay elements z^{-7} and the storage units (shifters) can be implemented using TDC1011 variable-length shift registers by TRW Inc., which provide programmable lengths between 3 and 18 stages with byte-wide words. The storage units require 8 words of memory space. Every 8 clock cycles the same coefficient comes out of the register, takes part in the update process, gets modified and stored back as last in the queue and at the same time its value is inserted into the FIR part.

The error filtering element can be implemented by a shift right operation and an adder (subtracter). The shifter should be strapped for sign extension in the case of the two's complement mode of operation.

Another shifter can be used for the multiplication by 2μ. The value of μ depends on the particular application, but for purpose of example let find one reasonable value. Suppose that the output noise level after convergence is -10 dB, which is a normal value for 8-bit precision. If the 0 dB power is 0.5 (sine wave with amplitude one), then $\xi_{out\infty} = 0.05$. Substituting that value for ξ_{min} and $B = 7$ in (3.10) gives $\mu > 0.0028$ and $2\mu > 0.057$. It is convenient then to choose $2\mu = 0.0625$, which corresponds to right shift by four positions (division by 16). As a result the whole system uses only two additional 8 by 8 bit multipliers. However, for greater flexibility and precision a true multiplier can be used for the multiplication by 2μ.

Care must be taken to insure the initialization to zero of all filter coefficients in the FIR part and all values in the storage units during initial start-up. Otherwise problems may arise that will lead to nonconvergence.

The nature of the signals x_k and d_k depends on the particular application, as mentioned in chapter one and will not be discussed here. The proper synchronization between the components of the system and many more hardware details are subject to actual implementation details. For that reason they will not be addressed in this text.

The transversal part of the adaptive system was built at the Department of Electrical Engineering, University of Ottawa. The author has tested it and the results

showed proper performance. Work remains to be done on the adaptive part of the system.

The potential applications of the adaptive filter are in the field of noise cancelling, spectrum estimation, linear prediction, etc. for frequencies up to 5 MHz. It can be used in video systems, radar and sonar signal processors, digital communications systems and many other applications.

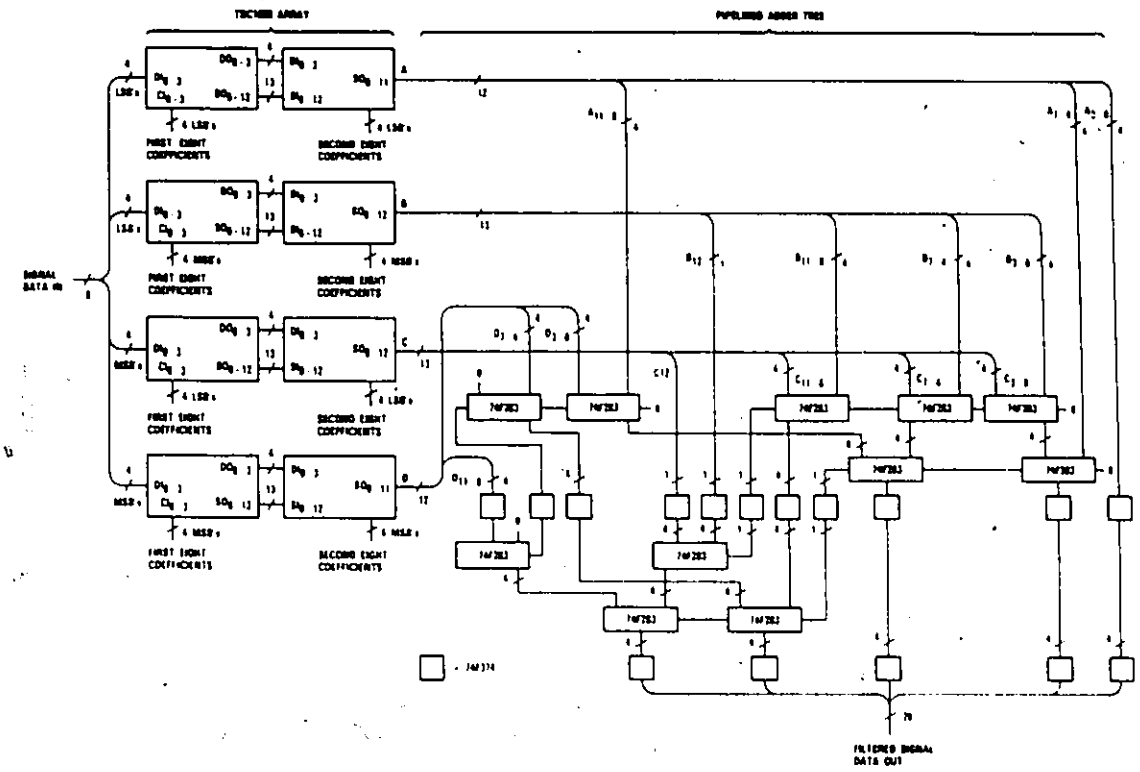


Fig.4.3 Architecture for 16-tap 8-bit word length FIR digital filter using TDC1028 (Reprinted from VLSI Data Book, TRW Inc., 1986)

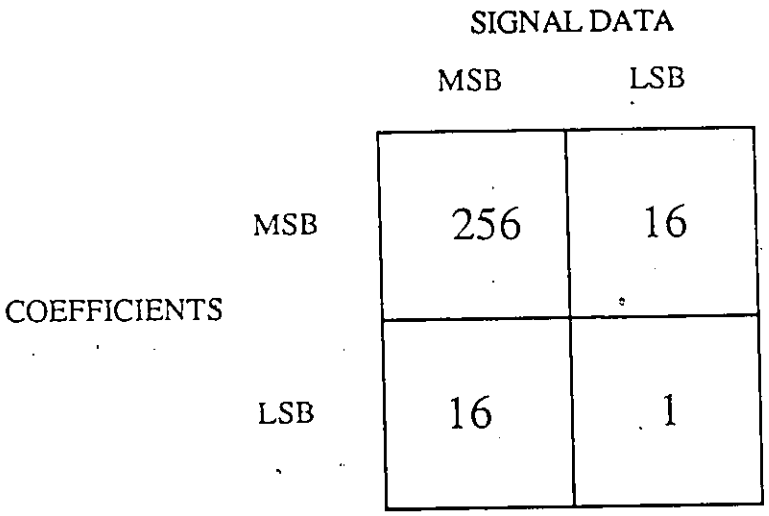


Fig.4.4 Result weights of the sections

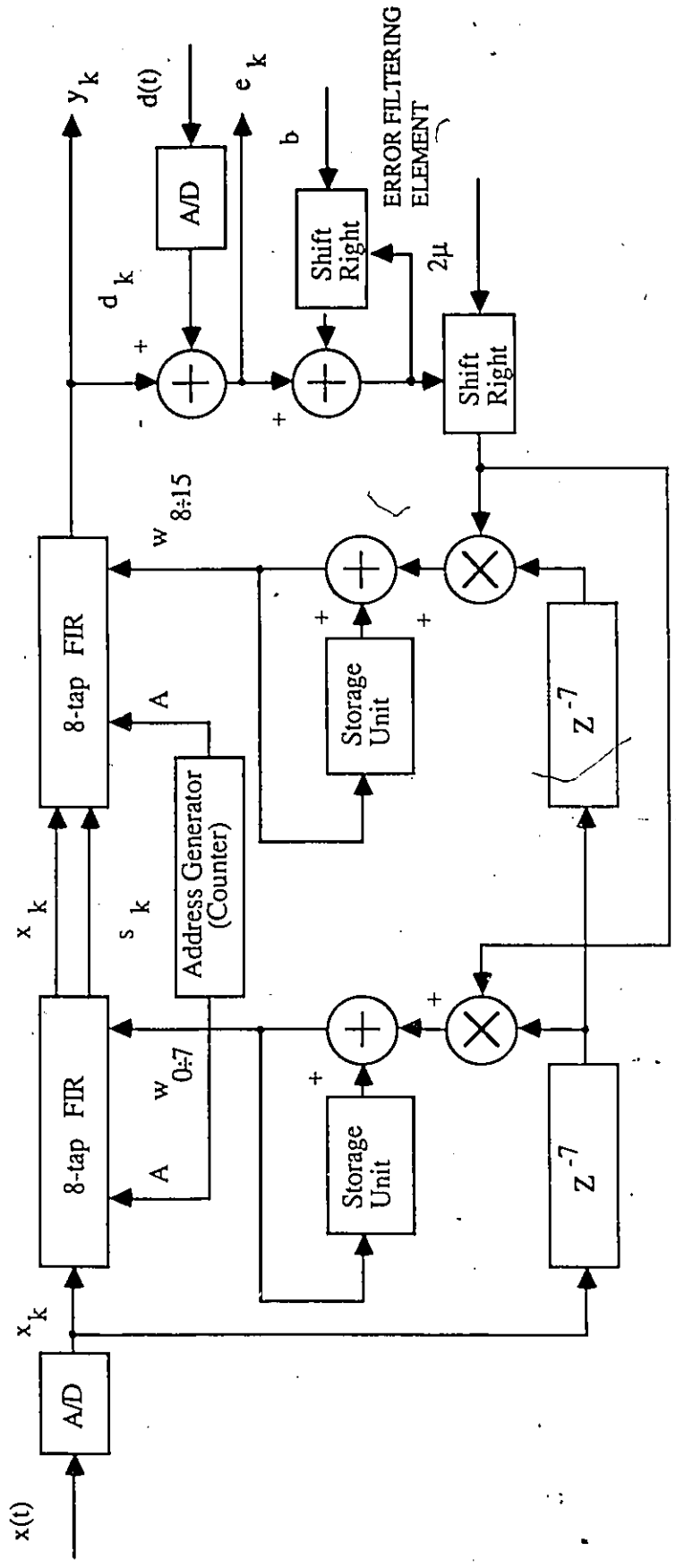


Fig.4.5 16-tap, 8-bit word length FIR adaptive digital filer -- functional diagram

CONCLUSION

The main results from the work done in this thesis are found in the last three chapters.

The first result that deserves attention consists in the attempt to clarify the effect of error signal filtering on the performance of the LMS algorithm. A good insight is achieved into the problem. The modified LMS algorithm, called FLMS algorithm, is shown to be very simple to implement. It requires almost no additional computations. On the other hand, improvement over the LMS algorithm can be achieved only in certain cases and usually additional a priori information is needed. Also the factor of improvement is moderate. Future research can be oriented towards more sophisticated ways of processing the error signal. More work can be done also to find a better explanation of the misadjustment problem.

Another important result is the proposed modified LMS algorithm suitable for VLSI applications. It is proved that they are efficient and easy to implement. More work can be done in this direction too.

The field of applications of adaptive algorithms is very large and all indications are present that it will continue to grow.

APPENDIX A

Derivation of Equation 3.4

The weight vector recursion for the LMS algorithm is given by:

$$\mathbf{W_{k+1}} = \mathbf{W_k} + 2\mu e_k \mathbf{X_k} \qquad (A.1)$$

Let us substitute e_k by its full expression and take the expected value from both sides:

$$E[\mathbf{W_{k+1}}] = E[\mathbf{W_k}] + E[2\mu \mathbf{X_k}(d_k - \mathbf{X_k^T W_k'})]$$

$$= E[\mathbf{W_k}] + 2\mu E[\mathbf{X_k} d_k] - 2\mu E[\mathbf{X_k X_k^T W_k'}] \qquad (A.2)$$

But we know that $E[\mathbf{X_k X_k^T}] = \mathbf{R}$ and $E[\mathbf{X_k} d_k] = \mathbf{P}$. Also, because the input vector $\mathbf{X_k}$ is assumed to be independent of the coefficient vector $\mathbf{W_k}$ for all k, the expected value of the product $\mathbf{X_k X_k^T W_k'}$ can be separated into a product of two expected values, i.e.:

$$E[\mathbf{X_k X_k^T W_k'}] = E[\mathbf{X_k X_k^T}] E[\mathbf{W_k'}] \qquad (A.3)$$

Thus eqn. (A.2) becomes:

$$E[\mathbf{W_{k+1}}] = E[\mathbf{W_k}] + 2\mu \mathbf{P} - 2\mu \mathbf{R} E[\mathbf{W_k}] \qquad (A.4)$$

Now by multiplying the last term in eqn. (A.4) from the right by $E[\mathbf{W_k}] E[\mathbf{W_k^{-1}}]$ (which will not change the equality) and combining some terms we obtain:

$$E[\mathbf{W_{k+1}}] = (\mathbf{I} - 2\mu E[\mathbf{W_k'}] E[\mathbf{W_k^{-1}}]) E[\mathbf{W_k}] + 2\mu \mathbf{P} \qquad (A.5)$$

This result is the same as eqn. (3.4).

REFERENCES AND BIBLIOGRAPHY

[1] T.Tjahjadi, W.Steenaart ,"Adaptive filter realization with a minimum number of multipliers," *IEEE Trans. Circ. Syst.*, vol. 32, No. 3, pp.209-216, March 1985

[2] B.Ahuja, M.Copeland, C.Chan, "A sampled analog MOS LSI adaptive filter ," *IEEE Journal of Solid-State Circuits*, vol. 14, No.1, pp.148-154, Febr . 1979

[3] B.Widrow et al., "Adaptive noise cancelling: principles and applications," *IEEE Proc.*, No.63, pp.1692-1716, 1975

[4] B.Widrow, E. Walach, "On the statistical efficiency of the LMS algorithm with nonstationary inputs", *IEEE Trans. Inf. Theory*, vol. 30, pp.211-221, March 1984

[5] B.Widrow, S.Stearns, "*Adaptive Signal Processing*," Englewood Cliffs, N.J., Prentice-Hall, 1985

[6] M.White et al., "Charge-coupled device (CCD) adaptive discrete analog signal processing," *IEEE Journal Solid-State Circuits*, vol. 14, No.1, pp.132-147 Febr. 1979

[7] C.Cowan et al., "CCD-based adaptive filters: realization and analysis," *IEEE Trans. ASSP*, vol. 29, pp. 220-229, April 1981

[8] A.Gersho, "Adaptive filtering with binary reinforcement," *IEEE Trans. Inf. Th.* vol. 30, No.2, pp. 191-199, March 1984

[9] E.Feuer, E.Weinstein, "Convergence analysis of LMS filters with uncorrelated gaussian data," *IEEE Trans. ASSP-33*, No.1, pp.222-230, Febr.1985

[10] R.Gitlin, E.Weinstein, "On the required tap-weight precision for digitally implemented, adaptive, mean-squared equalizers," *Bell Syst. Techn. Journal*, pp.301-321, Febr. 1979

[11] C.Caraiscos, B.Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. ASSP-32*, No.1, pp. 34-41, Febr. 1984

[12] S.Qureshi, "Adaptive equalization," *IEEE Proceedings*, vol. 73, No.9, pp.1349-1387, Sept. 1985

[13] G.Ungerboeck, "Theory on the speed of convergence in adaptive equalizers," *IBM Journal Res. Develop.*, pp. 546-555, November 1972

[14] J.Mazo,"On the independence theory of equalizer convergence," *Bell Syst. Techn. Journal*, vol. 58, pp. 963-993, May-June 1979

[15] R.Siller Jr., "Multipath Propagation," *IEEE Communic. Magazine*, vol. 22, No.2 pp. 6-15, Febr. 1984

[16] G.Clarc, S.Mitra, S.Parker, "Block implementation of adaptive digital filters," *IEEE Trans. ASSP-29*, No.3, pp. 744-752, June 1981

[17] R.J.Keeler, "Non-optimal convergence of adaptive LMS with uncorrelated data," *Int. Conf. ASSP, 1978*, pp.91-95

[18] M.Larimore, J.Treichler, R.Johnson, "SHARF: an algorithm for adapting IIR digital filters," *IEEE Trans. ASSP-28*, No.4, pp.428-440, August 1980

[19] R.Johnson et al., "SHARF convergence properties," *IEEE Trans. ASSP-29.*,No.3 pp.659-670, June 1981

[20] W.Hodgkiss, "Selecting the length of an adaptive transversal filter," *Int. Conf. ASSP 1978*, pp.96-99,

[21] K.Martin, T.Sun, "Adaptive filters suitable for real-time spectral analysis," *IEEE Trans. Circ. Syst.*, vol. 33, No.2, pp. 218-229, Febr. 1986

[22] R.Harris, D.Chabries, F.Bishop, "A variable step (VS) adaptive filter algorithm," *IEEE Trans. ASSP-34*, No.2, April 1986

[23] J.Proakis, "*Digital Communications,*" Mc.Graw-Hill 1983

[24] A.Oppenheim, R.Shafer, "*Digital Signal Processing,*" Prentice-Hall 1975

[25] P.Fabre, C.Gueguen, "Improvement of the fast recursive least-squares algorithm via normalization: a comparative study," *IEEE Trans. ASSP-34*, No.2, pp.296-309, April 1986

[26] D.Maiwall, H.Kaeser, P.Closs, "An adaptive equalizer with significantly reduced number of operations," *Int. Conf. ASSP 1978*, pp.100-104

[27] J.Cioffi, "The block-processing FTF adaptive algorithm," *IEEE Trans. ASSP-34*, No.1, pp. 77-95, Febr. 1986

[28] N.Bershad, "On the optimum data nonlinearity in LMS adaptation," *IEEE Trans. ASSP-34*, No.1, pp. 69-77, Febr. 1986

[29] D.Duttweiler, "Adaptive filter performance with nonlinearities in the correlation multiplier," *IEEE Trans. ASSP-30*, No.4, pp. 578-586, August 1982

[30] C.Cowan et al.," Implementation of a 64-point adaptive filter using an analogue CCD programmable filter," *Electronics Letters*, vol. 15, pp. 568-569, 1978

[31] C.Cowan et al., "Noise cancelling and inverse filtering using compact, high performance, CCD adaptive filter," *Electronics Letters*, vol. 15, pp. 35-37, 1979

[32] "*VLSI Data Book,*" TRW Inc., 1986

[33] D.Falconer, L.Ljung, "Application of fast Kalman estimation to adaptive equalization," *IEEE Trans. Communic.* vol.26, pp.1439-1446, October 1978