Floating Point Roundoff Error Analysis of the Adaptive Exponential Sliding Window RLS Algorithm for Time-Varying Systems in the Presence of Noise

by

S.H. Ardalan

Center for Communications and Signal Processing
Department of Electrical Computer Engineering
North Carolina State University

February 1986

CCSP-TR-86/7

Table of Contents

Floating Point Roundoff Error Analysis of the Adaptive Exponential Sliding Window RLS Algorithm for Time-Varying Systems in the Presence of Noise

by
S. H. Ardalan

Abstract

A floating point error analysis of the Recursive Least Squares and Least Mean Squares (LMS) algorithm is presented. Both the prewindowed growing memory RLS algorithm ($\lambda=1$) for stationary systems and the exponential sliding window RLS algorithm ($\lambda<1$) for time varying systems are studied. For both algorithms the expression for the mean square prediction error and the expected value of the weight error vector norm are derived in terms of the variance of the floating point noise sources. The results point to a trade off in the choice of the forgetting factor, λ. In order to reduce the effects of addivitive noise and the floating point noise due to the inner product calculation of the desired signal, λ must be chosen close to one. One the other hand, the floating point noise due to floating point additon in the weight vector update recursion increases as $\lambda \rightarrow 1$. Floating point errors in the calculation of the wieght vector correction term, however, do not affect the steady state error and have transient effect. For the prewindowed growing memory RLS algorithm, exponential divergence may occur due to errors in the floating point addition in the weight vector update recursion. Conditions for weight vector updating termination are also presented for stationary systems. The results for the LMS algorithm show that the excess mean square error due to floating point arithmetic increases inversely to loop gain for errors introduced by the summation in the weight vector recursion. The calculation of the desired signal prediction and prediction error lead to an additive noise term as in the RLS algorithm. Simulations are presented which confirm the theoretical findings of the paper.

1.1 Introduction

The Recursive Least Squares (RLS) algorithm has found wide application in adaptive filtering problems [1,2,3]. In particular, the algorithm offers very fast convergence and tracking capability. Unlike the LMS algorithm, the convergence of the RLS algorithms is independent of the signal statistics, i. e., eigenvalue spread of the signal covariance matrix. This has lead to the application of the RLS algorithm to many adaptive filtering problems. However, finite wordlength effects on the digital implementation of this algorithm have not been thoroughly analyzed until recently. In [4] the error propagation of the prewindowed growing memory ($\lambda = 1$) and the the exponential sliding window ($\lambda < 1$) RLS algorithm for time varying systems was analyzed. In the paper, the propagation of a single error in the continuous use of the various algorithms was studied. It was shown that the effects of a single error decays exponentially for the conventional RLS algorithm, where the base of the decay is equal to the forgetting factor λ. As pointed out in [4], however, little work has appeared in the literature concerning finite wordlength effects on the performance of adaptive RLS algorithms. In particular, finite precision arithmetic produces perturbations on the estimation of the system impulse response at each iteration. Furthermore, it is of interest to analyze the effects of each operation on the performance of various algorithms. As will be shown in this paper, certain finite precision errors tend to destabilize the algorithm while some only cause a bias in the estimates. On the other hand, some operations do not effect steady state performance at all. For these reasons, the analysis of the effects of all errors introduced at each iteration is of interest.

Recently a fixed point and floating point error analysis of the Least Mean Squares (LMS) algorithm was presented in [7]. In [8] a fixed point roundoff error analysis of the prewindowed growing memory RLS algorithm was presented. In that paper, a closed form solution to the excess mean square error due to fixed point arithmetic is derived. It was found that the mean square prediction error diverged linearly with the number of iterations prior to the freezing of adaptation (due to the Kalman gain vector approaching zero).

The result paralleled that of [7] where it was shown that the roundoff error increases inversely to the adaptation loop gain for the LMS algorithm. In this paper, a floating point round off error analysis of the RLS algorithm for $\lambda = 1$ and $\lambda < 1$ is presented. The results are also extended to the LMS algorithm.

1.2 The RLS Algorithm

The RLS algorithm can be summarized as follows: The desired signal d(n) is predicted from the input samples by convolving the input samples with a weight vector. The prediction error is then calculated as the difference between this signal and a measurement of the desired signal which may be corrupted by noise. The prediction error is used to update the weight vector to minimize the accumulated sum of the square of the residual error. The weight vector that minimizes this criterion at each iteration is written recursively. This leads to the Kalman gain vector which when multiplied by the scalar prediction error forms the weight vector update. Therefore, for the purposes of this paper, the algorithm involves two primary calculations. The calculation of the prediction of the desired signal d(n) through an inner product of the weight vector and the vector of delayed elements of the input samples; and the calculation of the weight vector update. The weight vector update involves the calculation of the weight vector correction through the multiplication of the Kalman gain vector and the scalar prediction error and adding the correction term to the current estimate of the weight vector.

For the purposes of analysis we assume that the Kalman gain is calculated with infinite precision and a floating point quantized version is available. This assumption is based on the fact that the Kalman gain can be calculated using different algorithms and that as a preliminary step we are attempting to isolate the causes of error in the floating point implementation of the algorithm.

As in the case with the floating point error analysis of fixed digital filters [9,10] and the LMS algorithm in [7] the errors introduced by floating point operations are modeled as follows: For summation $fl(x+y)=(x+y)(1+\epsilon)$ and for multiplication $fl(xy)=xy(1+\delta)$. The relative error terms ϵ and δ are zero mean white independent random numbers. They are uncorrelated with $x+y$ and xy. Using the above models the errors introduced by floating point implementation are incorporated into the algorithm.

2.1 Brief Summary of Results.

For both algorithms the expression for the mean square prediction error and the expected value of the weight error vector norm are derived in terms of the variance of the floating point noise sources. The weight error vector is defined as the vector difference between the the current estimate and the optimal weight vector or true impulse response of the system. Hence, if the mean value of the weight error vector norm converges to zero, the algorithm converges to the optimal solution. The results point to a trade off in the choice

of the forgetting factor, λ. In order to reduce the effects of additive noise and the floating point noise due to the inner product calculation of the desired signal, λ must be chosen close to one. On the other hand, the floating point noise due to floating point addition in the weight vector update recursion increases as $\lambda \to 1$. Floating point errors in the calculation of the weight vector correction term, however, do not affect the steady state error and have a transient effect. For the prewindowed growing memory RLS algorithm, exponential divergence may occur due to errors in the floating point addition in the weight vector update recursion. However, if λ is decreased the errors due to floating point operations in the weight vector update addition decay exponentially and a more stable algorithm results. This is in agreement with [4]. However, additive noise and floating point noise in the calculation of the desired signal will produce a bias in the estimate of the optimal weight vector for $\lambda < 1$. Moreover, if λ is close to one but not equal to one we predict that divergence may occur.

The results for the LMS algorithm show that the excess mean error due to floating point arithmetic increases inversely to loop gain for errors introduced by the summation in the weight vector recursion. The calculation of the desired signal prediction and prediction error lead to an additive noise term as in the RLS algorithm. Significantly, as in the RLS algorithm, the precision used in the calculation of the weight vector correction term does not effect steady state performance.

2.2 Paper Organization.

The paper is organized as follows: In the next section the infinite precision RLS algorithm is presented, followed by a review of floating point error models. In section 3, the floating point errors are incorporated into the RLS algorithm. It is shown that the mean square prediction error is related to the mean value of the weight error vector norm which is equal to the trace of the covariance matrix of the weight error vector. In section 6, a summary of the expressions for the weight error covariance matrix is presented. In section 7, the conditions for which updating may cease are analysed for stationary systems. A complete derivation of the results, summarized in section 6, appears in section 8. In section 9, the results are extended to the LMS algorithm. Finally, numerical simulations of floating point RLS and LMS algorithms are presented which validate the results of the paper.

3. Recursive Least Squares (RLS) Algorithm: Infinite Precision

The Recursive Least Squares (RLS) algorithm is used to solve the system identification problem as shown in Figure 1. In the figure x(n) is the input sample to the system and z(n) represents the output of the system corrupted by additive noise v(n). The system is described by its impulse response w_i^* i=0,...,N-1 which is convolved with the input to produce the output signal d(n) as follows:

$$d(n) = \sum_{i=0}^{N-1} w_i^* x(n-i) = \underline{w}^{*T}\underline{x}(n) \tag{3.1}$$

The measured output is corrupted by additive noise v(n),

$$z(n) = d(n) + v(n)$$

Here we assume that the impulse response $\underline{w}^*$ has insignificant terms beyond N samples. The notation $\underline{x}(n)$ signifies the vector of the last N samples while $\underline{w}^*$ is a vector representing the N values of the impulse response:

$$\underline{x}(n)=[x(n),\ x(n-1),\ \cdots\ x(n-N+1)]^T$$

$$\underline{w}^* = [w_0^*, w_1^*, \cdots\ w_{N-1}^*]^T \tag{3.2}$$

The RLS algorithm produces the weight vector $\underline{w}(n)$ which is an estimate at time n of the true impulse response $\underline{w}^*$. The RLS algorithm operates on $\underline{x}(n)$ and z(n) to obtain an estimate of $\underline{w}^*$ based on the previous estimate as follows:

$$\underline{w}(n) = \underline{w}(n-1) + \underline{K}(n)\, e(n) \tag{3.3}$$

where

$$e(n) = z(n) - \hat{d}(n) \tag{3.4}$$

is the prediction error and

$$\hat{d}(n)=\underline{w}^T(n-1)\underline{x}(n) \tag{3.5}$$

is the prediction of d(n) based on N previous samples of $\underline{x}(n)$. The Kalman gain can be shown [1] to be

$$\underline{K}(n) = \left[\sum_{i=0}^{n} \lambda^{n-i}\underline{x}(i)\underline{x}^T(i)\right]^{-1}\underline{x}(n) \tag{3.6}$$

The derivation leading to (3.3) with the Kalman gain given by (3.6) is based on minimizing the accumulated sum of the square of the residual errors up to time n with respect

to the weight vector $\underline{w}$ (n) and writing $\underline{w}(n)$ recursively. For the prewindowed growing memory RLS algorithm $\lambda = 1$. However for time varying systems in which the impulse response $\underline{w}^*$ may change, $\lambda < 1$ is chosen. In this case the algorithm is referred to as the exponential sliding window RLS algorithm. For $\lambda < 1$ old samples lose their significance in the calculation of $\underline{w}(n)$.

It has been shown that in the infinite precision RLS algorithm $\underline{w}(n)$ converges to $\underline{w}^*$ when the additive noise source v(n) is a white random process and $\lambda = 1$.

The issue addressed in this paper concerns this convergence of $\underline{w}(n)$ toward $\underline{w}^*$ when the RLS algorithm described in equations (3.3-3.5) do not have infinite precision arithmetic but are performed on real machines using floating point arithmetic.

In order to evaluate this convergence, we define

$$\underline{\theta}'(n) = \underline{w}'(n) - \underline{w}^* \tag{3.7}$$

where the primes denote quantities determined using floating point arithmetic. Thus, $\underline{\theta}'(n)$ is the weight error vector denoting the difference between the floating point estimate $\underline{w}'(n)$ and the true impulse response $\underline{w}^*$

In what follows we show that $\underline{\theta}'(n)$ can be expressed as the sum of two terms:

$$\underline{\theta}'(n) = \underline{\chi}(n) + \underline{\psi}(n) \tag{3.8}$$

Here, $\underline{\psi}(n)$ in the absence of additive noise is the excess floating point error vector which would not exist if infinite precision arithmetic could be used. In the absence of floating point errors, $\underline{\chi}(n)$ would be the only component of the weight error vector and it has been shown [1] that $\underline{\chi}(n)$ would approach $\underline{0}$ as n gets larger. However, floating point arithmetic also produces errors in $\underline{\chi}(n)$ so that it does not approach $\underline{0}$ as n increases. Thus both $\underline{\chi}(n)$ and $\underline{\psi}(n)$ contribute to the estimation error.

In the next section we describe the floating point round off error models and include them in the floating point RLS algorithm.

4. Floating Point Roundoff Error Models

In this paper the errors introduced by floating point operations are modeled as follows: For multiplication

$$fl[x \cdot y] = x \cdot y[1 - \epsilon] \tag{4.1}$$

were ϵ is the relative error and is modeled as a zero mean random variable independent

of x,y and (x·y). It has been found [6.8] that $\sigma_\epsilon^2 = E(\epsilon^2) \sim (0.18)2^{-2B}$ where B is the number of bits used to represent the mantissa.
For addition

$$fl[x+y] = (x+y)\,[1+\delta] \quad (4.2)$$

where δ is the relative error modeled as a zero mean random variable with variance σ_δ^2. The actual distribution of δ and ϵ are not important in what follows as long as the assumption that each error term is a zero mean, independent in time, random variables is valid.

Note that in floating point operations, addition and multiplication both introduce errors, whereas, in fixed point operations (assuming no overflow) only multiplication introduces error.

5. Floating Point Implementation of the RLS Algorithm

In this section we incorporate the errors introduced by the floating point implementation of the RLS algorithm into the algorithm.

5.1 Desired Signal Prediction

Consider the calculation of the prediction of the desired signal through the inner product (3.5). Let $\rho(n)$ denote the error introduced by floating point operations in the calculation of the inner product. Then,

$$\hat{d}'(n) = \underline{x}^T(n)\underline{w}'(n-1) + \rho(n) \quad (5.1)$$

For the above equation $\hat{d}'(n)$ is the prediction of the desired signal using the floating point RLS algorithm and $\underline{w}'(n)$ is the floating point weight vector. The primes denote the floating point variables as opposed to the infinite precision variables of (3.1-3.6). Denoting the floating point errors produced in the summation and multiplication operations in the calculation of $\hat{d}'(n)$ by $\epsilon_i(n)$ and $v_i(n)$ respectively, then,

$$\rho(n) = x_0(n)w'(n)\epsilon_0(n) + \cdots + x_{N-1}(n)w'_{N-1}(n)\epsilon_{N-1}(n) + \tag{5.2}$$

$$[x_0(n)w'_0(n)+x_1(n)w'_1(n)]\upsilon_1(n) + \ldots +$$

$$[x_0(n)w'_0(n) + \ldots + x_{N-1}(n)w'_{N-1}(n)]\upsilon_{N-1}(n)$$

We assume that $\epsilon_i(n)$ and $\upsilon_i(n)$, i=0,...,N-1 are zero mean white random sequences and therefore $\rho(n)$ is a zero mean white random sequence. We have [4],

$$\sigma_\rho^2 = E[\rho^2(n)] = \sigma_\epsilon^2 \sum_{k=0}^{N-1} \sigma_x^2 \, E[w'^2_k(n)] + \tag{5.3}$$

$$\sigma_\upsilon^2\{E[x_0(n)w'_0(n)+x_1(n)w'_1(n)]^2 +$$

$$\ldots + E[x(n)w'_0(n) + \ldots + x_{N-1}(n)]^2\}$$

The expression (5.3) for σ_ρ^2 can be written in terms of the autocorrelation matrix R_x and the optimal (Wiener) vector $\underline{w}^*$ as in [4]. But this assumes that $\underline{w}'(n)$ converges. We will show, however, that $\underline{w}'(n)$ might diverge and thus σ_ρ^2 will diverge also due to floating point errors introduced in the weight vector update recursion. Nevertheless, the point to consider is that prior to divergence σ_ρ^2 depends on the floating point noise sources and the statistics of x(n) and $\underline{w}'(n)$.

5.2 Prediction Error Calculation

Next consider the floating point error introduced in the computation of the prediction error (3.4). We have,

$$e'(n) = [z(n) - \hat{d}'(n)][1+\delta(n)] \; . \tag{5.4}$$

where we assume that the relative error $\delta(n)$ is a zero mean white random sequence. If we substitute (3.1) for z(n) and (5.1) for $\hat{d}'(n)$ then we can write

$$e'(n) = \underline{x}^T(n)[\underline{w}^* - \underline{w}'(n-1)] + \delta(n)\underline{x}^T(n)[\underline{w}^* - \underline{w}'(n-1)] - \rho(n) - v(n) \tag{5.5}$$

where we have neglected the second order noise term $\rho(n)\delta(n)$.

5.3 Weight Vector Update Recursion

Once the prediction error is computed, the weight vector is updated by the recursive

equation (3.3). Assuming that the floating point quantized Kalman gain vector elements are available at each step, the recursive update equation for each weight vector element including floating point noise sources is

$$w_i'(n) = \{w_i'(n-1) + K_i(n)e'(n)[1+\mu_i(n)]\}[1+\alpha_i(n)] \tag{5.6}$$

where $\mu_i(n)$ and $\alpha_i(n)$ are the floating point noise sources due to multiplication and addition respectively. Expanding (5.6) and neglecting second order noise terms we obtain,

$$w_i'(n) = w_i'(n-1) + K_i(n)e'(n) + w_i'(n-1)\alpha_i(n) + K_i(n)e'(n)[\alpha_i(n)+\mu_i(n)] \tag{5.7}$$

If we define the diagonal matrices,

$$\alpha_{NN}(n)=\mathrm{diag}[\alpha_i(n)] \tag{5.8}$$

$$\phi_{NN}(n)=\mathrm{diag}[\alpha_i(n)+\mu_i(n)] \tag{5.9}$$

then the weight vector update equation becomes

$$\underline{w}'(n) = \underline{w}'(n-1) + \underline{K}(n)e'(n) + \alpha_{NN}(n)\underline{w}'(n-1) + \phi_{NN}(n)\underline{K}(n)e'(n) \tag{5.10}$$

Substituting for $e'(n)$ from (5.6) and neglecting second order noise terms we obtain,

$$\underline{w}'(n) = \underline{w}'(n-1) - \underline{K}(n)\underline{x}^T(n)[\underline{w}'(n-1) - \underline{w}^*] - \underline{K}(n)\eta(n) -$$

$$\delta(n)\underline{K}(n)\underline{x}^T(n)[\underline{w}'(n-1) - \underline{w}^*] + \alpha_{NN}(n)\underline{w}'(n-1) +$$

$$\phi_{NN}(n)\underline{K}(n)\underline{x}^T(n)[\underline{w}'(n-1) - \underline{w}^*] \tag{5.11}$$

where we have defined,

$$\eta(n) = v(n) - p(n) . \tag{5.12}$$

It is convenient to write the recursion (5.11) in terms of the weight error vector defined in

(3.7). Therefore, subtracting $\underline{w}^*$ from both sides of (5.12) and substituting (3.7) we obtain,

$$\underline{\theta}'(n) = \underline{\theta}'(n-1) - \underline{K}(n)\underline{x}^T(n)\underline{\theta}'(n-1) - \delta(n)\underline{K}(n)\underline{x}^T(n)\underline{\theta}'(n-1) +$$

$$\phi_{NN}(n)\underline{K}(n)\underline{x}^T(n)\underline{\theta}'(n-1) + \alpha_{NN}(n)[\underline{\theta}'(n-1) + \underline{w}^*] - \underline{K}(n)\eta(n) \qquad (5.13)$$

Collecting terms,

$$\underline{\theta}'(n) = [I - \underline{K}(n)\underline{x}^T(n) - \tilde{\phi}_{NN}(n)\underline{K}(n)\underline{x}^T(n) + \alpha_{NN}(n)]\underline{\theta}'(n-1) + \alpha_{NN}(n)\underline{w}^* - \eta(n)\underline{K}(n) \qquad (5.14)$$

where

$$\tilde{\phi}_{NN}(n) = \phi_{NN}(n) + \delta(n)I \qquad (5.16)$$

Define the matrix

$$\beta_{NN}(n) = \alpha_{NN}(n) - \phi_{NN}(n)\underline{K}(n)\underline{x}^T(n) \qquad (5.17)$$

and the vector

$$\xi(n) = \alpha_{NN}(n)\underline{w}^* - \eta(n)\underline{K}(n) \qquad (5.18)$$

Then substituting into (5.15) we get

$$\underline{\theta}'(n) = [I - \underline{K}(n)\underline{x}^T(n) + \beta_{NN}(n)]\underline{\theta}'(n-1) + \xi(n) \qquad (5.19)$$

Equation (5.19) represents the recursive update equation of the weight error vector in terms of the noise sources due to floating point operations. For infinite precision, and if $v(n) = 0$,

$$\beta_{NN}(n) - 0$$
$$\xi(n) - 0$$

and

$$\underline{\theta}'(n) - \underline{\theta}(n) = [I - \underline{K}(n)\underline{x}^T(n)]\underline{\theta}(n-1) \qquad (5.20)$$

which is the exact recursion for the infinite precision update equation of the weight error vector.

5.3 Weight Error Vector Norm and Mean Square Prediction Error

We can express the floating point prediction error (5.5) using (3.7) and (5.12) as,

$$e'(n) = \left(-\underline{x}^T(n)\underline{\theta}'(n)+\eta(n)\right)[1 + \delta(n)] \tag{5.21}$$

Define the Mean Square Prediction Error:

$$\sigma_e^2(n) = E\{e'^2(n)\} = \tag{5.22}$$

$$E\{\underline{x}^T(n)\underline{\theta}'(n)\underline{\theta}'^T(n)\underline{x}(n)\}[1 + \sigma_\delta^2] + \sigma_\eta^2$$

Or,

$$\sigma_e^2(n) = \mathrm{Trace}[R_x R_\theta(n)](1+\sigma_\delta^2) + \sigma_\eta^2 \tag{5.23}$$

where,

$$R_x = E\{\underline{x}(n)\underline{x}^T(n)\} \tag{5.24}$$

and,

$$R_\theta(n) = E\{\underline{\theta}'(n)\underline{\theta}'^T(n)\} \tag{5.25}$$

Noting that,

$$\mathrm{Trace}[R_\theta(n)] = E\{||\underline{\theta}'(n)||^2\} \tag{5.26}$$

and

$$\mathrm{Trace}[R_x] = N\sigma_x^2 \tag{5.27}$$

we can write (5.23) as follows,

$$\sigma_e^2(n) = \sigma_x^2 E\{||\underline{\theta}'(n)||^2\}(1 + \sigma_\delta^2) + \sigma_\eta^2 \tag{5.28}$$

From (5.28) it is clear that the mean square prediction error is related to the expected value of the norm of the weight error vector $\underline{\theta}'(n)$. From (5.26) the expected value of this norm is related to the covariance matrix of the weight error vector. In the next section we present a summary of expressions derived in detail in section 9 for the covarience matrix of the floating point weight error vector.

6. Summary of Results

As was pointed out in section 3, the floating point weight error vector can be expressed as the sum of two vectors,

$$\underline{\theta}'(n) = \underline{\chi}(n) + \underline{\psi}(n) \tag{6.1}$$

The covariance of the weight error vector is thus,

$$R_{\theta}(n) = R_{\chi}(n) + R_{\psi}(n) \tag{6.2}$$

where we have neglected the cross correlation between $\underline{\chi}(n)$ and $\underline{\psi}(n)$ (see section 8).

In the following paragraphs we present expressions for $R_{\chi}(n)$ and $R_{\psi}(n)$ for the stationary and time varying RLS algorithms. Using equations (5.26) and (5.28) the mean square prediction error and the expected value of the weight error vector norm can be computed by substituting for R_{θ}.

In the expressions to follow,

$$\sigma_{\alpha}^2 = E\{\alpha_i^2(n)\}$$

is the variance of the floating point noise due to addition in the weight vector update recursion (5.6). The variance,

$$\sigma_{\eta}^2 = \sigma_v^2 + \sigma_{\rho}^2$$

is the sum of the variance of the additive noise v(n) and the noise due to the vector inner product calculation of $\hat{d}'(n)$ (5.1). In the derivation leading to the results, the input sample x(n) is assumed to be a zero mean, independent in time, white Gaussian random variable with variance σ_x^2.

Prewindowed Growing Memory RLS Algorithm ($\lambda = 1$)

For this case the following results are obtained,

$$R_{ij}(n) = \frac{e^{n\sigma_\alpha^2}}{n^2} \left\{ \frac{1}{\sigma_\alpha^4} \text{diag}\,[w_i^{*2}] + \frac{1}{\sigma_\alpha^2} \frac{\sigma_\eta^2}{\sigma_x^2} \right\} \tag{6.3}$$

$$R_\chi(n) = \underline{\theta}(0)\underline{\theta}^T(0)\frac{1}{n^2} e^{\frac{1}{n}} \cdot e^{-2\gamma_c} e^{n\sigma_\alpha^2} \tag{6.4}$$

Assuming updating does not stop, an examination of (6.3-6.4) shows that the RLS algorithm diverges as the number of iterations (n) becomes large. This divergence is exponential with respect to n. It is observed that the *divergence terms are due to floating point errors in the calculations of the weight vector update* (3.3), i.e., $\alpha_i(n)$. Also, *the steady state solution is independent of the floating point error introduced in the calculation of the weight vector correction term* . i.e., $\mu_i(n)$. Hence, no steady state degradation results if low precision is used in the calculation of these terms, i.e., $\underline{K}(n)e(n)$ in (3.3).

If updating stops after n_T iterations then the following expression holds for $R_{ij}(n)$ provided n is large but small compared to $\frac{1}{\sigma_\alpha^2}$ such that $e^{n\sigma_\alpha^2} \approx 1$.

$$R_{ij}(n) = \left\{ \frac{n}{3}\sigma_\alpha^2 \text{ diag}\,[w_i^{*2}] + \frac{1}{n} \frac{\sigma_\eta^2}{\sigma_x^2} \right\} \tag{6.5}$$

Assuming no errors in the calculation of the update for $\underline{w}(n)$, and $\sigma_\alpha^2 \to 0$ it is shown in section 8 that

$$\lim_{\sigma_\alpha^2 \to 0} R_\theta(n) = \underline{\theta}(0)\underline{\theta}^T(n)\frac{1}{n^2}e^{-2\gamma_c} + \frac{1}{n}\frac{\sigma_\eta^2}{\sigma_x^2}I \tag{6.6}$$

where $\gamma_c = 0.57721$ is Eulers constant.

In other words, the errors due to the additive noise and inner product calculation of the prediction of the desired signal are averaged out by the algorithm. They will not cause a bias in $\underline{w}(n)$ as it converges to $\underline{w}^*$. However, they will contribute a noise term with variance σ_η^2 to the prediction error as shown in (5.28). Notice that the steady state RLS solution is independent of the initial condition. However, with floating point errors the steady state result depends on the initial condition.

Exponentially Windowed RLS Algorithm ($\lambda > 1$)

In the case of the exponential sliding window RLS algorithm the following results are obtained:

$$R_{\psi}(n) = C(M) + \sigma_{\alpha}^2 \frac{e^{(n-M)\upsilon} - e^{-\upsilon}}{1-e^{-\upsilon}} \text{diag}\,[w_i^{*2}] + (1-\lambda)^2 \frac{\sigma_{\eta}^2}{\sigma_x^2} \frac{e^{(n-M)\upsilon} - e^{-\upsilon}}{1-e^{-\upsilon}} I \quad (6.7)$$

$$R_{\chi}(n) = \underline{\theta}(0)\underline{\theta}^T(0)\,(1-\lambda)^2 e^{n\upsilon} \quad (6.8)$$

where,

$$C(M) = \sum_{i=1}^{M-1} \{\sigma_{\alpha}^2 \text{diag}\,[w_i^{*2}] + \frac{1}{i^2} \frac{\sigma_{\eta}^2}{\sigma_x^2}\} e^{(n-i)\upsilon} (1-\lambda)^{2\lambda^{i-1}} \quad (6.9)$$

and

$$\upsilon = \sigma_{\alpha}^2 - 2(1-\lambda) \quad (6.10)$$

The matrix C(M) collects the terms for which $i<M$, where the index M is defined as the index for which $\lambda^M << 1$, or $M > \frac{1}{1-\lambda}$

An examination of the results reveals a tradeoff in the choice of the forgetting factor λ. In order to reduce the contribution of the additive noise and vector inner product floating point noise in the calculation of $d'(n)$ to the estimation of $\underline{w}^*$, we must choose λ close to one. On the other hand in order to track a time varying system λ must be smaller than one. Furthermore, as λ becomes close to one, υ in (6.9) may become positive. This in turn leads to exponential divergence or an amplification of the estimation error due to the floating point addition in the weight error vector update as predicted in (6.6) and (6.7).

If we choose λ such that $\upsilon<0$, then the steady state weight error vector covariance matrix becomes.

$$R_{\theta}(n) = C(M) + \sigma_{\alpha}^2 \frac{e^{-\upsilon}}{e^{-\upsilon} - 1} \text{diag}\,[w_i^{*2}] + (1-\lambda)^2 \frac{\sigma_{\eta}^2}{\sigma_x^2} \frac{e^{-\upsilon}}{e^{-\upsilon} - 1} I \quad (6.11)$$

If $\sigma_{\alpha}^2 << 2(1-\lambda)$ then (6.11) becomes,

$$R_{\theta}(n) = C(M) + \sigma_{\alpha}^2 \frac{e^{-\upsilon}}{2(1-\lambda)} \text{diag}\,[w_i^{*2}] + (1-\lambda)^2 \frac{\sigma_{\eta}^2}{\sigma_x^2} \frac{e^{-\upsilon}}{2} I \quad (6.12)$$

The above expression clearly shows that increasing λ ($\lambda \rightarrow 1$), in order to reduce the mean square error due to additive noise amplifies the floating point error due to the addition in the weight vector update recursion (σ_{α}^{2}). Decreasing λ for tracking time varying systems improves stability (in the case of the conventional RLS algorithm) and reduces the floating point error due to this operation but causes an increase in the bias due to additive noise.

In the next section conditions are derived for the early termination of weight vector updating in the floating point RLS algorithm for stationary systems. The conditions depend on the forgetting factor, the additive noise variance, the size of the weight vector component's mantissa and the true weight vector $\underline{w}^{*}$.

7. Termination of Updating for Stationary Systems

An examination of the weight vector update recursion (3.3) reveals that the condition for the ith component of the weight vector not to be updated is,

$$|K_i(n)e(n)| < 2^{-B_\alpha - 1}|w_i(n)| \tag{7.1}$$

where B_α is the number of bits used to represent the mantissa of the weights. This expression is similar to the condition presented in [7] for the floating point LMS algorithm. Since (see Appendix A),

$$E\{||\underline{K}(n)||^2\} \approx \left(\frac{1-\lambda}{1-\lambda^k}\right)^2 \frac{N}{\sigma_x^2} \tag{7.2}$$

then,

$$E\{K_i^2(n)\} \approx \left(\frac{1-\lambda}{1-\lambda^k}\right)^2 \frac{1}{\sigma_x^2} \tag{7.3}$$

Termination of updating results after the weights approach the true weights ($w_i(n) \approx w_i^*$) and the major component of the prediction error is the noise source $\eta(n)$ consisting of additive noise and floating point roundoff error noise. Hence, $\sigma_e^2 \approx \sigma_\eta^2$. Therefore, the condition (7.1) leads to,

$$\left(\frac{1-\lambda}{1-\lambda^k}\right)^2 \sigma_\eta^2 < \frac{2^{-2B_\alpha}}{4}|w_i^*|^2 \tag{7.4}$$

For $\lambda = 1$ updating terminates after n_T iterations where,

$$n_T \approx \frac{2\sigma_\eta}{2^{-B_\alpha}|w_i^*(\min)|} \tag{7.5}$$

For $\lambda < 1$, n_T is,

$$n_T \approx \frac{\ln\left[1 - \dfrac{2(1-\lambda)\sigma_\eta}{2^{-B_\alpha}|w_i^*(\min)|}\right]}{\ln\lambda} \tag{7.6}$$

Note that updating never terminates if

$$\lambda < 1 - \frac{2^{-B_\alpha - 1}|w_i^*(\min)|}{\sigma_\eta} \tag{7.7}$$

From the (7.5), the prewindowed growing memory RLS algorithm will terminate updating when the additive noise v(n) is very small. In fact, it is possible that the algorithm will terminate after the first N iterations if $\sigma_\eta^2 \approx 0$. In this case,

$$E\{||\underline{\theta}'||^2\} = ||\underline{w}^*||^2\sigma_\alpha^2 \tag{7.8}$$

If σ_η^2 is large and B_α is also large, then n_T can be large enough such that the steady state results of this paper are valid. Furthermore, in many practical cases $w_i^* \approx 0$, for some i, in which case updating may never terminate for the ith component of the weight vector. In cases where n_T is small compared to $\frac{1}{\sigma_\alpha^2}$, equation (6.5) must be used for the steady state solution. Similar arguments hold for cases where $\lambda < 1$ and (7.7) is not satisfied. Otherwise, for λ small enough, updating will continue and the steady state solutions are valid.

6. Derivation of Floating Point Roundoff Effects on the RLS Algorithm

If we write the recursion (5.19) for $\underline{\theta}'(n)$ in terms of the initial condition $\underline{\theta}'(0)$, then we obtain

$$\underline{\theta}'(n) = \left\{\prod_{i=1}^{n} [I-\underline{K}(i)\underline{x}^T(i) + \beta_{NN}(i)]\right\}\underline{\theta}(0) + \tag{8.1}$$

$$\sum_{i=1}^{n} \left\{\prod_{j=i+1}^{n} [I - \underline{K}(j)\underline{x}^T(j) + \beta_{NN}(j)]\right\}\underline{\xi}(i)$$

Note that we define

$$\prod_{j=n+1}^{n} [\cdot] = 1 \tag{8.2}$$

Define the weight excess error vector as

$$\underline{\psi}(n) = \sum_{i=1}^{n} \left\{\prod_{j=i+1}^{n} [I - \underline{K}(j)\underline{x}^T(j) + \beta_{NN}(j)]\right\}\underline{\xi}(i) \tag{8.3}$$

Also define

$$\underline{\chi}(n) = \left\{\prod_{i=1}^{n} [I - \underline{K}(i)\underline{x}^T(i) + \beta_{NN}(i)]\right\}\theta(0) \tag{8.4}$$

Then

$$\underline{\theta}'(n) = \underline{\chi}(n) + \underline{\psi}(n) \tag{8.5}$$

Neglecting the floating point noise sources in the matrix, $\beta_{NN}(i)$, $\underline{\chi}(n)$ describes the convergence of the weight error vector towards $\underline{0}$. That is $\underline{w}(n)$ approaches $\underline{w}^*$ as $n \to \infty$. In this case $\underline{\psi}(n)$ represents an excess weight error vector due to floating point operations in the algorithm. Since the elements of $\underline{\xi}(i)$ are zero mean white random quantaties we can neglect the crosscorrelation between $\underline{\chi}(n)$ and $\underline{\psi}(n)$. Thus, the covariance of $\underline{\theta}'(n)$ becomes,

$$R_{\theta}(n) = E\{\underline{\theta}'(n)\underline{\theta}'^T(n)\} = E\{\underline{\chi}(n)\underline{\chi}^T(n)\} + E\{\underline{\psi}(n)\underline{\psi}^T(n)\} \tag{8.6}$$

We now proceed to calculate

$$R_{\psi\psi}(n) = E\{\underline{\psi}(n)\underline{\psi}^T(n)\} = E\{\sum_{i=1}^{n}\sum_{j=i}^{n}\prod_{k=i+1}^{n}[I - \underline{K}(k)\underline{x}^T(k) + \beta_{NN}(k)]$$

$$\underline{\xi}(i)\underline{\xi}^T(j)\cdot\prod_{m=0}^{n-j-1}[I - \underline{x}(n-m)\underline{K}^T(n-m) + \beta_{NN}^T(n-m)] \tag{8.8}$$

A major simplification of (8.8) results when we note that $\underline{\xi}$ (i) and $\underline{\xi}$ (j) are independent zero mean random vectors for $i \neq j$. Thus,

$$R_\xi(j,k) = E\{\underline{\xi}(i)\underline{\xi}^T(j)\} = \begin{matrix} R_\xi(i) & i=j \\ 0_{NN} & i\neq j \end{matrix} \tag{8.9}$$

Hence terms where $i \neq j$ drop out in (8.8). Now,

$$R_\xi(i) = E\{\underline{\xi}(i)\underline{\xi}^T(i)\} = E\{\alpha_{NN}(i)\underline{w}^*\underline{w}^{T*}\alpha_{NN}(i)\} + E\{\eta^2(i)\underline{K}(i)\underline{K}^T(i)\} \tag{8.10}$$

where we have substituted (5.18) for $\underline{\xi}$ (i). Before we proceed further, we make the assumption that x(n) is a white Gaussian sequence independent in time with autocorrelation matrix,

$$R_x = E\{\underline{x}(n)\underline{x}^T(n)\} = \sigma_x^2 I$$

The following relationships are established in Appendix A where we have used the "averaging principle" [5.6]:

$$E\left\{\underline{K}(k)\underline{K}^T(k)\right\} \approx \left(\frac{1-\lambda}{1-\lambda^k}\right)^2 \frac{1}{\sigma_x^2} I \tag{8.11}$$

$$E\left\{\underline{K}(k)\underline{x}^T(k)\right\} \approx \left(\frac{1-\lambda}{1-\lambda^k}\right) I \tag{8.12}$$

$$E\left\{\underline{K}(k)\underline{x}^T(k)\underline{x}(k)\underline{K}^T(k)\right\} \approx \left(\frac{1-\lambda}{1-\lambda^k}\right)^2 (N+2)\, I \tag{8.13}$$

Hence,

$$R_\xi(i) = \sigma_x^2\ \mathrm{diag}\,[w_i^{*2}] - \frac{\sigma_\eta^2}{\sigma_x^2}\left(\frac{1-\lambda}{1-\lambda^k}\right)^2 I \tag{8.14}$$

Thus $R_\xi(i)$ is a diagonal matrix. Since x(n) is a white Gaussian sequence, when we take the expectation in (8.8) all terms involving the vector $\underline{x}(n)$ will yield diagonal matrices. Furthermore, in (8.8) all terms $j \neq i$ drop out based on (8.9). Also, the elements of the matrix $\beta_{NN}(n)$ are independent white random variables. Thus, the expectation operator will cause all off diagonal elements of (8.8) to become zero. Since diagonal matrices can be permuted, anticipating the expectation operator we can write (8.8) as follows,

$$R_\psi(n) = \sum_{i=1}^{n} \sigma_\xi^2(i) \prod_{k=i+1}^{n} \Xi(k) \tag{8.15}$$

where we have defined,

$$\Xi(k) = E\{[I - \underline{x}(k)\underline{K}^T(k) + \beta_{NN}(k)]\,[I - \underline{K}(k)\underline{x}^T(k) + \beta_{NN}(k)]\} \tag{8.16}$$

Now, from (5.17) the elements of $\beta_{NN}(k)$ are zero mean. We have using (5.17), (5.16), (5.1),

$$\begin{aligned} E\{\beta_{NN}(k)\beta_{NN}^T(k)\} = E\{&\alpha_{NN}(k)\alpha_{NN}^T(k) + \phi_{NN}(k)\underline{K}(k)\underline{x}^T(k)\underline{x}(k)\underline{K}^T(k)\phi_{NN}^T(k) \\ &- \alpha_{NN}(k)\underline{x}(k)\underline{K}^T(k)\phi_{NN}^T(k) - \phi_{NN}^T(k)\underline{K}(k)\underline{x}^T(k)\alpha_{NN}^T(k)\} \end{aligned} \tag{8.17}$$

$$E\{\beta_{NN}(k)\beta_{NN}^T(k)\} = \sigma_\alpha^2 I + (N+2)\left(\frac{1-\lambda}{1-\lambda^k}\right)^2 \left(\sigma_\alpha^2 + \sigma_\mu^2 + \sigma_\delta^2\right) I - \sigma_\alpha^2 \left(\frac{1-\lambda}{1-\lambda^k}\right) I$$

Substituting (8.12), (8.13) and (8.18) into (8.16) we obtain

$$\Xi(k) = \left[1 - \frac{2(1-\lambda)}{1-\lambda^k} + \left(\frac{1-\lambda}{1-\lambda^k}\right)^2 (N+2) + \sigma_\alpha^2 + \left(\frac{1-\lambda}{1-\lambda^k}\right)^2 (N+2)\sigma_\tau^2\right] I \tag{8.19}$$

In the above equations we have defined

$$\sigma_\tau^2 = \sigma_\alpha^2 + \sigma_\mu^2 + \sigma_\delta^2 \tag{8.20}$$

Substituting (8.19) into (8.15) we obtain (j=0,1,... N-1),

$$R_\psi(n) = \sum_{i=1}^{n} \prod_{k=i-1}^{n} \left[1 - \frac{2(1-\lambda)}{1-\lambda^k} + \sigma_\alpha^2\right] \left[\sigma_\alpha^2\ \mathrm{diag}\,[w_j^{*2}] + \sigma_\eta^2 \left(\frac{1-\lambda}{1-\lambda^i}\right)^2 \frac{1}{\sigma_x^2} I\right]$$

where we have used the approximation that for k large and $\lambda \sim 1$.

$$\left(\frac{1-\lambda}{1-\lambda^k}\right)^2 << \frac{1-\lambda}{1-\lambda^k} \tag{8.22}$$

Using the approximation

$$(1+\epsilon)^n \sim e^{\epsilon n} \qquad \epsilon << 1$$

we can write (8.21) as,

$$R_{ww}(n) = \sum_{i=1}^{n} \left\{ \sigma_\alpha^2 \ \text{diag}\,[w_j^{*2}] + \left(\frac{1-\lambda}{1-\lambda^i}\right)^2 \frac{1}{\sigma_x^2}\sigma_\eta^2 I \right\} e^{(n-i)\sigma_x^2 - 2(1-\lambda)\sum_{k=i+1}^{n} \frac{1}{1-\lambda^k}} \tag{8.23}$$

The above result is valid for n large. The initial transient period is not accurately predicted by the above equation due to our approximations, in particular (8.22). The analysis to follow separately considers the two cases $\lambda = 1$ for the prewindowed growing memory RLS algorithm and $\lambda < 1$ for the exponentially windowed RLS algorithms. The reason for this is that different steady state solutions and approximations apply for the two cases.

Prewindowed Growing Memory RLS Algorithm ($\lambda = 1$)

In this case (8.23) becomes,

$$R_{ww}(n) = \sum_{i=1}^{n} \left\{ \sigma_\alpha^2 \ \text{diag}\,[w_i^{*2}] + \frac{1}{i^2}\frac{\sigma_\eta^2}{\sigma_x^2} I \right\} e^{(n-i)\sigma_x^2} e^{-2\sum_{k=i+1}^{n} \frac{1}{i}} \tag{8.24}$$

In Appendix B we show that

$$\sum_{k=i+1}^{n} \frac{1}{i} \sim \ln\frac{n}{i} + \frac{1}{2}\,\frac{n+i+1}{n(i+1)} \qquad n >> 1 \tag{8.25}$$

Substituting into (8.24),

$$R_{ww}(n) = \sum_{i=1}^{n} \{\cdot\} e^{(n-i)\sigma_x^2} \frac{i^2}{n^2} e^{-\frac{n-i-1}{n(i-1)}} \tag{8.26}$$

As i becomes large for n large.

$$e^{-\frac{n-i-1}{n(i-1)}} \approx 1$$

Now for σ_α^2 small and n large,

$$\sum_{i=1}^{n} e^{-i\sigma_\alpha^2} = e^{-\sigma_\alpha^2} \frac{(1-e^{-n\sigma_\alpha^2})}{1-e^{-\sigma_\alpha^2}} \approx \frac{e^{-\sigma_\alpha^2}}{\sigma_\alpha^2} \approx \frac{1}{\sigma_\alpha^2} \tag{8.27}$$

From Appendix B,

$$\sum_{i=1}^{n} i^2 e^{-i\sigma_\alpha^2} \approx 2\sigma_\alpha^{-6} \quad ; \sigma_\alpha \neq 0 \quad , \quad n > \frac{1}{\sigma_\alpha^2} \tag{8.28}$$

Hence

$$R_{\psi}(n) = \frac{e^{n\sigma_\alpha^2}}{n^2} \left\{ \frac{1}{\sigma_\alpha^4} \text{ diag } [w_i^{*2}] + \frac{1}{\sigma_\alpha^2} \frac{\sigma_\eta^2}{\sigma_x^2} I \right\} \tag{8.29}$$

If n is large but small compared to $\frac{1}{\sigma_\alpha^2}$ such that $e^{n\sigma_\alpha^2} \approx 1$, then (8.28) reduces to $\frac{n^3}{3}$ and,

$$R_{\psi}(n) = \left\{ \frac{n}{3} \sigma_\alpha^2 \text{ diag } [w_i^{*2}] + \frac{1}{n} \frac{\sigma_\eta^2}{\sigma_x^2} I \right\} \tag{8.30}$$

Assuming no errors in the calculation of the update for $\underline{w}(n)$, $\sigma^2_{\alpha} - 0$ and noting that

$$\lim_{\sigma_\alpha^2 \to 0} \sum_{i=1}^{n} e^{-i\sigma_\alpha^2} = n \tag{8.31}$$

equation (8.26) reduces to,

$$\lim_{\sigma_\alpha^2 \to 0} R_{\psi}(n) = \frac{1}{n} \frac{\sigma_\eta^2}{\sigma_x^2} I \tag{8.32}$$

Next consider the covariance of $\underline{\chi}$ (n) which contains the information regarding the convergence of the weight error vector. Define,

$$R_{\chi}(n) = E\{\underline{\chi}(n) \underline{\chi}^T(n)\} \tag{8.33}$$

From (8.4) substituting for $\underline{\chi}(n)$

$$R_\chi(n) = \underline{\theta}(0)E\left\{\prod_{i=1}^{n}\ [I-\underline{K}(i)\underline{x}^T(i)+\beta_{NN}(i)]\ \prod_{m=0}^{n-1}\ [I-\underline{x}(n-m)\underline{K}^T(n-m)+\beta_{NN}(n-m)]\right\}\underline{\theta}^T(0) \tag{8.34}$$

From (8.16)

$$R_\chi(n) = \underline{\theta}(0)\ \prod_{i=1}^{n}\ \Xi\ (i)\ \underline{\theta}^T(0) \tag{8.35}$$

Substituting for Ξ (i) from (8.19),

$$R_\chi = \underline{\theta}(0)\ \prod_{i=1}^{n}\ (1-\frac{2}{i}+\sigma_\alpha^2\ )\underline{\theta}^T(0) = \underline{\theta}(0)\underline{\theta}^T(0)\ e^{-2\sum_{i=1}^{n}\frac{1}{i}}\ e^{n\sigma_\alpha^2} \tag{8.36}$$

where we have neglected terms of order $O(i^{-2})$. Now,

$$\sum_{i=1}^{n}\ \frac{1}{i} \approx \ln n - \frac{1}{2n} + \gamma_c \quad ;\gamma_c = 0.57721\ \ n >> 1 \tag{8.37}$$

Thus,

$$R_\chi(n) = \underline{\theta}(0)\underline{\theta}^T(0)\frac{1}{n^2}\ e^{\frac{1}{n}}\cdot e^{-2\gamma_c}\ e^{n\sigma_\alpha^2} \tag{8.38}$$

Exponentially Windowed RLS Algorithm (λ 1)

Define

$$g(\lambda,n) = \sum_{k=1}^{n}\ \frac{1}{1-\lambda^k} \tag{8.39}$$

in Appendix B we show that

$$\lim_{\lambda\to 1} g(\lambda,n) \approx n + \frac{1}{1-\lambda}\ \ln\frac{1}{1-\lambda} \tag{8.40}$$

Define

$$f(\lambda,i,n) = \sum_{k=i-1}^{n}\ \frac{1}{1-\lambda^k} = g(\lambda,n) - g(\lambda,i) \tag{8.41}$$

From Appendix B.

$$\lim_{\lambda\to 1} f(\lambda,i,n) \approx (n-i) - \lambda^{i-1}\frac{1}{1-\lambda}\ \ln\ \frac{1}{1-\lambda} \tag{8.42}$$

Hence (8.23) becomes,

$$R_{\psi}(n) = \sum_{i=1}^{n} \{\cdot\} e^{(n-i)\sigma_{\alpha}^2} e^{-2(1-\lambda)(n-i)} e^{\ln(1-\lambda)^{2\lambda^i+1}}$$

$$R_{\psi}(n) = \sum_{i=1}^{n} \{\cdot\} e^{(n-i)\sigma_{\alpha}^2} e^{-2(1-\lambda)(n-i)} (1-\lambda)^{2\lambda^i+1} \tag{8.43}$$

Consider the index i=M for which $\lambda^i << 1$, or $M > \frac{1}{1-\lambda}$ then (j=0,1,....,N-1),

$$R_{\psi}(n) = \sum_{i=M}^{n} \left\{ \sigma_{\alpha}^2 \ \text{diag}\,[w_j^{*2}] \ + (1-\lambda)^2 \frac{\sigma_{\eta}^2}{\sigma_x^2} \ I \right\} e^{-i\upsilon} + C(M) \tag{8.44}$$

where we have made the substitution

$$\upsilon = \ \sigma_{\alpha}^2 \ - 2(1-\lambda) \tag{8.45}$$

The matrix C(M) collects the terms for which i<M.

$$C(M) = \sum_{i=1}^{M-1} \{\cdot\} e^{(n-i)\upsilon} (1-\lambda)^{2\lambda^i+1} \tag{8.46}$$

$$R_{\psi}(n) = C(M) + \ \sigma_{\alpha}^2 \ \frac{e^{(n-M)\upsilon} - e^{-\upsilon}}{1-e^{-\upsilon}} \ \text{diag}\,[w_i^{*2}] \ + (1-\lambda)^2 \frac{\sigma\eta^2}{\sigma_x^2} \frac{e^{(n-M)\upsilon} - e^{-\upsilon}}{1-e^{-\upsilon}} \ I \tag{8.47}$$

If $\upsilon < 0$ as n becomes large

$$R_{\psi}(n) = C(M) + \ \sigma_{\alpha}^2 \ \frac{e^{-\upsilon}}{e^{-\upsilon}-1} \ \text{diag}\,[w_i^{*2}] \ + (1-\lambda)^2 \frac{\sigma_{\eta}^2}{\sigma_x^2} \ \frac{e^{-\upsilon}}{e^{-\upsilon}-1} \ I \tag{8.48}$$

If $\upsilon > 0$, n large

$$R_{\psi}(n) = C(M) + \ \sigma_{\alpha}^2 \ \frac{e^{(n \ - \ M)\upsilon}}{1-e^{-\upsilon}} \cdot \ \text{diag}\,[w_i^{*2}] \ + (1-\lambda)^2 \frac{\sigma_{\eta}^2}{\sigma_x^2} \ \frac{e^{(n \ - \ M)\upsilon}}{1-e^{-\upsilon}} \ I \tag{8.49}$$

Next examine R_{γ} for $\lambda < i$. From (8.36),

$$R_\chi(n) = \underline{\theta}(0) \prod_{i=1}^{n} \Xi(i) \underline{\theta}^T(0) \tag{8.49}$$

where,

$$\Xi(i) \approx \left(1-2\frac{(1-\lambda)}{1-\lambda^i} + \sigma_\alpha^2\right) I \approx e^{-2\frac{(1-\lambda)}{1-\lambda^i}} e^{\sigma_\alpha^2} \tag{8.50}$$

Hence

$$R_\chi(n) = \underline{\theta}(0)\, \underline{\theta}^T(0)\, e^{n\sigma_\alpha^2}\, e^{-2(1-\lambda)n}\, e^{-2\ln\frac{1}{1-\lambda}} \tag{8.51}$$

where we have used

$$\sum_{i=1}^{n} \frac{1}{1-\lambda^i} = n + \frac{1}{1-\lambda}\ln\frac{1}{1-\lambda} \tag{8.52}$$

Thus,

$$R_\chi(n) = \underline{\theta}(0)\underline{\theta}^T(0)\,(1-\lambda)^2 e^{n\upsilon} \tag{8.54}$$

where υ is defined by (8.46).

9. Extension to the Least Mean Squares Algorithm

In this section, the errors introduced by floating point operations in the LMS algorithm are derived using the approach for the RLS algorithm. In the LMS algorithm the prediction of the desired signal and the prediction error are computed as in the RLS algorithm, (3.3) and (3.4). However, the weight update recursion is computed by replacing the Kalman gain (3.6) with the simpler expression $\gamma \underline{x}(n)$,

$$\underline{w}(n) = \underline{w}(n-1) + \gamma \underline{x}(n)e(n) \tag{9.1}$$

For stability, γ must satisfy

$$\gamma < \frac{2}{\lambda_{max}}$$

where λ_{max} is the maximum eigenvalue of the autocorrelation matrix

$$R_x = E\{\underline{x}(n)\underline{x}^T(n)\} \tag{9.2}$$

Comparing (9.1) to (3.3) we observe that the development leading to (8.8) in the case of the RLS algorithm can be carried out by replacing $\underline{K}(n)$ with $\gamma \underline{x}(n)$ for the LMS algorithm. Hence, for the LMS algorithm, the floating point weight error vector becomes comes:

$$\theta'(n) = \prod_{i=1}^{n} [I - \gamma \underline{x}(i)\underline{x}^T(i) + \beta_{NN}(i)]\hat{\theta}(0) + \tag{9.3}$$

$$\sum_{i=1}^{n} \left[\prod_{j=i-1}^{n} [I - \gamma \underline{x}(j)\underline{x}^T(j) + \beta_{NN}(j) \right] \xi(i)$$

where

$$\xi(i) = \alpha_{NN}(i)\underline{w}^* - \eta(i)\gamma \underline{x}(i) \tag{9.4}$$

and

$$\beta_{NN}(n) = \alpha_{NN}(i) - \tilde{\phi}_{NN}(n)\gamma \underline{x}(n)\underline{x}^T(n) \tag{9.5}$$

Note that the floating point errors, $\alpha_i(n)$, $\mu_i(n)$, $\delta(n)$, $\rho(n)$ now apply to the LMS algorithm.

Proceeding as in the RLS algorithm we can derive the following corresponding expressions for the LMS algorithm.

$$R_\xi = \sigma_\alpha^2 \mathrm{diag}[w_i^{*2}] + \sigma_\eta^2 \gamma^2 R_x \tag{9.6}$$

$$\Xi(k) = \nu I \tag{9.7}$$

$$\nu = 1-2\gamma\sigma_x^2+(N+2)\sigma_x^4\gamma^2+\sigma_\alpha^2-2\sigma_\alpha^2\gamma\sigma_x^2+(N+2)\gamma^2\sigma_x^4\sigma_\tau^2 \tag{9.8}$$

Note that $\Xi(k)$ is diagonal. Hence, for the LMS algorithm,

$$R_{\psi} = \sum_{i=1}^{n} \prod_{k=i+1}^{n} \Xi(k)\ E\{\xi(i)\xi^T(i)\} \tag{9.9}$$

Or,

$$R_{\psi} = \sum_{i=1}^{n} R_\xi\, \nu^{n-i} \approx R_\xi\,(1-\nu)^{-1} \tag{9.10}$$

(Note that $\nu<1$ since $\gamma << \frac{1}{\sigma_x^2}$ for practical purposes). Substituting for R_ξ from (9.6) we obtain.

$$\mathrm{Trace}\,(R_{\psi}) = \left[\sigma_\alpha^2||\underline{w}^*||^2 + \gamma^2\sigma_\eta^2 N\sigma_x^2\right] \frac{1}{1-\nu} \tag{9.11}$$

For γ small, $\nu \sim$ 1-2 $\gamma\sigma_x^2$ from (9.8). Hence,

$$\mathrm{Trace}\ (R_{\psi}) = \frac{\sigma_\alpha^2||w^*||^2}{2\gamma\sigma_x^2} + \frac{1}{2}\gamma N\sigma_\eta^2 \tag{9.12}$$

Consider now the floating point errors introduced in $\underline{\chi}(n)$. From (8.15) we obtain for the LMS algorithm,

$$R_\chi(n) = \underline{\theta}\ (0)\underline{\theta}^{\ T}(0)\nu^n \tag{9.13}$$

Examining (9.13) we note that in the absence of floating point errors, $\lim_{n\to\infty} \nu^n \to 0$ must hold for the algorithm to converge. Assuming γ to be small (this is the case in many applications of the LMS algorithm) we can write (9.8) as

$$\nu \sim 1 - 2\gamma\sigma_x^2 + \sigma_\alpha^{\ 2} \tag{9.14}$$

For stability

$$2\gamma\sigma_x^2 << 1 \tag{9.15}$$

Hence,

$$\nu \sim e^{-2\gamma\sigma_x^2} e^{\sigma_\alpha^2} \tag{9.16}$$

Finally,

$$R_\chi(n) = \underline{\theta}(0)\underline{\theta}^T(0)\, e^{-2\gamma\sigma_x^2 n} \cdot e^{n\sigma_\alpha^2} \tag{9.17}$$

The expected value of the floating point weight error vector is

$$E\{||\underline{\theta}'(n)||^2\} = \text{Trace}[R_\chi(n) + R_\psi(n)] = \tag{9.18}$$

$$||\underline{\theta}(0)||^2 e^{-2\gamma\sigma_x^2 n} \cdot e^{n\sigma_\alpha^2} + \frac{\sigma_\alpha^2 ||w^*||^2}{2\gamma\sigma_x^2} + \frac{1}{2}\gamma N\sigma_\eta^2$$

From this expression it is clear that a tradeoff exists in the choice of the loop gain γ. In order to reduce the bias due to additive noise and the inner product calculation of the desired signal prediction, γ must be reduced. On the other hand, this amplifies the floating point roundoff error due to the floating point weight vector update calculation. This observation is similar to the tradeoff in the choice of the forgetting factor in the time varying RLS algorithm. In both algorithms the precision used in the calculation of the weight vector correction term has no effect on the steady state performance. This result can be used to simplify the hardware realization of the algorithm.

In the absence of floating point errors and additive noise, σ_η^2–0 and σ_α^2–0. Then (9.18) reduces to,

$$E\{||\underline{\theta}'(n)||^2\} = ||\underline{\theta}(0)||^2 e^{-2\gamma\sigma_x^2 n} \tag{9.19}$$

Thus the LMS algorithm will converge to the true impulse response $\underline{w}^*$ independent of the initial conditions. The rate of convergence is proportional to the loop gain γ.

The expression for the mean square prediction error can be obtained from (5.26) and (5.28) by substituting for $E\{||\underline{\theta}'(n)||^2\}$,

$$\sigma_e^2(n) = ||\underline{\theta}(0)||^2 \sigma_x^2 e^{-2\gamma\sigma_x^2 n} \cdot e^{n\sigma_\alpha^2} + \frac{\sigma_\alpha^2 ||w^*||^2}{2\gamma} + \frac{\sigma_x^2}{2}\gamma N\sigma_\eta^2 - \sigma_\eta^2 \tag{9.20}$$

where we have neglected the term σ_δ^2.

10. Simulation Results

In this section the effects of floating point errors on the performance of the RLS and LMS algorithms will be presented through simulation. Using assembly level subroutines (frexp and ldexp) in UNIX it is possible to quantize the mantissa of floating point numbers. In the simulations to follow the operations (3.1) through (3.3) for the RLS algorithm and (3.1), (3.2) and (9.1) for the LMS algorithm are carried out with various number of bits allocated to the mantissa. In the simulations the taps of the weight vector are estimated from the input and output (desired signal) sequence of a white Gaussian noise process convolved with a 9 tap impulse response. In the simulations, the norm of the weight error vector,

$$||\underline{\theta}(n)||^2 = \underline{\theta}^T(n)\underline{\theta}(n)$$

where,

$$\underline{\theta}(n) = \underline{w}(n) - \underline{w}^*$$

is plotted in dB's against the number of iterations. The number of bits used for the mantissa in the floating point operations are indicated by B_α for the floating point addition in the weight vector update, B_μ for the floating point calculation of the weight vector update term ($\underline{K}(n)e(n)$), and B_η for the inner product calculation of the prediction of the desired signal. If the number of bits allocated to the mantissa equals 23, then this corresponds to 32 bit floating point operations. In the Figures if B is not indicated for a particular operation then it is equal to 23 bits.

The effects of the forgetting factor, λ, on the convergence of the RLS algorithm for various number of bits allocated to the mantissa in the floating point addition of the weight vector update recursion (B_α) is presented in Figures 2 and 3. All other operations are carried out with 23 bit mantissas. In Fig. 2, B_α=9 bits, and the forgetting factor is changed. The variance of the additive noise (σ_v^2) is 10^{-6} in both Figures. In the infinite precision case, the steady state weight error vector norm should decrease as the forgetting factor approches 1 when additive noise is present. However, as Fig. 2 clearly shows, the steady state error actually increases due to floating point errors ($\alpha_i(n)$) when $\lambda \rightarrow 1$, which is in agreement with the theoretical result (6.12). In Fig. 3, we observe that the norm decreases as the number of iterations increases for the infinite precision calculation when λ=1. However, for B_α=10 the steady state error increases with $\lambda \rightarrow 1$. For comparison, the finite precision and infinite precision results for λ=0.99 are also presented.

Based on the theoretical results in section 6, the number of bits used for B_μ should not effect steady state performance. This result is verified in the simulation presented in Fig.

3 where B_μ is changed from 23 to 8 bits ($\sigma_v^2 = 10^{-6}$). Very little difference is observed between the two cases. According to the theoretical results of section 6, decreasing λ should increase the contribution of the floating point error, in the calculation of the desired signal prediction, to the steady state weight error vector norm. This is verified in Fig. 4 where B_η=9 bits and $\sigma_v^2 = 0$. The steady state error decreases as $\lambda \rightarrow 1$.

Early termination of updating is examined in Fig. 5 for the prewindowed growing memory case. Note that the steady state error increases approximately by 6 dB as B_α is decreased by one bit since, based on the arguments of section 7, the steady state error is proportional to σ_α^2 when the additive noise is very small ($\sigma_v^2 = 10^{-10}$ in this case).

In the Fast RLS algorithms [3,1] the Kalman gain exhibits large random changes, although its norm should decrease as the number of iterations increases. This is particularly true when a very small soft constraint is used. In these cases updating may never terminate and the divergence due to floating point errors predicted in this paper may well occur. This divergence phenomenon is examined in the simulations presented in Figures 6 and 7, where the Fast Kalman algorithm is used to compute the Kalman gain with double precision floating point. We observe that the norm increases after initial convergence to a small error. The trend towards divergence is amplified when B_α is decreased from 10 bits to 6 bits. Significantly, as predicted in this paper, the precision used in B_μ does not effect the steady state solution which is demonstrated in Fig. 8 in the case of the Fast Kalman algorithm. Note the difference compared to using $B_\alpha = 10$ in Fig. 6. Single precision 32 bit floating point simulations of the Fast Kalman algorithm seem to diverge after a large number of iterations even though the algorithm initially converges to a very small steady state error. This phenomenon can be partially explained by the results of this paper.

In Figures 9 through 11 simulation results for floating point operations are presented for the LMS algorithm. The equivalent performance to 32 bit floating point arithmetic is presented in Fig. 9 with 23 bits used in the mantissas. The loop gain was γ=0.01 in this case. Again based on the result for the floating point error analysis of the LMS algorithm (9.18) the steady state error does not depend on the number of bits used for B_μ in the weight vector update recursion (9.1). This result is verified in Fig. 9, where the performance of the LMS algorithm is hardly degraded when the mantissa is quantized from MU=23 bits to MU=8 bits in the calculation of $\gamma \underline{x}(n)e(n)$. However, as is predicted by

(9.18) floating point errors due to the summation operation in the weight vector update recursion (σ_α^2), do degrade performance inversely to the loop gain γ. In Fig. 10, the performance degradation due to floating point errors with an 8 bit mantissa (B_α=8) in the calculation of the weight vector update summation is presented for various loop gains γ. Clearly, the noise due to floating point operations in this calculation is inversely related to the loop gain as predicted in (9.18). Finally, the effect of floating point errors in the calculation of the desired signal and prediction error is presented in Fig. 11. In this case, the floating point error behaves like an additive noise term and decreases as the loop gain decreases.

11. Summary

In this paper a floating point roundoff error analysis of the RLS and LMS adaptive filtering algorithms was presented. For the RLS algorithm, the prewindowed, growing memory and the exponential sliding window algorithms were considered. It was shown that the noise contribution of certain floating point operations increased as the forgetting factor decreased while the noise due to other operations increased. Some floating point operations did not lead to any steady state increase in noise. In general, divergence was found to be less likely when the forgetting factor was small.

Similar results were obtained for the LMS algorithm where the steady state excess errors due to floating point summation in the weight vector recursion increase inversely to the loop gain. The results agree with reference [7]. The errors in the calculation of the weight vector correction term do not affect the steady state performance of the LMS algorithm. The floating point errors in the calculation of the prediction of the desired signal and prediction error lead to an additive white noise term as in the RLS algorithm. Finally, simulations were presented which confirmed the theoretical findings.

12.Appendix A

Define

$$\bar{R}(i) = \sum_{j=0}^{i} \lambda^{i-j}\, \underline{x}(j)\underline{x}^T(j) \tag{A.1}$$

Therefore, the Kalman gain given by (2.6) can be written as,

$$\underline{K}(n) = \bar{R}^{-1}(n)\underline{x}(n) \tag{A.2}$$

In the following we assume that x(n) is a white Gaussian process with variance σ_x^2. Taking the expected value of (A.1) we obtain,

$$E\{\bar{R}(i)\} \approx \sum_{j=0}^{i} \lambda^{i-j}\, \sigma_x^2\, I = \frac{1-\lambda^{i+1}}{1-\lambda}\, \sigma_x^2\, I \tag{A.3}$$

Now consider the expected value of $\underline{K}(n)\underline{x}^T(n)$.

$$E\{\underline{K}(n)\underline{x}^T(n)\} = E\{\bar{R}^{-1}(n)\underline{x}(n)\underline{x}^T(n)\} \tag{A.4}$$

Since $\bar{R}^{-1}(n)$ is "slowly" varying with respect to $\underline{x}(n)\underline{x}^T(n)$ for λ close to one, we can make use of the averaging principle [5,6] to obtain,

$$E\{\underline{K}(n)\underline{x}^T(n)\} = E\{\bar{R}^{-1}(n)\}\, E\{\underline{x}(n)\underline{x}^T(n)\} \tag{A.5}$$

$$= \left(\frac{1-\lambda}{1-\lambda^{n+1}}\right) I$$

Similarly we can derive,

$$E\{\underline{K}(n)\underline{K}^T(n)\} = E\{\bar{R}^{-1}(n)\underline{x}(n)\underline{x}^T(n)\bar{R}^{-1}(n)\} = \left(\frac{1-\lambda}{1-\lambda^{n-1}}\right)^2 \frac{1}{\sigma_\alpha^2}\, I \tag{A.6}$$

$$E\{\underline{K}(n)\underline{x}^T(n)\underline{x}(n)\underline{K}^T(n)\} = E\{\bar{R}^{-1}(n)\underline{K}(n)\underline{x}^T(n)\underline{x}(n)\underline{x}^T(n)\bar{R}^{-1}(n)\} \tag{A.7}$$

When x(n) is a white Gaussian random process it has been shown in [12] and [8] that,

$$E\{\underline{x}(n)\underline{x}^T(n)\underline{x}(n)\underline{x}^T(n)\} = (N+2)\sigma_x^4 I \tag{A.8}$$

Hence,

$$E\{\underline{K}(n)\underline{x}^T(n)\underline{x}(n)\underline{K}^T(n)\} = \left(\frac{1-\lambda}{1-\lambda^{n-1}}\right)^2 (N+2)\, I \tag{A.9}$$

For $\lambda=1$ we have

$$\left(\frac{1-\lambda}{1-\lambda^{n-1}}\right) = \frac{1}{n} \tag{A.10}$$

13. Appendix B

Let

$$g(n) = \sum_{i=1}^{n} \frac{1}{i} \tag{B.1}$$

For $n >> 1$, $g(n)$ can be approximated as,

$$g(n) \approx \ln n - \frac{1}{2n} + \gamma_c \;\; ; \gamma_c = 0.57721 \;\; n >> 1 \tag{B.2}$$

Hence,

$$\sum_{k=i+1}^{n} \frac{1}{i} = g(n) - g(i) = \ln \frac{n}{i} + \frac{1}{2}\,\frac{n+i+1}{n(i+1)} \tag{B.3}$$

Let

$$G(n.x) = \sum_{i=1}^{n} e^{-ix} \tag{B.4}$$

Then

$$G_2(n,x) = \sum_{i=1}^{n} i^2 e^{-ix} = \frac{\partial^2 G(n,x)}{\partial i^2} \tag{B.5}$$

Thus,

$$G_2(n,x) = \frac{1}{(1-e^{-x})^3}\left\{[e^{-x}(1-e^{-nx}) - n^2 e^{-nx}(1-e^{-x})](1-e^{-x}) - 2e^{-x}[ne^{-nx}(1-e^{-x}) - (1-e^{-nx})e^{-x}]\right\} \tag{B.6}$$

For $x << 1$ and $n >> \frac{1}{x}$

$$G_2(n,x) \approx \frac{1}{x^3} \tag{B.7}$$

Let

$$u(\lambda,n) = \sum_{k=1}^{n} \frac{1}{1-\lambda^k} \tag{B.8}$$

Then

$$u(\lambda,n) = \sum_{k=1}^{n}\sum_{i=0}^{\infty} \lambda^{ik} = \sum_{i=0}^{\infty}\sum_{k=1}^{n} \lambda^{ik} = \sum_{i=0}^{\infty} \frac{\lambda^i(1-\lambda^{ni})}{1-\lambda^i} \tag{B.9}$$

$$u(\lambda,n) = n + \sum_{i=1}^{\infty} \frac{\lambda^i}{1-\lambda^i}(1-\lambda^{-ni})$$

For n large, $\lambda < 1$, $\lambda^{ni} << 1$. If we make use of the following approximation [13]

$$\lim_{x\to 1} \sum_{i=1}^{\infty} \frac{x^i}{1-x^i} \approx \frac{1}{1-x} \ln \frac{1}{1-x} \tag{B.10}$$

we can write (B.9) as follows,

$$\lim_{\lambda\to 1} u(\lambda,n) = n + \frac{1}{1-\lambda} \ln \frac{1}{1-\lambda} \tag{B.11}$$

Now define

$$f(\lambda,i,n) = \sum_{k=i+1}^{n} \frac{1}{1-\lambda^k} = u(\lambda,n) - u(\lambda,i) \tag{B.12}$$

$$= \sum_{k=i+1}^{n} \sum_{j=0}^{\infty} \lambda^{jk} = \sum_{j=0}^{\infty} \sum_{k=i+1}^{n} \lambda^{jk}$$

$$= \sum_{j=0}^{\infty} \lambda^{j} \frac{(\lambda^{i+1} - \lambda^{n+1})}{1 - \lambda^{j}}$$

$$= (n-i) + \sum_{j=1}^{\infty} \frac{\lambda^{j}[\lambda^{i+1} - \lambda^{n+1}]}{1 - \lambda^{j}}$$

For n large ,

$$\lim_{\lambda \to 1} f(\lambda,i,n) \approx (n-i) + \frac{\lambda^{i-1}}{1-\lambda} \ln \frac{1}{1-\lambda} \qquad \text{(B.13)}$$

14.References

(1) Lennart Ljung and Torsten Soderstrom, *Theory and Practice of Recursive Identification*, MIT Press, 1983

(2) T.A.C.M. Claasen and W.F.G. Mecklenbrauker, "Adaptive Techniques for Signal Processing and Communications," *IEEE Communications Magazine*, November 1985, Vol. 23, No. 11

(3) D.D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," *IEEE Trans. on Comm.*, Vol. COM-26, No.10, October 1978, pp 1436-1446.

(4) Stefan Ljung and Lennart Ljung, "Error Propagation Properties of Recursive Least-squares Adaptation Algorithms," *Automatica*, Vol. 21 No. 2, pp. 157-167, 1985

(5) C. Samson and V. U. Reddy, "Fixed Point Error Analysis of the Normalized Ladder Algorithm," *IEEE Trans. on Acoustics, Speech and Sig. Proc.*, Vol. ASSP-31, October 1983

(6) Y. Iiguni, H. Sakai, and H. Tokumaru, "Convergence Properties of Simplified Gradient Adaptive Lattice Algorithms," *IEEE Trans. on Acoustics, Speech and Sig. Proc.*, Vol. ASSP-33, No. 6, December 1985

(7) C.Caraiscos, and B. Liu, "A Roundoff Error Analysis of the LMS Adaptive Algorithm," *IEEE Trans. on Acoustics, Speech and Sig. Proc.*, Vol. ASSP-32, No. 1, February 1984, pp.34-41.

(8) Sasan Ardalan, S.T. Alexander, "Finite Wordlength Analysis of the Recursive Least Squares Algorithm", *Proceedings Eighteenth Annual Asilomar Conference on Circuits, Systems and Computers*, Monterey , CA ,November 4, 1984.

(9) Bede Liu, T. Kaneko, "Error Analysis of Digital Filters Realized with Floating-Point Arithmetic," *Proceedings of the IEEE*, Vol 57, No. 10, October 1969.

(10) A. V. Oppenheim and R. W. Shafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1975.

(11) A.B. Spirad and D.L. Snyder, "Quantization Errors in Floating-Point Arithmetic," *IEEE Trans Acous. Speech, and Sig. Proc.*, vol. ASSP-26, pp. 456-463, Oct. 1978.

(12) L.L. Horowitz and K.P. Senne, "Performance Advantage of Complex LMS for Controlling Narrow-Band Adaptive Arrays," *IEEE Trans. on Acoustics, Speech and Sig. Proc.*, Vol. ASSP-29, No.3, June 1981.

(13) L.B.W. Jolley, *Summation of Series*,, Dover Publications, Inc., New York 1961.

Fig. 1. Adaptive Filtering as System Identification

Fig. 2 Effects of λ on Weight Error Vector Norm

Fig. 3. Effects of λ on Weight Error Vector Norm

Fig. 4. Effect of B_μ on Weight Error Vector Norm

Fig. 5. Effect of λ and B_η

Fig. 6. Early Termination of Updating $\lambda=1$, $\sigma_v^2 = 10^{-10}$

Fig. 7. Fast Kalman Algorithm. $B_\alpha = 10$

Fig. 8. Fast Kalman Algorithm. $B_\alpha = 8$

Fig. 9. Fast Kalman Algorithm. $B_\mu = 23$ and 8

Fig. 10. LMS Algorithm, $B_\mu = 23$ and 8

Fig. 11. LMS Algorithm, Effects of Loop Gain γ , With $B_\alpha = 8$

Fig. 12. LMS Algorithm. Effects of Loop Gain γ, With $B_\eta = 8$

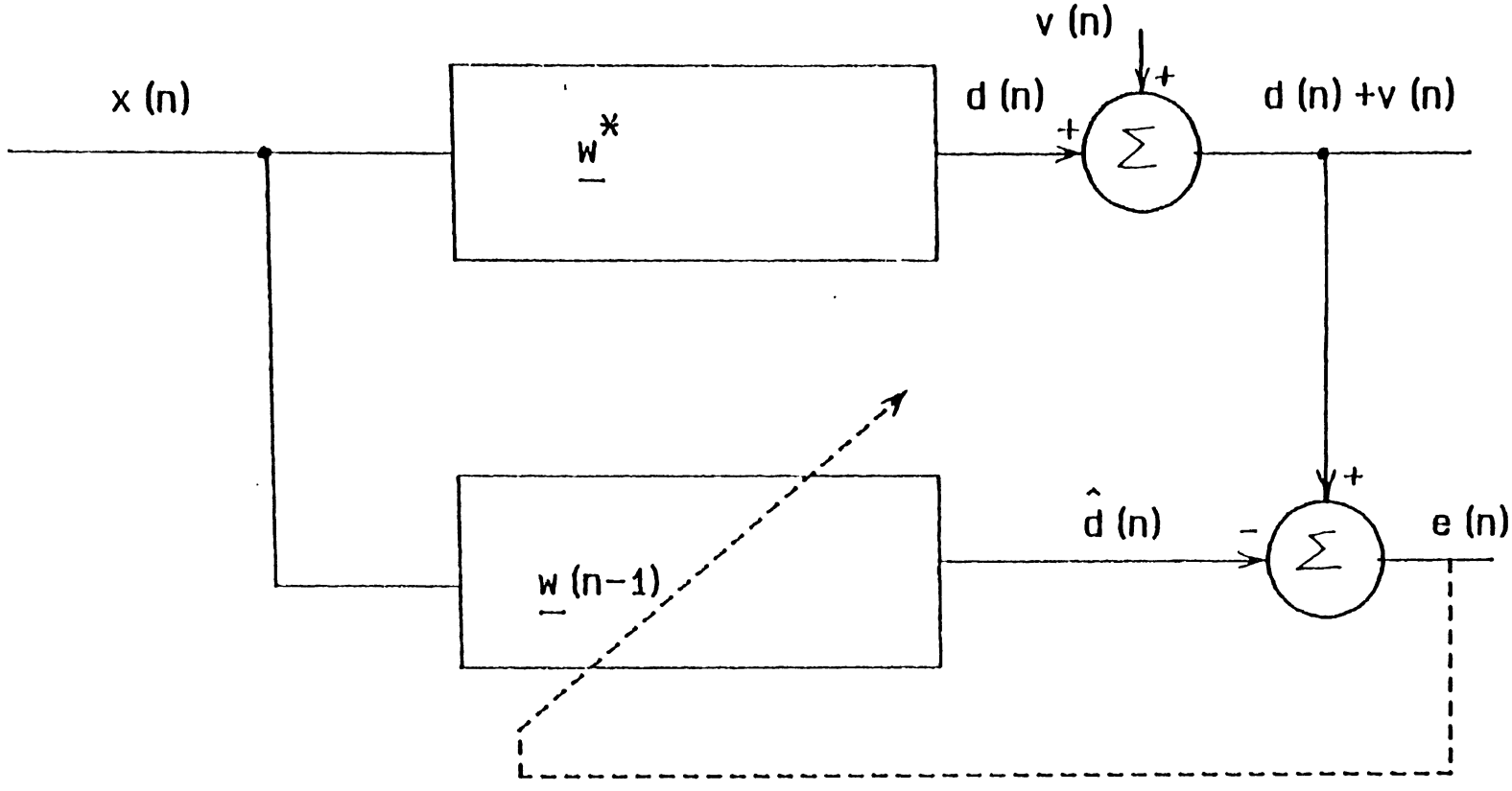

Fig. 1. Adaptive Filtering as System Identification

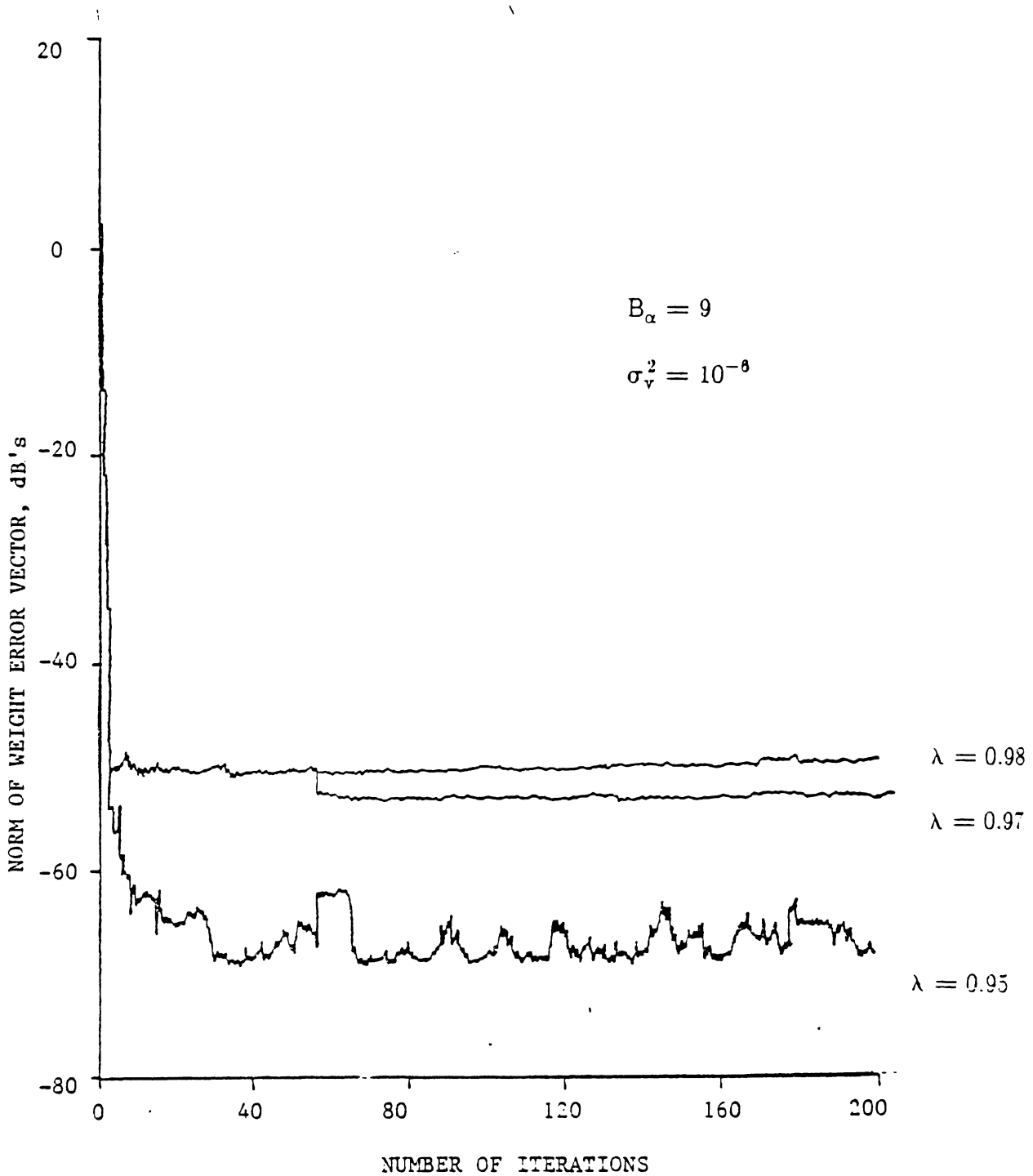

Fig. 2 Effects of λ on Weight Error Vector Norm

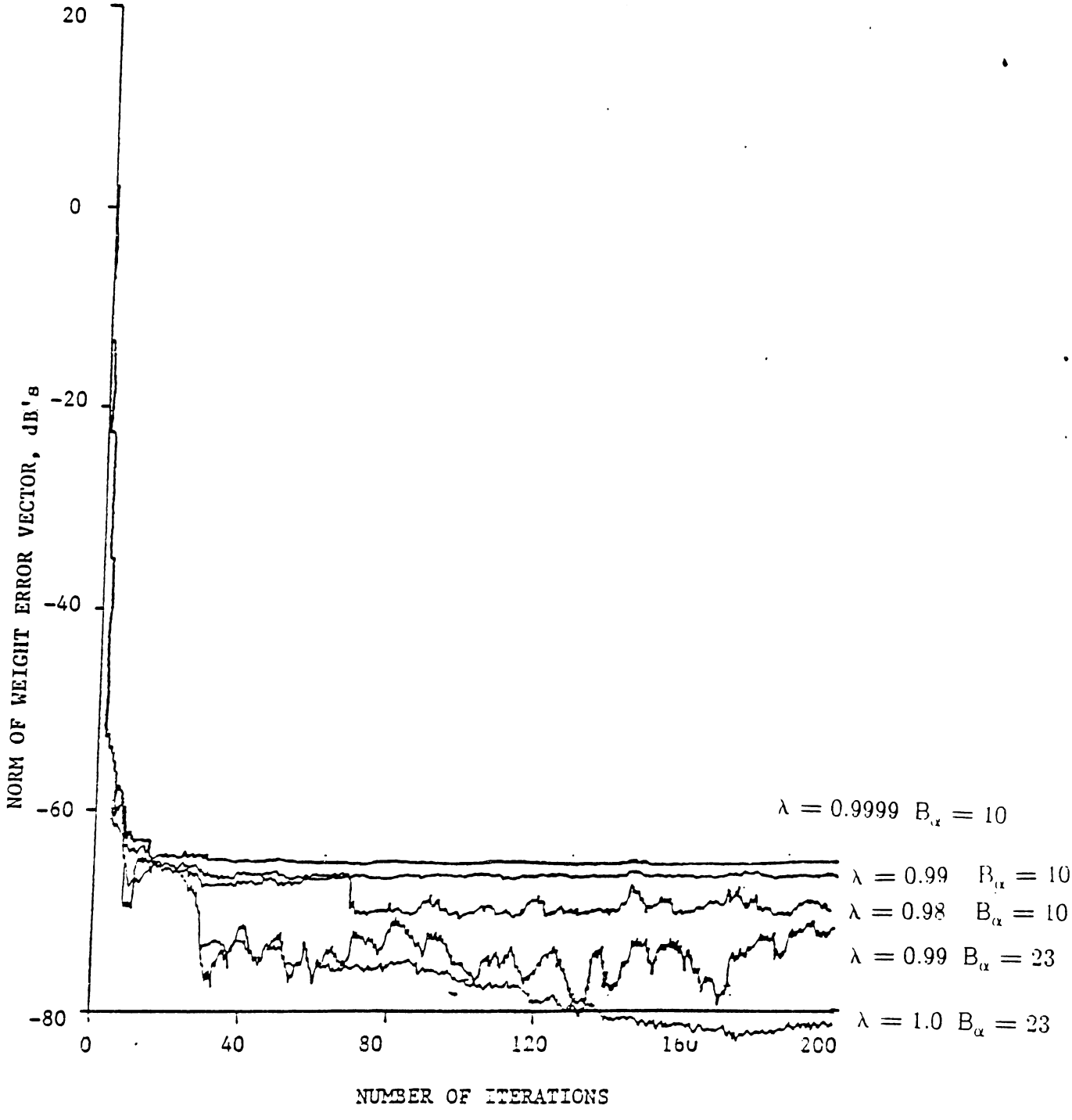

Fig. 3. Effects of λ on Weight Error Vector Norm

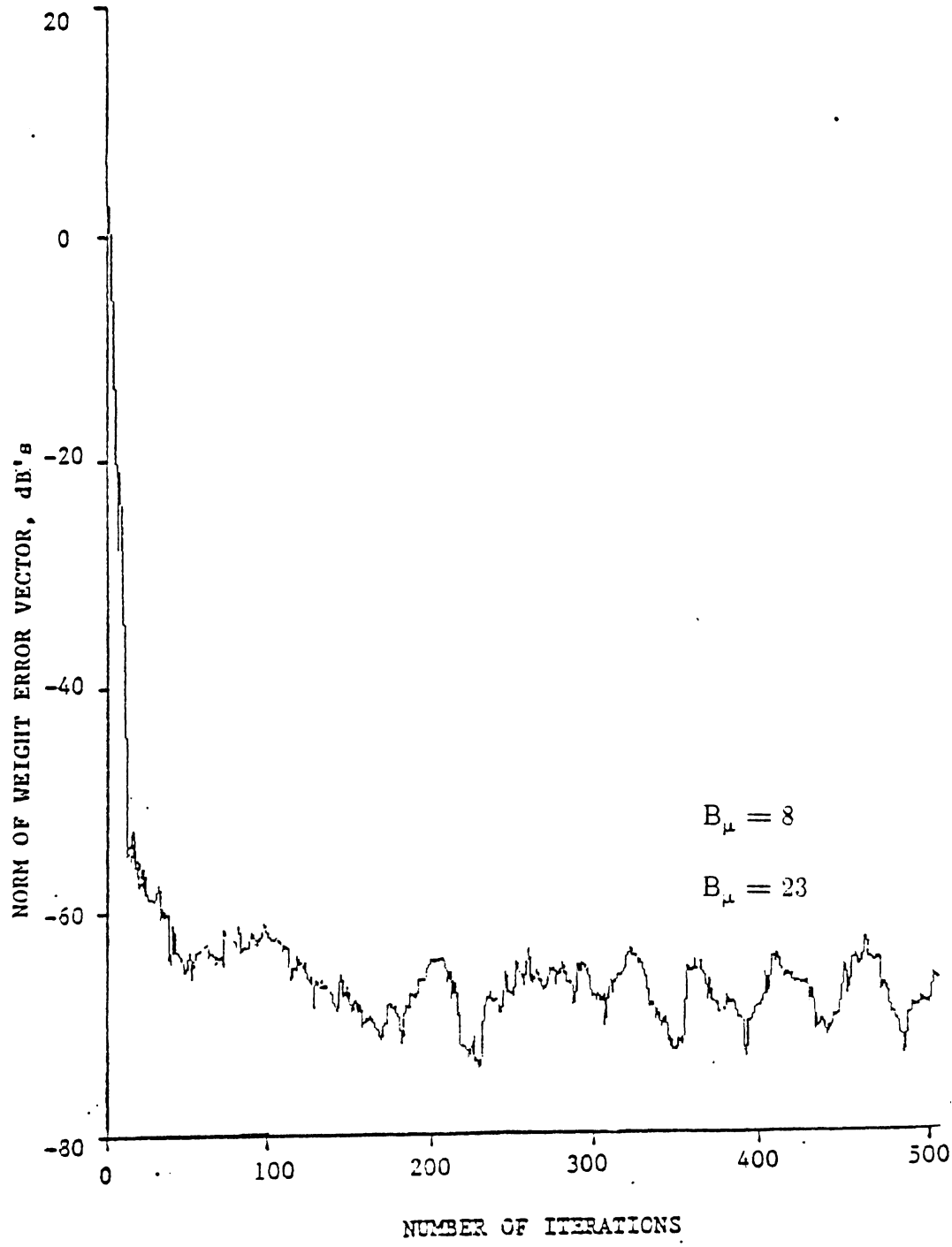

Fig. 4. Effect of B_μ on Weight Error Vector Norm

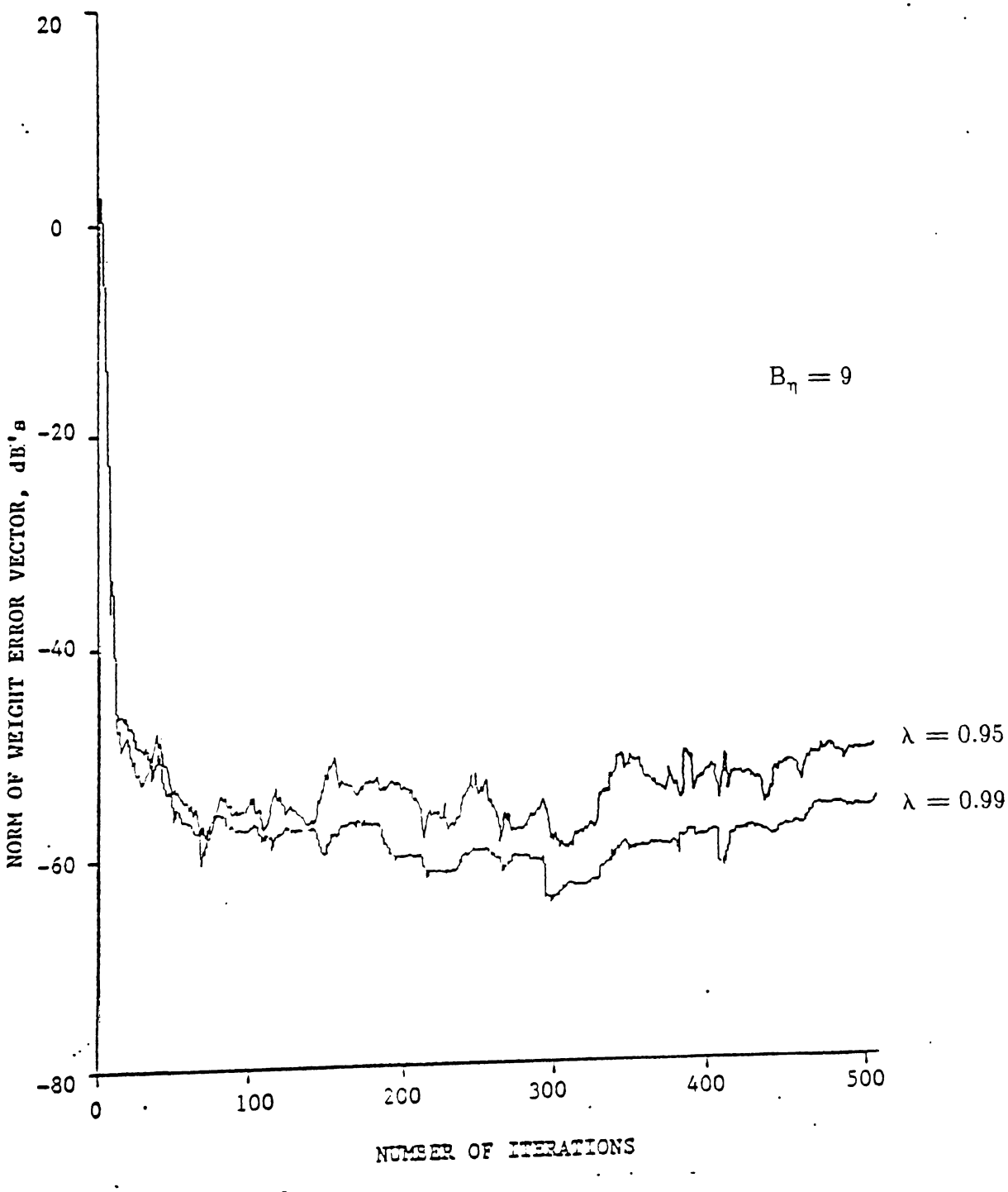

Fig. 5. Effect of λ and B_η

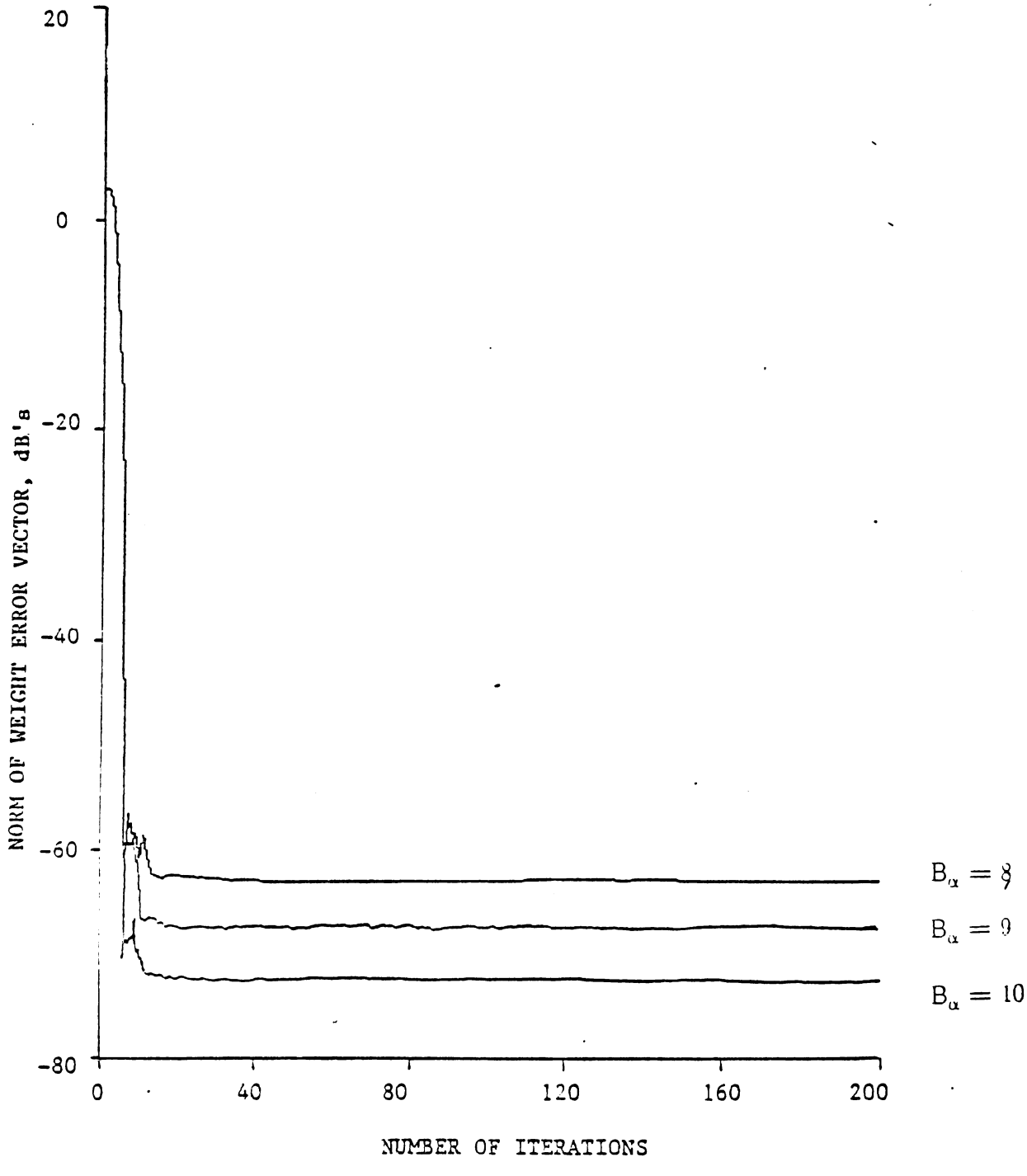

Fig. 6. Early Termination of Updating $\lambda=1$, $\sigma_v^2 = 10^{-10}$

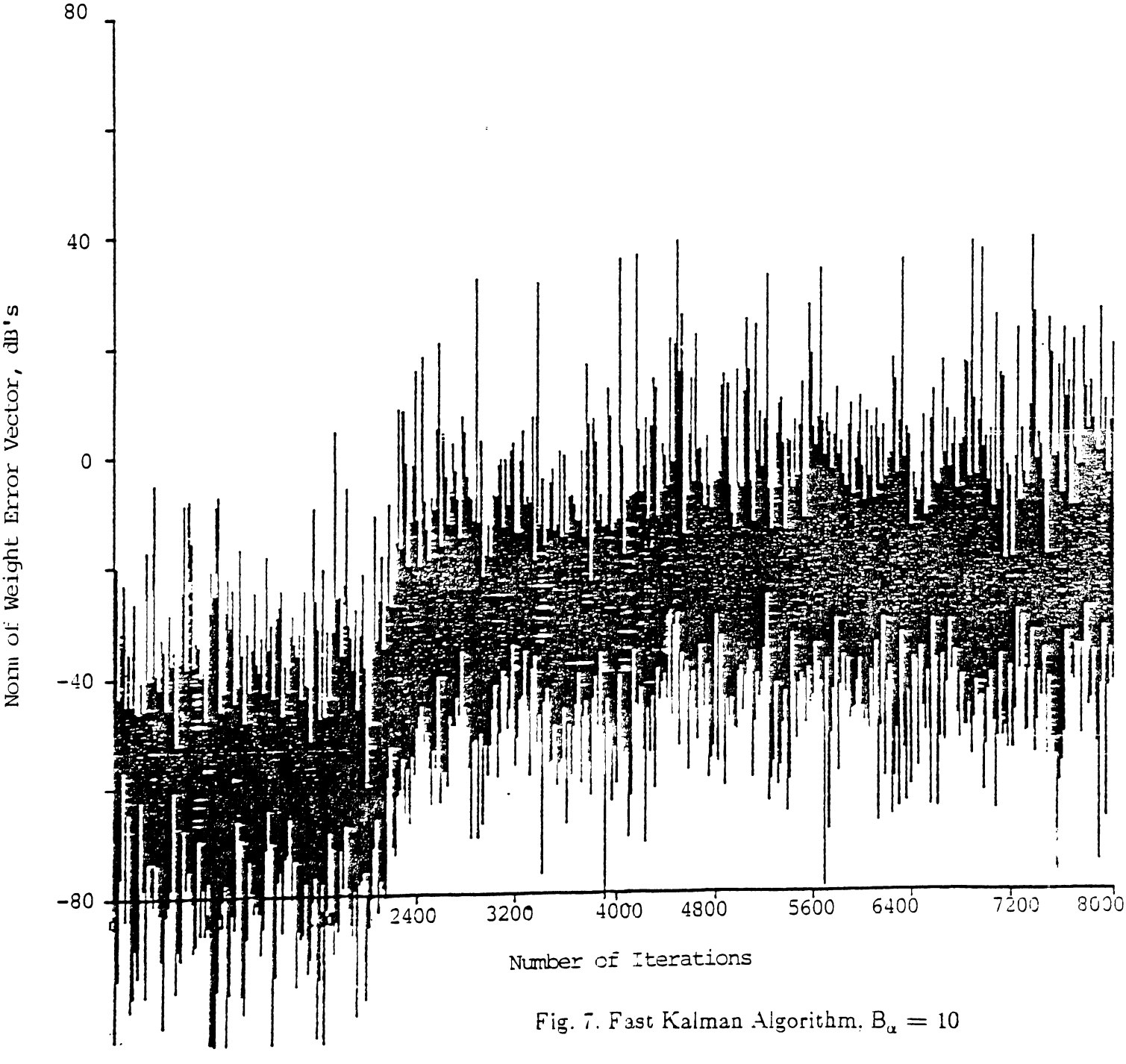

Fig. 7. Fast Kalman Algorithm, $B_\alpha = 10$

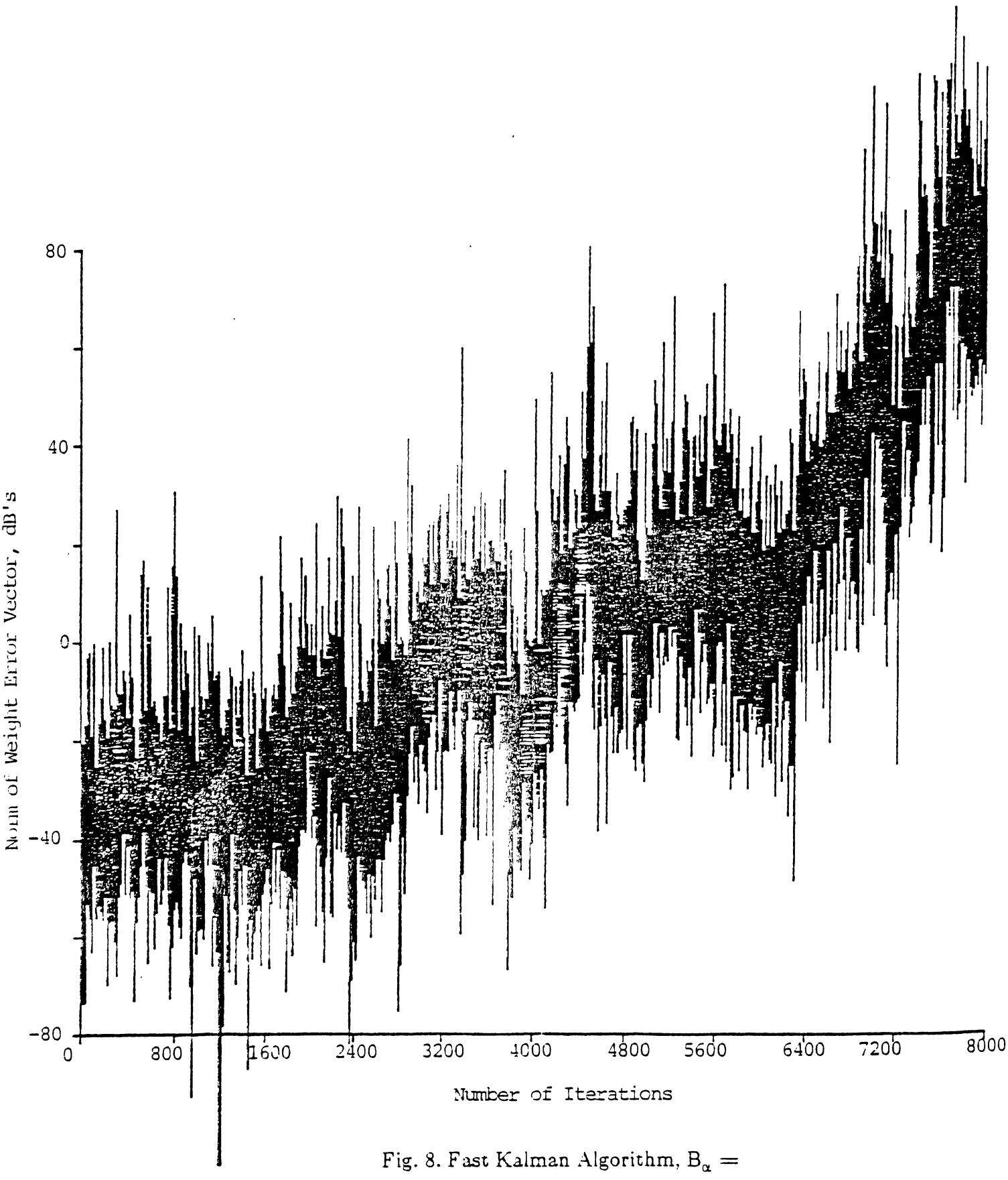

Fig. 8. Fast Kalman Algorithm, $B_\alpha =$

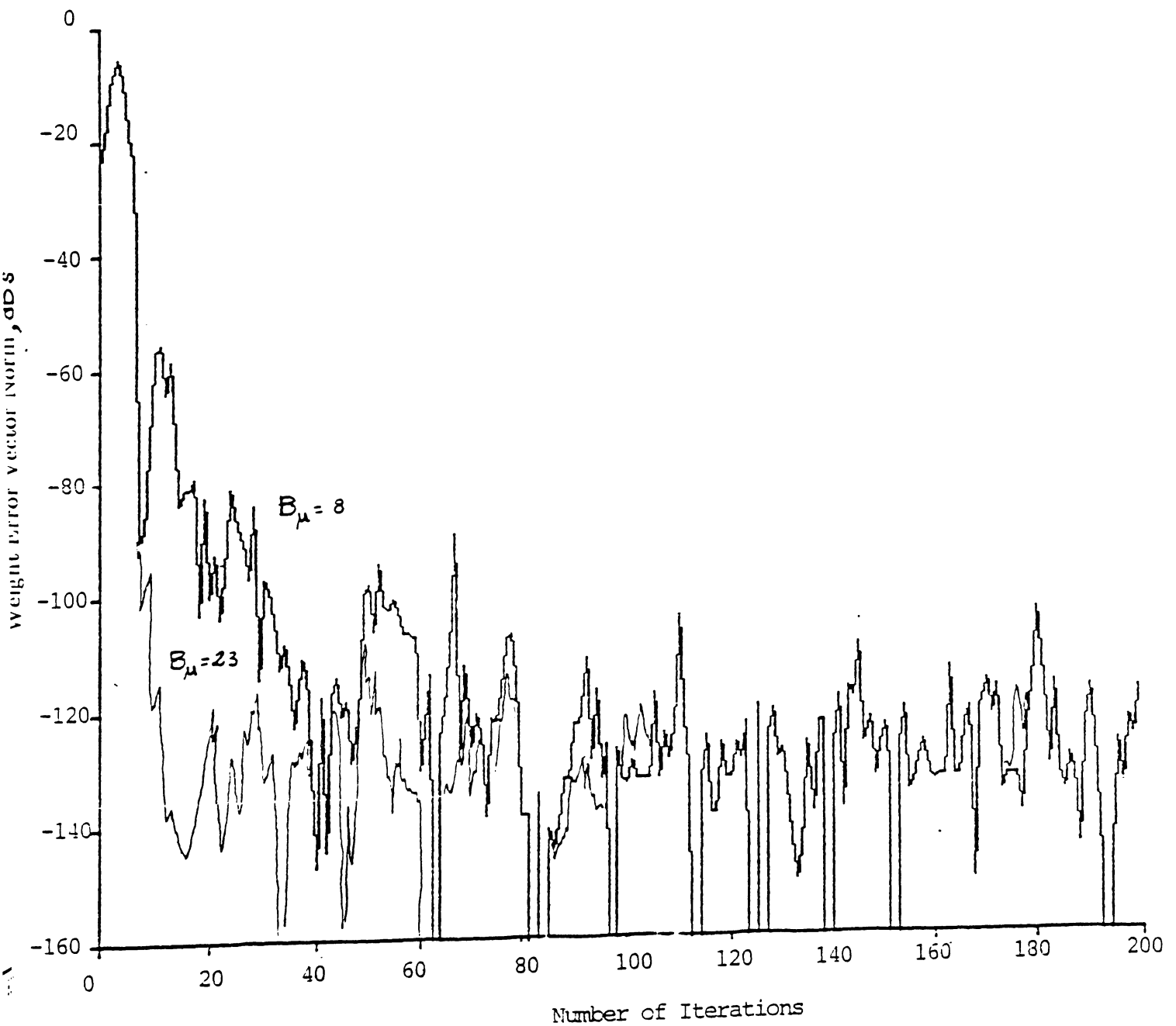

Fig. 9. Fast Kalman Algorithm, $B_\mu = 23$ and 8.

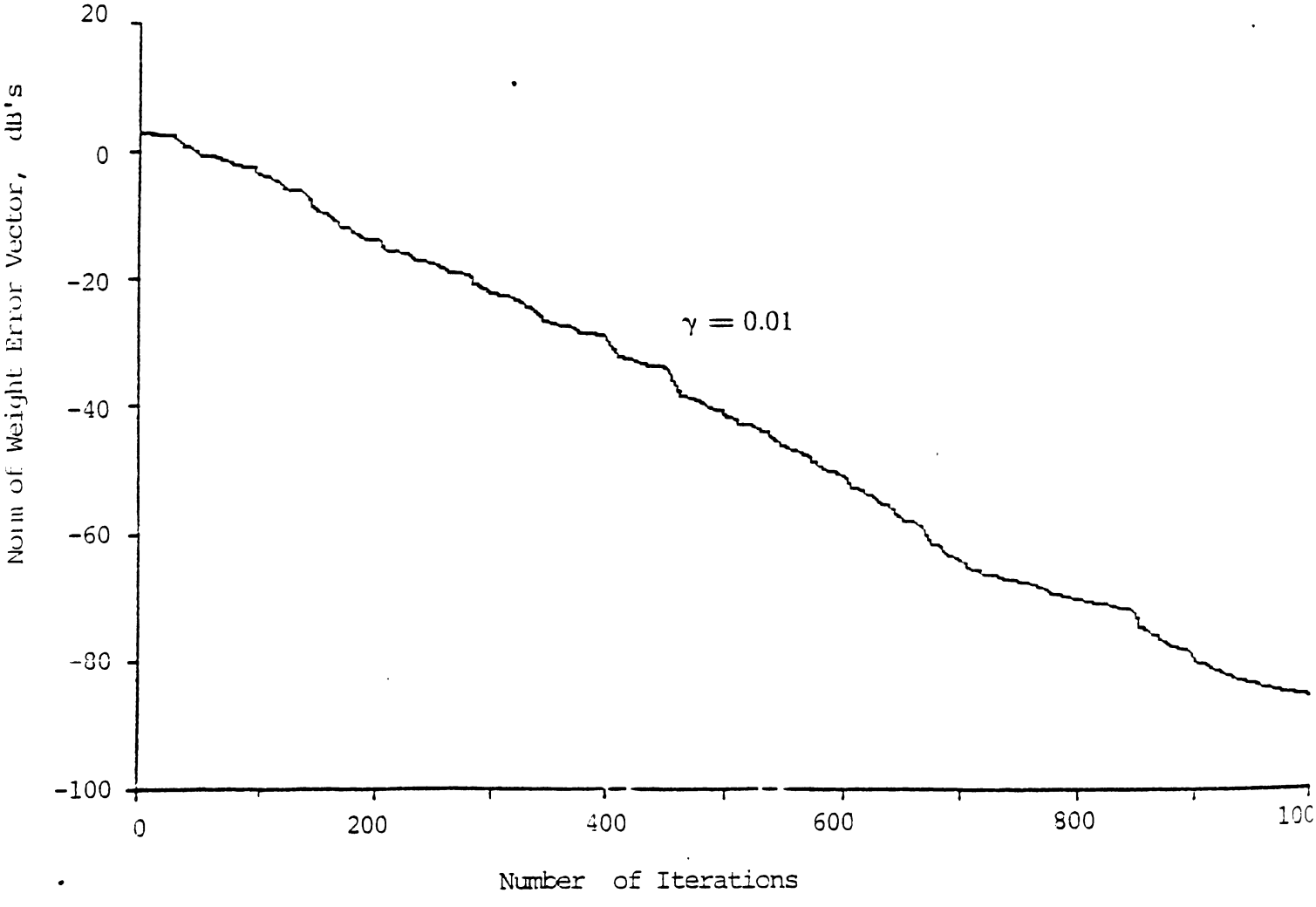

Fig. 10. LMS Algorithm. $B_\mu = 23$ and 3

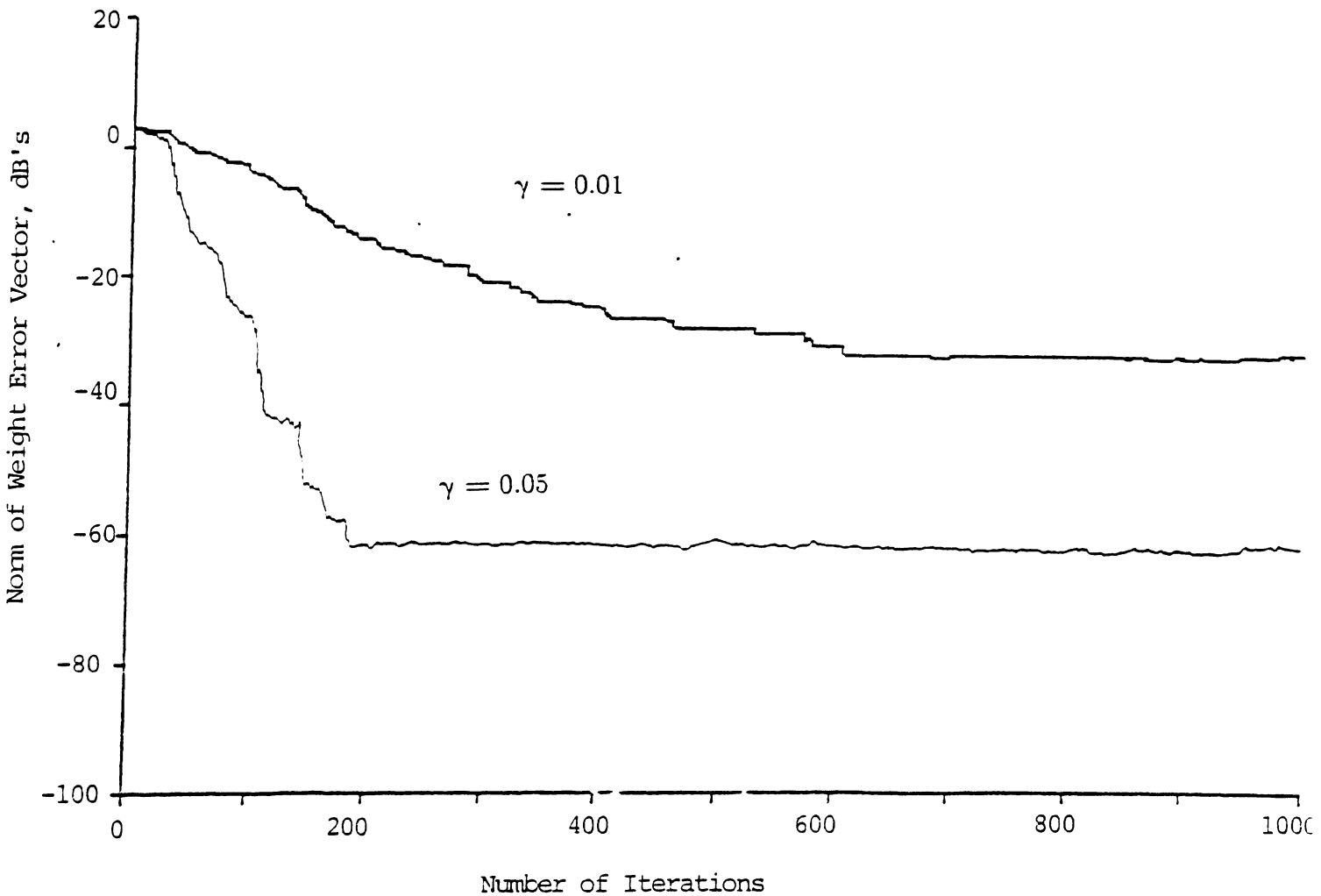

Fig. 11. LMS Algorithm. Effects of Loop Gain γ . With $B_\alpha = 3$

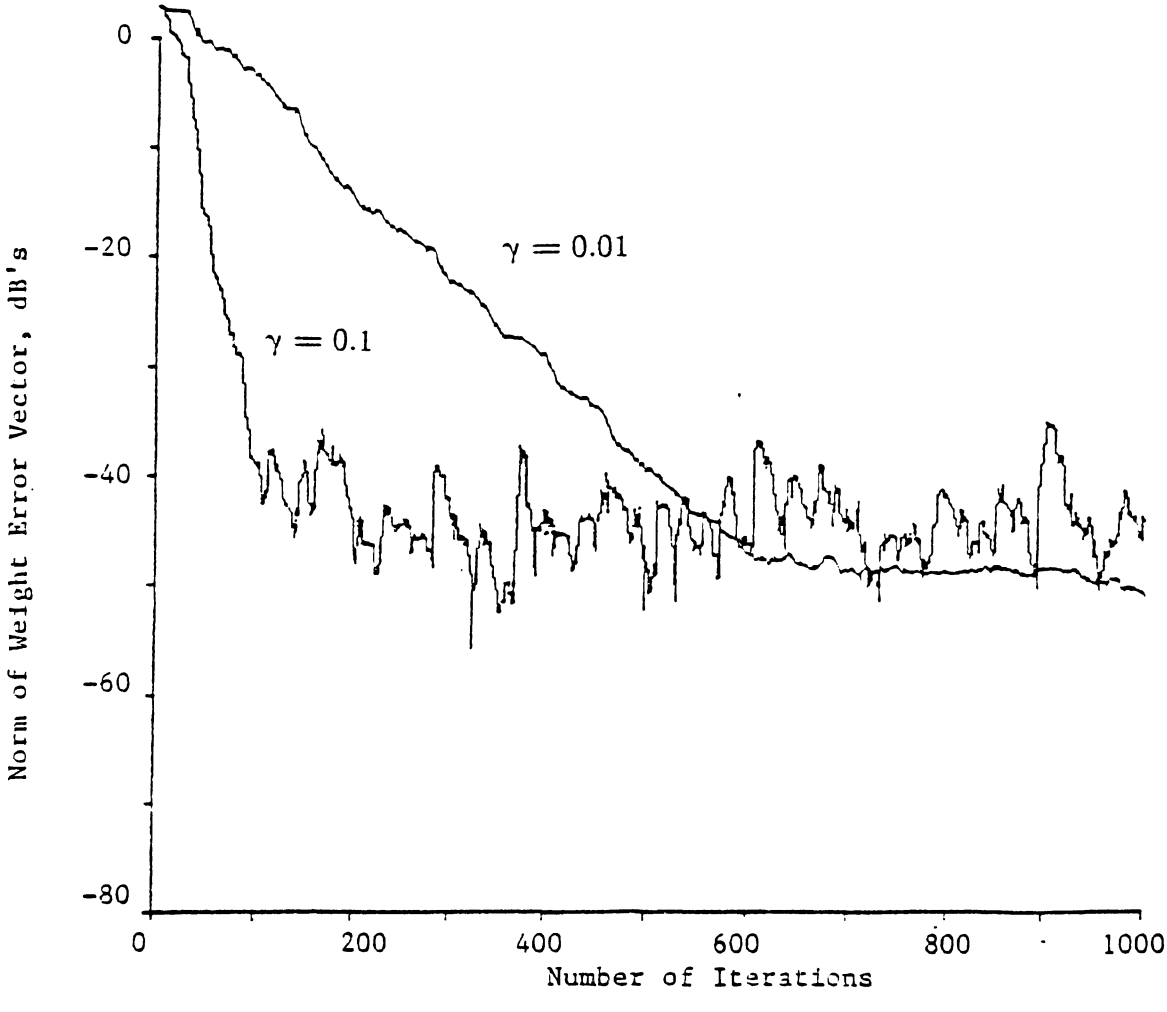

Fig. 12. LMS Algorithm, Effects of Loop Gain γ, With $B_\eta = 8$