COMPUTATIONALLY EFFICIENT ALGORITHMS FOR THIRD ORDER ADAPTIVE VOLTERRA FILTERS

Xiaohui Li and W. Kenneth Jenkins
Dept. of Electrical and Computer Engr. and
The Coordinated Science Laboratory
University of Illinois
Urbana, IL

Charles W. Therrien
Dept. of Electrical and Computer Engr.
Naval Postgraduate School
Monterey, CA

Abstract - The input autocorrelation matrix for a third order (cubic) Volterra adaptive filter for general colored Gaussian input processes is analyzed to determine how to best formulate a computationally efficient fast adaptive algorithm. When the input signal samples are ordered properly within the input data vector, the autocorrelation matrix of the cubic filter inherits a block diagonal structure, with some of the sub-blocks also having diagonal structure. A computationally efficient adaptive algorithm is presented that takes advantage of the sparsity and unique structure of the correlation matrix that results from this formulation.

1. INTRODUCTION

Recently it was shown that for quadratic Volterra adaptive filters with Gaussian input signals, a computationally efficient adaptive algorithm can be developed by exploiting the special structure of the input autocorrelation matrix that results from properly ordering the input samples within the input data vector [2]. However, in many applications it may be necessary to include a the third order nonlinear term in the Volterra expansion to accurately model realistic nonlinearities. This paper develops a computationally efficient adaptive update algorithm for the third order case by extending the analysis and development that was previously shown to be effective in reducing computational complexity in the second order case.

The third order Volterra filter of finite memory is defined as:

$$y(n) = \sum_{m_1=0}^{N-1} h_1(m_1)x(n-m_1) + \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2(m_1,m_2)x(n-m_1)x(n-m_2) + \sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1}\sum_{m_3=m_2}^{N-1} h_3(m_1,m_2,m_3)x(n-m_1)x(n-m_2)x(n-m_3) \quad (1)$$

where N is the memory length of the filter. The filter output is simply the linear combination of its linear input signals and the second and the third order cross products of its linear input signals. Because of the linear relationship between the filter output and the filter coefficients, many LMS-based adaptive algorithms for linear adaptive filtering can be used for the third order adaptive Volterra filter. By formulating the second and the third order nonlinear terms, the eigenvalue spread of the autocorrelation matrix of the third order filter input is increased dramatically. Because of this, many LMS based linear adaptive algorithms are not very effective in increasing the convergence speed. Also, because the matrix is non-Toeplitz, it is even more difficult to develop a fast adaptive algorithm which generally requires calculating. either explicitly or implicitly, the inverse of the autocorrelation matrix.

The third order Volterra filter can be represented in vector format as:

$$y(n) = \mathbf{W}^T(n)\mathbf{X}(n), \quad (2)$$

where the vector $\mathbf{X}(n)$ contains the N linear terms of the input signal x(n), the second order and the third order nonlinear terms generated from the N linear input signals of x(n). We define the following data vectors:

$$\mathbf{x}_l(n) = [\, x(n) \;\; x(n-1) \;\cdots\; x(M+1)\,]^T \quad (3)$$

$$\mathbf{x}_2(n) = [\, x^2(n) \;\; x^2(n-1) \;\cdots\; x^2(M+1)\,]^T \quad (4)$$

$$\mathbf{x}_3(n) = [\, x^3(n) \;\; x^3(n-1) \;\cdots\; x^3(M+1)\,]^T \quad (5)$$

$$\mathbf{x}_{2c}(n) = [\, x(n)x(n-1) \;\cdots\; x(M+2)x(M+1)]^T \quad (6)$$

$$\mathbf{x}_{3c}(n) = [\, x(n)x(n-1)x(n-2) \cdots x(M+3)x(M+2)x(M+1)]^T \quad (7)$$

$$\mathbf{x}_{sqc}(n) = [\, x^2(n)x(n-1) \;\; x^2(n)x(n-1) \cdots x^2(M+2)x(M+1)]^T \quad (8)$$

where M = n - M. It can be shown that the data vectors defined in (3.3) - (3.8) contain all the elements required for the third order Volterra filter input vector $\mathbf{X}(n)$. Therefore, $\mathbf{X}(n)$ can be represented as:

$$\mathbf{X}(n) = [\, \mathbf{x}_l^T(n) \;\; \mathbf{x}_3^T(n) \;\; \mathbf{x}_{sqc}^T(n) \;\; \mathbf{x}_2^T(n) \;\; \mathbf{x}_{2c}^T(n) \;\; \mathbf{x}_{3c}^T(n)\,]^T.$$

For zero mean and independently distributed Gaussian signal it can be shown that the autocorrelation matrix of the third order Volterra filter input equals:

$$\mathbf{R}_X = \begin{bmatrix} \mathbf{R}_l & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{sq} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_c \end{bmatrix} \quad (9)$$

where $\mathbf{R}_{sq}=E[\,\mathbf{x}_2(n)\mathbf{x}_2^T(n)]$. $\mathbf{R}_t$ is defined as:

$$\mathbf{R}_l = E[\mathbf{x}_l\mathbf{x}_l^T] \quad (10)$$

where $\mathbf{x}_f(n) = [\,\mathbf{x}_1^T(n)\ \ \mathbf{x}_3^T(n)\ \ \mathbf{x}_{sqc}^T(n)\,]^T$. $\mathbf{R}_c$ is the block diagonal matrix defined as:

$$\mathbf{R}_c = E[\mathbf{x}_c(n)\mathbf{x}_c^T(n)], \tag{11}$$

where $\mathbf{x}_c(n) = [\,\mathbf{x}_{2c}^T(n)\ \ \mathbf{x}_{3c}^T(n)\,]^T$. It is straightforward to show that the sub-matrix $\mathbf{R}_c$ is simply a diagonal matrix. Furthermore, the matrix $\mathbf{R}_{sq}$ is an NxN non-Toeplitz matrix, and $\mathbf{R}_f$ is an $(N^2+N)\times(N^2+N)$ non-Toeplitz matrix.

The above analysis shows that for white Gaussian input signals, the autocorrelation matrix is partially decoupled and has a block diagonal matrix. Recalling that an uncorrelated Gaussian signal is also independently distributed, we might expect that the use of a linear transform to decorrelate a colored Gaussian input signal will lead to an efficient 3rd order adaptive update algorithm with decreased complexity by providing a more completely decoupled covariance matrix structure. Another important issue is a large difference in eigenvalue spreads between autocorrelation matrices of the Volterra filter input constructed from colored and white Gaussian signals, respectively. We have compared the eigenvalue spreads of two matrices under different scenarios. The results consistently show that a Volterra filter input constructed from a colored Gaussian input has an eigenvalue spread that is much higher than that of a filter with a white Gaussian input signal.

Based on the above discussion we propose a structure for the 3rd order adaptive Volterra filter that is shown in Fig. 1. For a colored Gaussian input signal x(n), $\mathbf{x}(n)$ is first linearly transformed, and each of the outputs of the linear transform is normalized by its own signal power. The outputs of power normalization are then used to construct the inputs for 3rd order Volterra filter. Assuming the linear transform can perfectly decorrelate the input $\mathbf{x}(n)$ and the power normalization is done appropriately, the components of the preprocessed signal becomes i.i.d. Gaussian signals. By constructing the inputs of the Volterra filter from these preprocessed signals, the input autocorrelation matrix of 3rd order Volterra filter is not only decoupled, but also its eigenvalue spread is decreased dramatically.

2. QUASI-NEWTON METHODS

In this section a fast algorithm for the 3rd order adaptive Volterra filter is developed based on the quasi-Newton method. The algorithm exploits the special structure of the 3rd order adaptive Volterra filter introduced in last section. In the following derivation, we assume the linear transform can perfectly decorrelate the input, so that for analysis purposes the input to the Volterra filter can be analyzed as an i.i.d. Gaussian signal.

To achieve fast convergence, the quasi-Newton algorithm [2]

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu\,\mathbf{R}_{\mathbf{X}}^{-1}(n)\mathbf{X}(n)e(n) \tag{12}$$

is used to update the filter coefficients. The main task here is to simplify the update of the Kalman gain $\mathbf{R}_{\mathbf{X}}^{-1}(n)\mathbf{X}(n)$. As shown in last section, for i.i.d. Gaussian input, the matrix $\mathbf{R}_{\mathbf{X}}$ is a block diagonal matrix. So the updating equation (12) can be decoupled into

$$\begin{aligned}
\mathbf{W}_f(n+1) &= \mathbf{W}_f(n) + \mu\mathbf{R}_f^{-1}(n)\mathbf{X}_f(n)e(n)\\
\mathbf{W}_{sq}(n+1) &= \mathbf{W}_{sq}(n) + \mu\mathbf{R}_{sq}^{-1}(n)\mathbf{X}_2(n)e(n)\\
\mathbf{W}_c(n+1) &= \mathbf{W}_c(n) + \mu\mathbf{R}_c^{-1}(n)\mathbf{X}_c(n)e(n)
\end{aligned} \tag{13}$$

Because the autocorrelation matrices $\mathbf{R}_f(n)$, $\mathbf{R}_{sq}(n)$, and $\mathbf{R}_c(n)$ have different structures, the Kalman gain vector associated with each of these autocorrelation matrices should be updated differently. The basic adaptive algorithm based on the quasi-Newton method for the 3rd adaptive Volterra filter can be summarized as:

- **Step 0**: Initialization (n=0)
  $\mathbf{W}(n) = [\mathbf{w}_f^T(n)\ \ \mathbf{w}_{sq}^T(n)\ \ \mathbf{w}_c^T(n)\,]^T = \mathbf{0}$
- **Step 1**:
  $$\mathbf{q}(n) = \mathbf{T}\mathbf{x}(n)$$
  $$\sigma_{q,i}^2(n) = \alpha\sigma_{q,i}^2(n-1) + (1-\alpha)q_i^2(n) \quad for\ i = 0\ to\ N-1$$
  $$x_i(n) = q_i(n)/\sigma_{q,i}(n) \quad for\ i = 0\ to\ N-1$$
- **Step 2**: Construct 3rd order Volterra filter input, $\mathbf{x}_f(n)$, $\mathbf{x}_2(n)$, and $\mathbf{x}_c(n)$ from $\mathbf{x}_f(n)$.
  $$\mathbf{X}(n) = [\mathbf{x}_f^T(n)\ \ \mathbf{x}_2^T(n)\ \ \mathbf{x}_c^T(n)\,]^T = \mathbf{0}$$
  $$\mathbf{W}(n) = [\mathbf{w}_f^T(n)\ \ \mathbf{w}_{sq}^T(n)\ \ \mathbf{w}_c^T(n)\,]^T$$
  $$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n)$$
- **Step 3**:
  (1) Update $\mathbf{R}_f^{-1}(n)\mathbf{X}_f(n)$ and $\mathbf{w}_f(n)$.
  (2) Update $\mathbf{R}_{sq}^{-1}(n)\mathbf{X}_2(n)$ and $\mathbf{w}_2(n)$.
  (3) Update $\mathbf{R}_c^{-1}(n)\mathbf{X}_c(n)$ and $\mathbf{w}_c(n)$.
- **Step 4**: n=n+1, *goto Step 1.*

The main task in updating the filter coefficients is to update the three Kalman gain vectors listed in Step 3. Since $\mathbf{R}_c$ is simply a diagonal matrix, the updating equation for $\mathbf{W}_c(n)$ becomes

$$w_{c,k}(n+1) = w_{c,k}(n) + \mu\frac{1}{\sigma_{c,k}^2}x_{c,k}(n)e(n), \tag{14}$$

where $\sigma_{c,k}^2 = E[x_{c,k}^2(n)]$ is the power estimate of the k-th element of $\mathbf{x}_c(n)$. If the input signal x(n) is stationary, $\mathbf{R}_s(n)$ consists of only two numerical values, one which appears along the main diagonal, and one which defines the off diagonal elements. An efficient algorithm for updating the Kalman gain vector $\mathbf{R}_{sq}^{-1}(n)\mathbf{X}_2(n)$ has been developed in [2] for the 2nd order adaptive Volterra filter. The algorithm can be directly applied to update the Kalman gain for 3rd order adaptive Volterra filter.

The matrix $\mathbf{R}_f$ is an $(N^2+N)\times(N^2+N)$ non-Toeplitz matrix. It requires $O(N^6)$ multiplications to update the

Kalman gain vector $\mathbf{R}_t^{-1}(n)\mathbf{X}_t(n)$ if traditional methods of calculating matrix inverse are utilized. The fast quasi-Newton algorithm, which utilizes the Toeplitz structure of the autocorrelation matrix, is simply not applicable here. The Conjugate Gradient method has been used recently [3] as an updating algorithm for the Kalman gain vector in adaptive filtering. The algorithm does not exploit any special structure of the autocorrelation matrix in calculating the Kalman gain, so it can be effectively applied to the nonlinear adaptive filter where the input autocorrelation matrix is non-Toeplitz.

2.1 The Conjugate Gradient Algorithm

The conjugate gradient algorithm is an iterative search method to minimize the quadratic cost function:

$$\nabla(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{R}\mathbf{w} - \mathbf{w}^T\mathbf{b}, \tag{15}$$

where $\mathbf{R}$ is an N×N symmetric positive definite matrix, and $\mathbf{w}$ and $\mathbf{b}$ are N×1 vectors. The optimum solution of the quadratic problem is:

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{b} \tag{16}$$

The conjugate gradient algorithm can be used to calculate the Kalman gain vector. In searching for the optimum solution at each iteration, the conjugate gradient algorithm searches through a set of N linearly independent direction vectors, $\mathbf{p}_0$, $\mathbf{p}_1$, ..., $\mathbf{p}_{N-1}$, which are $\mathbf{R}$-conjugate and linearly independent. Because these direction vectors span $\mathbf{R}^N$ and the range of the matrix $\mathbf{R}$, the solution of the optimization problem (15), $\mathbf{w}^*$, can be achieved after at most N iterations and can be expressed as the linear combination of the N direction vectors:

$$\mathbf{w}^* = \alpha_0\mathbf{p}_0 + \alpha_1\mathbf{p}_1 + \cdots + \alpha_{N-1}\mathbf{p}_{N-1} \tag{17}$$

It is demonstrated in [4] that the conjugate gradient algorithm can converge to the optimum solution faster if the matrix $\mathbf{R}$ is well conditioned or is a lower rank perturbation to identity matrix. For a linear system, the preconditioned conjugate gradient method tries to increase the convergence rate of the algorithm. It applies the normal conjugate gradient algorithm to solve the linear system equation:

$$\tilde{\mathbf{R}}\tilde{\mathbf{w}}^* = \tilde{\mathbf{b}} \tag{18}$$

where $\tilde{\mathbf{R}} = \mathbf{C}^{-1}\mathbf{R}\mathbf{C}^{-1}$, $\tilde{\mathbf{w}} = \mathbf{C}\mathbf{w}$, and $\tilde{\mathbf{b}} = \mathbf{C}^{-1}\mathbf{b}$. $\mathbf{C}$ is a symmetric positive definite matrix. It can be observed that if $\mathbf{C}$ is chosen such that the matrix $\tilde{\mathbf{R}}$ is well conditioned, then the conjugate gradient algorithm will take less iterations to converge to the optimum solution $\tilde{\mathbf{w}}^* = \mathbf{R}^{-1}\tilde{\mathbf{b}}$. The preconditioned conjugate gradient algorithm can be found in [4]. From the algorithm it can be seen that the preconditioner, $\mathbf{M} = \mathbf{C}^2$, should also be chosen such that it allows efficient computation of $\mathbf{z}_k = \mathbf{M}^{-1}\mathbf{r}_k$.

2.2 Preconditioned Conjugate Gradient Algorithm For 3rd Order Adaptive Volterra Filter

For the third order adaptive Volterra filter, the autocorrelation matrix $\mathbf{R}_t(n)$ is non-Toeplitz structured. Most preconditioners developed previously do not perform well here as these preconditioners are developed for the linear system $\mathbf{R}\mathbf{w} = \mathbf{b}$ where $\mathbf{R}$ has Toeplitz structure. One intuitive way of constructing a preconditioner $\mathbf{C}$ is to choose $\mathbf{M} = \mathbf{C}^2$ to be close to $\mathbf{R}$. This can be justified by the Incomplete Cholesky preconditioner design [4]. In this method the preconditioner is chosen as $\mathbf{M}=\mathbf{H}\bullet\mathbf{H}^T=\mathbf{C}^2$ where $\mathbf{H}$ is a matrix with certain sparse structure and is close to the Cholesky factor $\mathbf{G}$ of the matrix $\mathbf{R}$. That is $\mathbf{R}$ and $\mathbf{G}$ satisfy:

$$\mathbf{R} = \mathbf{G} * \mathbf{G}^T \tag{19}$$

The better the matrix $\mathbf{H}$ approximates $\mathbf{G}$, the closer the transformed matrix $\tilde{\mathbf{R}} = \mathbf{C}^{-1}\mathbf{R}\mathbf{C}^{-1}$ will be to the identity matrix. This shows that a good preconditioner can be chosen to approximate the matrix $\mathbf{R}$.

For the third order adaptive Volterra filter, the autocorrelation matrix $\mathbf{R}_t$ is represented as:

$$\mathbf{R}_t = \begin{bmatrix} E[\mathbf{x}_l\mathbf{x}_l^T] & E[\mathbf{x}_l\mathbf{x}_3^T] & E[\mathbf{x}_l\mathbf{x}_{2c1c}^T] \\ E[\mathbf{x}_3\mathbf{x}_l^T] & E[\mathbf{x}_3\mathbf{x}_3^T] & E[\mathbf{x}_3\mathbf{x}_{2c1c}^T] \\ E[\mathbf{x}_{2c1c}\mathbf{x}_l^T] & E[\mathbf{x}_{2c1c}\mathbf{x}_3^T] & E[\mathbf{x}_{2c1c}\mathbf{x}_{2c1c}^T] \end{bmatrix} \tag{20}$$

We want to use the sub-matrices along the diagonal of $\mathbf{R}_t$ to build the preconditioner. The idea is to retain as many characteristics of $\mathbf{R}$ as possible, while at the same time, making it easy to calculate the inverse of the $\mathbf{M}$. In this paper, two preconditioners are used to calculate $\mathbf{R}_t^{-1}(n)\mathbf{x}_t(n)$ for 3rd order adaptive Volterra filter. The first has the following format:

$$M_1 = \begin{bmatrix} E[\mathbf{x}_l\mathbf{x}_l^T] & E[\mathbf{x}_l\mathbf{x}_3^T] & 0 \\ E[\mathbf{x}_3\mathbf{x}_l^T] & E[\mathbf{x}_3\mathbf{x}_3^T] & \vdots \\ 0 & \cdots & E[\mathbf{x}_{2c1c}\mathbf{x}_{2c1c}^T] \end{bmatrix} \tag{21}$$

This preconditioner is simply the autocorrelation matrix $\mathbf{R}_t$ but with its off diagonal elements $E[\mathbf{x}_l\mathbf{x}_{2c1c}^T]$ and $E[\mathbf{x}_3\mathbf{x}_{2c1c}^T]$ set to zeros. For white Gaussian input signals, the sub-matrix in the upper-left corner of the above matrix is:

$$\begin{bmatrix} E[\mathbf{x}_l\mathbf{x}_l^T] & E[\mathbf{x}_l\mathbf{x}_3^T] \\ E[\mathbf{x}_3\mathbf{x}_3^T] & E[\mathbf{x}_3\mathbf{x}_3^T] \end{bmatrix}$$

Since its sub-matrices $E[\mathbf{x}_l\mathbf{x}_l^T]$, $E[\mathbf{x}_3\mathbf{x}_3^T]$ and $E[\mathbf{x}_l\mathbf{x}_3^T]$ are diagonal matrices, it is computationally simple to calculate the inverse of above sub-matrix. The sub-matrix

$E[\mathbf{x}_{2c1c}\mathbf{x}_{2c1c}^T]$ of M_1 is a non-Toeplitz matrix, so it is still computationally intensive to calculate its inverse.

The second preconditioner used in this paper is a simplified version of the first preconditioner. Define the signal vector $\mathbf{x}_{2c1c}(n)$ as:

$$\mathbf{x}_{2c1c}(n) = [\mathbf{x}_{2c,0}^T(n), \mathbf{x}_{2c,1}^T(n), \quad \cdots, \mathbf{x}_{2c,N-1}^T(n)]^T$$

where $\mathbf{x}_{2c,i}(n)$ is defined as:

$$\mathbf{x}_{2c,i}(n) = x^2(n-i)[\, x(n)\, x(n-1) \cdots \\ \cdots x(n-i+1)\, x(n-i-1) \cdots x(n-N+1)]$$

The second preconditioner is defined as:

$$M_2 = \begin{bmatrix} E[\mathbf{x}_l\mathbf{x}_l^T] & E[\mathbf{x}_l\mathbf{x}_3^T] & 0 & \cdots & 0 \\ E[\mathbf{x}_3\mathbf{x}_l^T] & E[\mathbf{x}_3\mathbf{x}_3^T] & & & \\ 0 & & E[\mathbf{x}_{2c,0}\mathbf{x}_{2c,0}^T] & \cdots & \vdots \\ \vdots & & 0 & \cdots & 0 \\ 0 & & \cdots & 0 & E[\mathbf{x}_{2c,N-1}\mathbf{x}_{2c,N-1}^T] \end{bmatrix}$$

Apparently only the N sub-autocorrelation matrices along the diagonal of $E[\mathbf{x}_{2c1c}\mathbf{x}_{2c1c}^T]$ are used as part of the preconditioner. For zero mean white Gaussian input signal, the above matrix can be expressed as:

$$M_2 = \begin{bmatrix} E[\mathbf{x}_l\mathbf{x}_l^T] & E[\mathbf{x}_l\mathbf{x}_3^T] & 0 & \cdots & 0 \\ E[\mathbf{x}_3\mathbf{x}_l^T] & E[\mathbf{x}_3\mathbf{x}_3^T] & & & \\ 0 & & E[x^6] & 0 & \vdots \\ \vdots & & 0 & \cdots & 0 \\ 0 & & \cdots & 0 & E[x^6] \end{bmatrix} \quad (21)$$

So this preconditioner is a block diagonal matrix. The first matrix along the diagonal is a tridiagonal matrix. The second matrix is simply a diagonal matrix. So the calculation of the inverse of M_2 is very much simplified.

3. COMPUTER EXPERIMENTS

Computer simulation was used to evaluate the performance of the 3rd order Volterra adaptive filtering algorithm introduced here. The filter is used to identify a 3rd order nonlinear system. The input to the system is a colored Gaussian input signal generated using a 7th order lowpass filter. At each update instance one iteration, two iterations, or the full number of iterations of the preconditioned conjugate gradient algorithm is applied in calculating $\mathbf{R}_l^{-1}(n)\mathbf{x}_l(n)$, using the second preconditioner described in the previous section. Figure 2 shows the experimental results, where, for comparison, the learning curve of the LMS filter is included. From the results it is seen that the new adaptive algorithm shows dramatic improvement in convergence rate in comparison to the LMS algorithm. The results also illustrate that the chosen preconditioner chosen is very effective in improving the convergence rate.

ACKNOWLEDGMENT

This work is supported by the Joint Services Electronics Program (JSEP) under contract number N00014-96-J-0129.

REFERENCES

[1] X. Li, W. K. Jenkins, and C. W. Therrien, "Algorithms for improved performance in adaptive polynomial fillers with Gaussian input signals," *Proc. of the Asilomar Conference on Signals, Systems, and Computers,* Pacific Grove, CA, November 1996.

[2] D. Marshall and W. Jenkins, "A fast quasi-Newton adaptive filtering algorithm," *IEEE Trans. Signal Proc.*, Vol. 40, no. 7, pp. 1652-1662, Jul. 1992.

[3] A. W. Hull, "Orthogonalization techniques of Toeplitz filters," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1994.

[4] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins Univ. Press, Baltimore, MD.

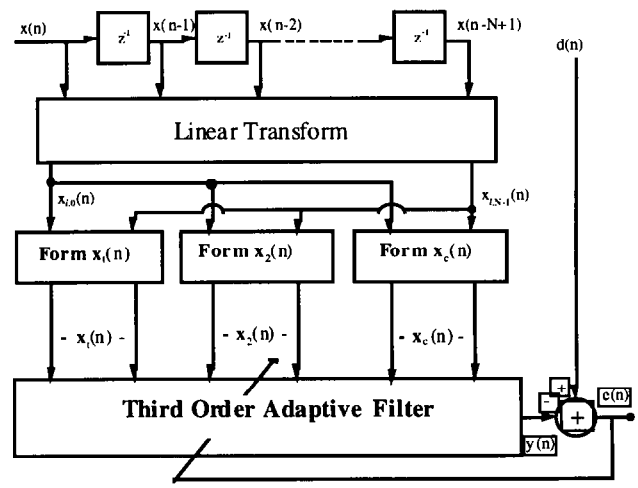

Fig. 1 Block diagram of the third order Volterra filter.

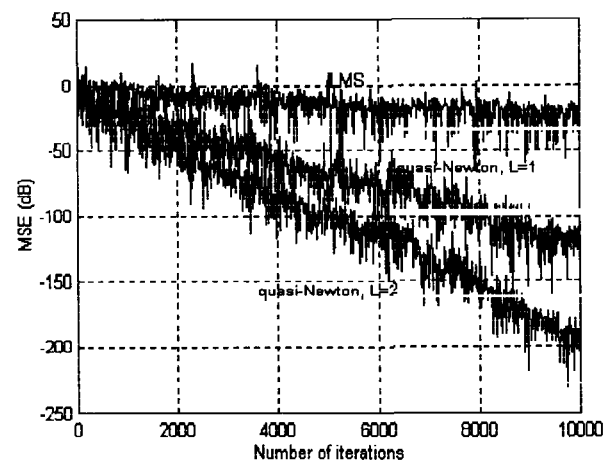

Fig. 2 Experimental evaluation of the new algorithm.