International Journal of Control Science and Engineering 2014, 4(2): 27-35
DOI: 10.5923/j.control.20140402.01

Adaptive Diagonal-Channel Bilinear Filters for Nonlinear Active Noise Control

Li Tan[1,*], Jean Jiang[1], Liangmo Wang[2]

[1]College of Engineering and Technology, Purdue University North Central, Westville, Indiana, USA
[2]School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing, China

Abstract This paper proposes a novel adaptive bilinear filter with a diagonal-channel structure for nonlinear active noise control. Using the diagonal-channel structure, the diagonal-channel bilinear filtered-X least mean square (DBFXLMS) and recursive least square (DBFXRLS) algorithms are derived. In order to reduce the computational load for DBFXRLS algorithm, the DBFXRLS algorithm with a sequential channel update (DBFXRLS-SEQ) is proposed. Computational complexity for each algorithm is analyzed. Computer simulations demonstrate the control performance improvement of the proposed algorithms.

Keywords Active noise control (ANC), Nonlinear adaptive filter, Adaptive bilinear filter, Adaptive Volterra filter, Diagonal-channel structure

1. Introduction

Active noise control (ANC) has shown effectiveness for controlling disturbance noise, such as noises from engines, blowers, fans, transformers, and compressors [1-5]. The noise types may include low-frequency noise, broadband noise, chaotic noise, and so on. Active noise control system using a linear adaptive controller with a low computational cost is successfully applied. Over the decades, the advances of digital signal processing (DSP) technology have inspired further research in nonlinear active noise control [6-15]. Due to the increase of the speed of DSP processors and the decrease of their costs, many advanced active noise control algorithms may be developed to tackle nonlinear problems in ANC systems and implemented to gain better noise control performance, including the recent development of nonlinear system control with sliding mode techniques [16-17] to solve more general nonlinear problems in the control systems.

Active noise control applies the principle of superposition of the primary noise source and the secondary source with its acoustic output being same amplitude and opposite phase of the primary noise source so that a quiet zone can be achieved at the cancelling point. The secondary source is produced by an adaptive filter using a filtered-X adaptive algorithm such as the least mean square (LMS) or the recursive least square (RLS) algorithm. A typical single-channel ANC system using a feed-forward controller scheme is shown in Figure 1.

* Corresponding author:
lizhetan@pnc.edu (Li Tan)
Published online at http://journal.sapub.org/control

The ANC system uses the reference signal $x(n)$ (from the reference microphone or sensor) to generate a cancelling signal $y(n)$ via a linear adaptive control filter whose transfer function is denoted by $W(z)$. The linear adaptive controller drives the secondary speaker to destructively attenuate the primary noise $d(n)$. An error microphone is used to detect the error signal $e(n)$. The error signal $e(n)$ and filtered reference signal $x'(n)$ are used by the adaptive algorithm to produce the controller coefficients. Note that $P(z)$ denotes the primary path between the reference sensor and the error microphone, $S(z)$ the secondary path between the ANC adaptive filter output and the error microphone, and $\overline{S}(z)$ the secondary path estimate which is used to generate the filtered reference signals for the adaptive algorithm.

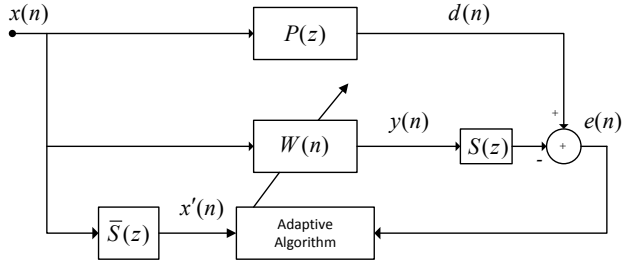

Figure 1. Block diagram of the single channel ANC system using a filtered-x adaptive algorithm

It is well known that linear controllers, especially the linear feed-forward controllers, have been successfully

applied to actively cancel the low-frequency noise [1-5]. However, if a noise process to be controlled is a nonlinear and deterministic process such as chaotic noise, or if the primary as well as secondary paths in an ANC system may exhibit the nonlinear behavior, or if the reference signal may have the saturation effect [9-15], the linear ANC system will suffer from performance degradation by using linear controllers. In order to tackle various nonlinear effects, many researchers have developed nonlinear noise control systems using neural networks, normalized Gaussian radial basis function networks, Volterra filtered-X LMS and RLS algorithms, filtered-s LMS, and bilinear filtered-X LMS algorithms [6-15]. The Volterra filtered-X adaptive algorithms [9-10] apply all of the possible products of the delayed input signals to construct nonlinear inputs for the feed-forward controller. The filtered-s algorithm proposed by Das and Panda [11] uses sine and cosine base functions according to the functional-link artificial neural network (FLANN) model to construct the nonlinear inputs. In fact, the nonlinear inputs can be produced using a generalized approach called the function expansion method [12, 13]. To reduce the number of filter coefficients, especially, for adaptive Volterra filters, Kuo and Wu [14] developed a bilinear filtered-X LMS (BFXLMS) algorithm. The developed BFXLMS algorithm is very effective for narrow band noise control when the adaptive bilinear filter is stable during the adaptation and the BFXLMS algorithm converges to the global minimum of the error function. In Kuo and Wu's original work, the diagonal-channel structure feature described in [18] for the cross terms of delayed input signal and delayed output signal was not considered. When taking into account for the diagonal-channel structure, various improved adaptive bilinear filtered-X algorithms can be developed. The initial results are reported in [15].

In this paper, Section 2 will first develop a diagonal-channel structure for bilinear filters. Based on the channel structure, the adaptive diagonal-channel bilinear filtered-X LMS (DBFXLMS) and RLS (DBFXRLS) algorithms will carefully and explicitly be derived. To reduce computations of the developed DBFXRLS algorithm, the sequential channel update strategy according to diagonal-channel signal vectors described in [19-20] are then derived. In addition, comparisons of computational complexities for the developed algorithms are included. Section 3 presents computer simulations to demonstrate the control performance improvement for the situations of using the chaotic noise and primary path having a bilinear model; the chaotic noise and linear primary path; and the saturated reference signal and linear primary path. Finally, conclusions are given in Section 4.

The main contributions of this paper include derivations of the diagonal-channel bilinear filtered-X least mean square (DBFXLMS) and recursive least square (DBFXRLS) algorithms, and the DBFXRLS algorithm with sequential update; computational load analysis for each developed algorithm; performance validation using computer simulations.

2. Development of Filtered-X Adaptive Algorithms for Diagonal-Channel Bilinear Filters

2.1. Diagonal-Channel Structure for Bilinear Filters

The relationship between input $x(n)$ and output $y(n)$ for a bilinear filter with a memory length of N described in [18] is defined below:

$$y(n)=\sum_{j=0}^{N}a_j(n)x(n-j)+\sum_{j=1}^{N}b_j(n)y(n-j)+\sum_{i=0}^{N}\sum_{j=1}^{N}c_{i,j}(n)x(n-i)y(n-j) \tag{1}$$

where $a_j(n)$, $b_j(n)$, $c_{i,j}(n)$ are filter coefficients. Note that $a_j(n)$ and $b_j(n)$ have $N+1$ and N coefficients, while $c_{i,j}(n)$ denotes a set of bilinear coefficients and has $N(N+1)$ elements. In order to derive a diagonal-channel form, (1) can be equivalently written as:

$$y(n)=\sum_{j=0}^{N}a_j(n)x(n-j)+\sum_{j=1}^{N}b_j(n)y(n-j)+\sum_{i=1}^{N}\sum_{j=0}^{N-i}g_{i,j}(n)x(n-j)y(n-j-i)+\sum_{i=1}^{N}\sum_{j=0}^{N-i}h_{i,j}(n)x(n-i-j)y(n-j) \tag{2}$$

where i denotes the bilinear filter channel number and j designates the time index. Expanding (2) leads to the following summation of all the channel outputs:

$$\begin{aligned}y(n)=&\sum_{j=0}^{N}a_j(n)x(n-j)+\sum_{j=1}^{N}b_j(n)y(n-j)\\&+\sum_{j=0}^{N-1}g_{1,j}(n)x(n-j)y(n-1-j)\\&+\sum_{j=0}^{N-2}g_{2,j}(n)x(n-j)y(n-2-j)+\ldots\\&+\sum_{j=0}^{0}g_{N,0}(n)x(n)y(n-N)\\&+\sum_{j=0}^{N-1}h_{1,j}(n)x(n-1-j)y(n-1-j)\\&+\sum_{j=0}^{N-2}h_{2,j}(n)x(n-2-j)y(n-1-j)+\ldots\\&+\sum_{j=0}^{0}h_{N,0}(n)x(n-N)y(n-1)\end{aligned} \tag{3}$$

where $g_{1,j}(n)$ has N coefficients for the channel input signal $x(n)y(n-1)$, $g_{2,j}(n)$ has $N-1$ coefficients for the channel input signal $x(n)y(n-2)$ and so on. Similarly, $h_{1,j}(n)$ has N coefficients for the channel input signal $x(n-1)y(n-1)$, $h_{2,j}(n)$ has $N-1$ coefficients for its channel input signal $x(n-2)y(n-1)$ and so on. There are $2N$ channels for the cross terms. Figure 2 depicts the diagonal-channels and signal elements for the case of $N=3$. It can be seen that the signal element in each channel is a delayed term of the previous one.

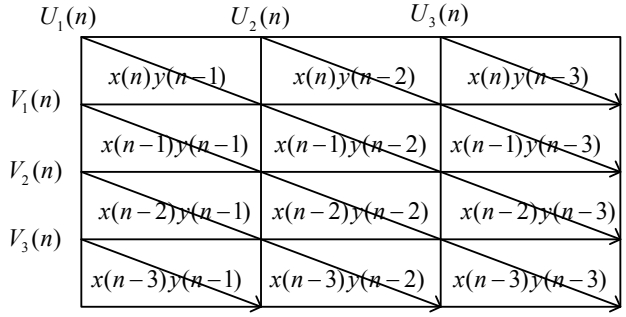

Figure 2. Cross terms of signal elements and diagonal-channel structure for $N=3$

Finally, the developed diagonal-channel bilinear filter can be expressed as

$$y(n)=A^T(n)X(n)+B^T(n)Y(n-1)+\sum_{i=1}^{N}G_i^T(n)U_i(n)+\sum_{i=1}^{N}H_i^T(n)V_i(n) \quad (4)$$

where the signal vectors and their corresponding coefficient vectors are listed below:

$$X^T(n)=[x(n)\ x(n-1)\ \cdots\ x_N(n-N)]^T \quad (5)$$

$$A^T(n)=[a_0(n)\ a_1(n)\ \cdots\ a_N(n)]^T \quad (6)$$

$$Y^T(n-1)=[y(n-1)\ y(n-2)\ \cdots\ y(n-N)]^T \quad (7)$$

$$B^T(n)=[b_1(n)\ b_2(n)\ \cdots\ b_N(n)]^T \quad (8)$$

For $i=1,2,\cdots,N$

$$U_i^T(n)=[x(n)y(n-i)\ x(n-1)y(n-i-1)\ \cdots\ x(n-N+i)y(n-N)]^T \quad (9a)$$

$$G_i^T(n)=[g_{i,0}(n)\ g_{i,1}(n)\ \cdots\ g_{i,N-i}(n)]^T \quad (9b)$$

and

$$V_i^T(n)=[x(n-i)y(n-1)\ x(n-i-1)y(n-i-1)\ \cdots\ x(n-N)y(n-1-N+i)]^T \quad (10)$$

$$H_i^T(n)=[h_{i,0}(n)\ h_{i,1}(n)\ \cdots\ h_{i,N-i}(n)]^T \quad (11)$$

Note that there are ($2N+2$) channels in total. Both $G_i^T(n)$ and $H_i^T(n)$ have dimensions of $N-i+1$. The combined weight vector and reference signal vector are given below:

$$W^T(n)=[A^T(n)\ B^T(n)\ G_1^T(n)\ \cdots\ G_N^T(n)\ H_1^T(n)\ \cdots\ H_N^T(n)] \quad (12)$$

$$L^T(n)=[X^T(n)\ Y^T(n-1)\ U_1^T(n)\ \cdots\ U_N^T(n)\ V_1^T(n)\ \cdots\ V_N^T(n)] \quad (13)$$

With the definitions of (12) and (13), the bilinear filter output can be expressed as a linear filter, that is,

$$y(n)=W^T(n)L(n) \quad (14)$$

Note that $W(n)$ has a dimension determined below:

$$N+1+N+2(1+2+\cdots+N)=N^2+3N+1 \quad (15)$$

2.2. Diagonal-Channel FXLMS Algorithm for Bilinear Filters

Using the same approach for deriving the adaptive Volterra filter [8], the adaptive algorithm can be achieved by minimizing the instantaneous squared error using the steepest descent algorithm:

$$W(n+1)=W(n)-\frac{\mu}{2}\nabla e^2(n) \quad (16)$$

where μ is a convergence factor and the gradient estimator is expressed as

$$\nabla e^2(n)=2e(n)\nabla e(n)=2e(n)\frac{\partial e(n)}{\partial W(n)} \quad (17)$$

Note that the error signal from the error microphone is

$$e(n)=d(n)-y(n)*s(n) \quad (18)$$

where $d(n)$, the output of the primary path $P(z)$, is the primary noise to be attenuated; $s(n)$ is the impulse response of the secondary path $S(z)$ and $*$ denotes linear convolution. Substituting (18) in (17), the error gradient in (17) can further be expressed as

$$\frac{\partial e(n)}{\partial W(n)}=-s(n)*\frac{\partial y(n)}{\partial W(n)} \quad (19)$$

Similar to [14], the following assumptions are made:

For $j=1,2,\cdots,N$

$$\frac{\partial y(n-j)}{\partial a_m(n)}=\frac{\partial y(n-j)}{\partial b_m(n)}=\frac{\partial y(n-j)}{\partial g_{k,m}(n)}=\frac{\partial y(n-j)}{\partial h_{k,m}(n)}\approx 0 \quad (20)$$

Using (20), it leads to the following

$$\frac{\partial u_i(n-j)}{\partial a_m(n)} = x(n-j)\frac{\partial y(n-i-j)}{\partial a_m(n)} \approx 0 \quad (21)$$

$$\frac{\partial u_i(n-j)}{\partial b_m(n)} = x(n-j)\frac{\partial y(n-i-j)}{\partial b_m(n)} \approx 0 \quad (22)$$

$$\frac{\partial u_i(n-j)}{\partial g_{k,m}(n)} = x(n-j)\frac{\partial y(n-i-j)}{\partial g_{k,m}(n)} \approx 0 \quad (23)$$

$$\frac{\partial u_i(n-j)}{\partial h_{k,m}(n)} = x(n-j)\frac{\partial y(n-i-j)}{\partial h_{k,m}(n)} \approx 0 \quad (24)$$

Similarly,

$$\frac{\partial v_i(n-j)}{\partial a_m(n)} = \frac{\partial v_i(n-j)}{\partial b_m(n)} = \frac{\partial v_i(n-j)}{\partial g_{k,m}(n)} = \frac{\partial v_i(n-j)}{\partial h_{k,m}(n)} \approx 0 \quad (25)$$

In the vector form, we obtain

$$\frac{\partial y(n)}{\partial A(n)} = X(n) + B^T(n)\frac{Y(n-1)}{\partial A(n)} + \sum_{i=1}^{N} H_i^T(n)\frac{\partial U_i(n)}{\partial A(n)} + \sum_{i=1}^{N} G_i^T(n)\frac{\partial V_i(n)}{\partial A(n)} \approx X(n) \quad (26)$$

Similarly,

$$\frac{\partial y(n)}{\partial B(n)} = Y(n-1) \quad (27)$$

$$\frac{\partial y(n)}{\partial H(n)} \approx [U_1^T(n) \ \cdots \ U_N^T(n)]^T \quad (28)$$

$$\frac{\partial y(n)}{\partial G(n)} \approx [V_1^T(n) \ \cdots \ V_N^T(n)]^T \quad (29)$$

Substituting (26)-(29) in (16) and replacing $s(n)$ with its estimate $\overline{s}(n)$ concludes the DBFXLMS algorithm:

$$A(n+1) = A(n) + \mu_a e(n) X'(n) \quad (30)$$

$$B(n+1) = B(n) + \mu_b e(n) Y'(n-1) \quad (31)$$

$$G_i(n+1) = G_i(n) + \mu_c e(n) U_i'(n) \quad i = 1,2,\cdots,N \quad (32)$$

$$H_i(n+1) = H_i(n) + \mu_c e(n) V_i'(n) \quad i = 1,2,\cdots,N \quad (33)$$

where μ_a, μ_b, and μ_c are the convergence factors which control the convergence speed and the stability of the algorithm for updating the feed forward coefficients $A(n)$, feedback coefficients $B(n)$, and cross term coefficients $G_i(n)$ and $H_i(n)$. Define the signal vector $L'(n)$ as

$$L'^T(n) = [X'^T(n) \ Y'^T(n-1) \ U_1'^T(n) \ \cdots \ U_N'^T(n) \ V_1'^T(n) \ \cdots \ V_N'^T(n)] \quad (34a)$$

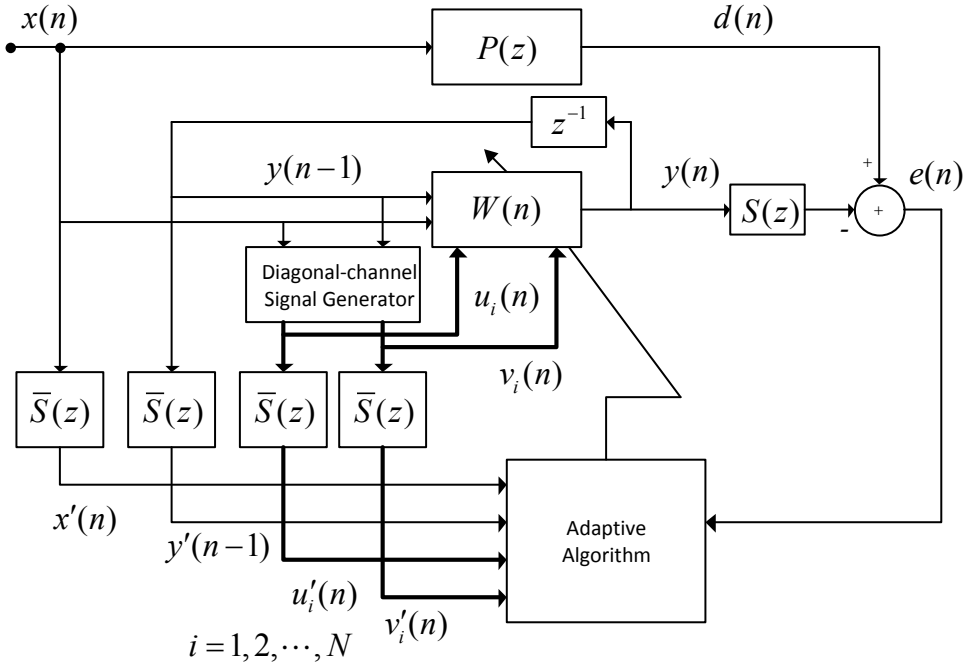

Figure 3. Diagonal-channel bilinear filtered-X LMS (DBFXLMS) algorithm

To guarantee the steady state solution, μ_a, μ_b, and μ_c must satisfy the following condition:

$$0 < \mu_a,\ \mu_b,\ \mu_c < \frac{2}{\lambda_{\max}} \tag{34b}$$

where $\lambda_{\max}$ is the maximum eigenvalue of the auto correlation matrix of the filtered input signals, $E\{L'(n)^T L'(n)\}$. It is also very important to note that the filtered reference signals can be obtained from the following ($2N+2$) channel filters:

$$x'(n) = \overline{s}(n) * x(n) \tag{35}$$

$$y'(n-1) = \overline{s}(n) * y(n-1) \tag{36}$$

$$u_i'(n) = \overline{s}(n) * [x(n)y(n-i)] \quad i = 1,2,\cdots,N \tag{37}$$

$$v_i'(n) = \overline{s}(n) * [x(n-i)y(n-1)] \quad i = 1,2,\cdots,N \tag{38}$$

The proposed DBFLMS algorithm is depicted in Figure 3.

2.3. FXRLS Algorithm for Bilinear Filters

Using (12) and (13), the standard linear FXRLS algorithm [2] can be modified to the DBFXRLS algorithm:

$$e(n) = d(n) - s(n) * y(n) \quad \text{(from the sensor)} \tag{39}$$

$$k(n) = \frac{\lambda^{-1} Q(n-1)L'(n)}{1 + \lambda^{-1} L'^{T}(n) Q(n-1) L'(n)} \tag{40}$$

$$W(n+1) = W(n) + k(n)e(n) \tag{41}$$

$$Q(n) = \lambda^{-1}[Q(n-1) - k(n)L'^{T}(n)Q(n-1)] \tag{42}$$

where λ ($0 << \lambda < 1$) is the forgetting factor and the initial $Q(0)$ is set to be $Q(0) = I/\delta$, where I is the identity matrix while $1/\delta$ is a very small number.

As shown in (34), the dimension of $L'(n)$ is in an order of N^2. Therefore, the computation requirement for DBFXRLS algorithm is an order of N^4. To reduce the computations, the sequential channel update used for VFXRLS and FSRLS in [19, 20] can directly be applied. The resultant DBFXRLS with sequential channel update (DBFXRLS-SEQ) can be expressed as

$$e(n) = d(n) - s(n) * y(n) \quad \text{(from the sensor)} \tag{43}$$

For $i = 1,2,...,2N+2$

$$k_i(n) = \frac{\lambda^{-1} Q_i(n-1)L_i'(n)}{1 + \lambda^{-1} L_i'^{T}(n) Q_i(n-1) L_i'(n)} \tag{44}$$

$$W_i(n+1) = W_i(n) + k_i(n)e(n) \tag{45}$$

$$Q_i(n) = \lambda^{-1}[Q_i(n-1) - k_i(n)L_i'^{T}(n)Q_i(n-1)] \tag{46}$$

Notice that the channels are re-indexed using the following weight vector notations for convenience:

$$W_1(n) = A(n) \tag{47}$$

$$W_2(n) = B(n) \tag{48}$$

$$W_3(n) = G_1(n),\ldots,\ W_{N+2}(n) = G_N(n) \tag{49}$$

$$W_{N+3}(n) = H_1(n),\ldots,\ W_{2N+2}(n) = H_N(n) \tag{50}$$

where $k_i(n)$, $Q_i(n)$, and $L_i'(n)$ are the gain vector, inverse of the correlation matrix, and filtered input vector from channel i, respectively. Again, since the sizes of the gain vector $k_i(n)$ and correlation matrix $Q_i(n)$ for channel i in the DBFXRLS-SEQ algorithm are much smaller than the one in the DBFXRLS algorithm, the computational load is greatly reduced. On the other hand, the DBFXRLS-SEQ algorithm also prevents the use of a larger size correlation matrix, which may unfavorably cause the algorithm to diverge.

Notice that the mean square error function of a bilinear filter is a nonlinear function of its coefficients. It may exhibit local minima such that the algorithm may not converge to the global minimum. The recursive model may be unstable unless the algorithm is carefully designed and filter stability is monitored.

2.4. Computational Complexity

As shown in Figure 3, the diagonal-channel generator produces $2N$ channel signals, hence, requiring $2N$ multiplications. The output for computing (4) requires

$$(2N) + (N+1) + N + 2(1+2+\cdots+N) = N^2 + 5N + 1 \tag{51}$$

multiplications and

$$N + (N-1) + 2(1+2+\cdots+N-1) + 3 = N^2 + N + 2 \tag{52}$$

additions. Generating $(2N+2)$ filtered channel signals requires $2(N+1)M$ multiplications and $2(N+1)(M-1)$ additions, where M is the number of coefficients in the secondary path estimate $\overline{S}(z)$. These multiplications and additions are the same for all the bilinear filter algorithms.

For the DBFXLMS algorithm, the multiplication load for (30)-(33) needs

$$2(N+1) + 2N + 2(1+2+\cdots+N) = N^2 + 5N + 2 \tag{53}$$

while the addition load requires

$$(N+1) + N + 2(1+2+\cdots+N) = N^2 + 3N + 1 \tag{54}$$

For the RLS based algorithms, starting with the standard linear filtered-X RLS algorithm with K filter coefficients, it is known that updating (40) needs $K^2 + 2K + 1$ multiplications and K^2 additions; (41) requires K

multiplications and additions, respectively; and (42) requires $3K^2$ multiplications and $2K^2 - K$ additions. The total numbers of multiplications and additions for the RLS algorithm with an input vector size of K are found as $4K^2 + 3K + 1$ and $3K^2$. For the bilinear filter with an input vector size of $K = N^2 + 3N + 1$ as shown in (15), the total number of multiplications is determined by

$$4(N^2 + 3N + 1)^2 + 3(N^2 + 3N + 1) + 1 \tag{55}$$

and the total number of additions becomes

$$3(N^2 + 3N + 1)^2 \tag{56}$$

The computation is in the order of N^4.

For the sequential channel update algorithm (DBFXRLS-SEQ), the number of multiplications is derived as

$$\begin{aligned} &4(N+1)^2 + 3(N+1) + 1 + 4N^2 + 3N + 1 \\ &+2[4(1^2 + \cdots + N^2) + 3(1 + \cdots + N) + N] \\ &= (8N^3 + 36N^2 + 61N + 36)/3 \end{aligned} \tag{57}$$

Similarly, the number of additions is derived as

$$\begin{aligned} &3(N+1)^2 + 3N^2 + 2 \times 3(1^2 + \cdots + N^2) \\ &= 2N^3 + 9N^2 + 7N + 3 \end{aligned} \tag{58}$$

Table 1. Multiplications for algorithms based on the bilinear and second-order Volterra models

Algorithm type	Multiplications	$N = 9$ $M = 4$
Second-order VFXLMS	$3(N+1)(N+4)/2 + (N+1) + (N+2)M$	249
DBFXLMS	$(N^2 + 5N + 2) + (N^2 + 5N + 1) + 2(N+1)M$	335
Second-order VFXRLS	$4[(N+1)(N+4)/2]^2 + 3(N+1)(N+4)/2 + 1$ $(N+1)(N+4)/2 + (N+1) + (N+2)M$	17215
DBFXRLS	$4(N^2 + 3N + 1)^2 + 3(N^2 + 3N + 1) + 1$ $+(N^2 + 5N + 1) + 2(N+1)M$	48059
DBFXRLS-SEQ	$(8N^3 + 36N^2 + 61N + 36)/3$ $+(N^2 + 5N + 1) + 2(N+1)M$	3318

Table 2. Additions for algorithms based on the bilinear and second-order Volterra models

Algorithm type	Additions	$N = 9$ $M = 4$
Second-order VFXLMS	$(N+1)(N+4) - 1 + (N+2)(M-1)$	162
DBFXLMS	$(N^2 + 3N + 1) + (N^2 + N + 2) + 2(N+1)(M-1)$	261
Second-order VFXRLS	$3[(N+1)(N+4)/2]^2 + (N+1)(N+4)/2$ $-1 + (N+2)(M-1)$	12772
DBFXRLS	$3(N^2 + 3N + 1)^2 + (N^2 + N + 2) + 2(N+1)(M-1)$	35795
DBFXRLSSEQ	$2N^3 + 9N^2 + 7N + 3 + (N^2 + N + 2)$ $+2(N+1)(M-1)$	2405

The computational complexities for the DBFXLMS, DBFXRLS, and DBFXRLS-SEQ algorithms along with the second-order Volterra filters are summarized in Table 1 and Table 2.

From Tables 1 and 2, it is clear that the computational complexity for DBFXRLS-SEQ algorithm has an order of N^3 and hence is significantly reduced in comparison to the DBFXLS algorithm with its computational complexity in an order of N^4. Among the proposed algorithms, the DBFXLMS algorithm has lowest computational load with an order of N^2.

3. Computer Simulations

3.1. Linear Primary Path with Logistic Noise

To demonstrate the performance of the propose algorithms, a linear primary path is assumed to be

$$P(z) = z^{-8} - 0.3z^{-9} + 0.2z^{-10} \tag{59}$$

The secondary path transfer function $S(z)$ and its estimate $\overline{S}(z)$ with non-minimum phase [10] are the same and given below:

$$S(z) = \overline{S}(z) = z^{-2} + 1.5z^{-3} - z^{-4} \tag{60}$$

A logistic chaotic noise [10] is chosen as the reference signal source for all the simulations and is generated using the following equation:

$$x(n+1) = \lambda x(n)[1 - x(n)] \tag{61}$$

where $\lambda = 4$ and $x(0) = 0.9$ are selected. This nonlinear noise process is then normalized to have a unit signal power of $\sigma_x^2 = 1.0$. The control performance is plotted using the normalized mean square error (NMSE) versus the number of iterations, which is defined below:

$$NMSE = 10\log_{10}\left(\frac{E\left(e^2(n)\right)}{\sigma_d^2}\right) \tag{62}$$

where σ_d^2 is the power of the primary noise at the canceling point. In addition, the signal power to noise power ratio (SNR) of 40 dB is used for all the simulations. A memory size of $N = 9$ is selected for both the second-order Volterra filter and bilinear filter. For the FXLMS based algorithms, $\mu_L = 0.001$ (linear signal vector) and $\mu_Q = 0.0005$ (quadratic signal vectors) are selected for the second-order Volterra filter; for the diagonal-channel bilinear filter, $\mu_a = 0.0005$, $\mu_b = 0.0002$, $\mu_c = 0.0002$ are chosen. Notice that all the step sizes are selected to be less than the corresponding upper bound of $2/tr(R_U)$ to ensure the algorithm convergence, where $tr(R_U)$ is the trace of the cross correlation matrix of the filtered input vector. For the FXRLS based algorithms, $\lambda = 0.9995$ is selected for the second-order Volterra filter. $\lambda = 0.9995$ and $\lambda = 0.997$ are selected, respectively, for the DBFXRLS and DBFXRLS-SEQ algorithms. Initial filter coefficients for all the algorithms are set to zero. Figure 4a displays the NMSE's versus the number of iterations obtained by the VFXLMS and DBFXLMS algorithms. After 500,000 iterations, the second-order VFXLMS algorithm achieves a NMSE value of -30 dB and the DBFXLMS algorithm obtains a NMSE value of -39 dB. The DBFXLMS achieves significantly better performance. Figure 4b shows the NMSE's versus the number of iterations achieved by the FXRLS based algorithms. As shown in Figure 4b, after 50,000 iterations, the second-order VFXRLS algorithm achieves an approximate NMSE value of -30 dB, while both DBFXRLS and DBFXRLS-SEQ algorithms obtain approximate NMSE value of -40 dB, respectively. It can be seen that the DBFXRLS algorithm has the best performance with a significantly large amount of computations (order of N^4) for each iteration; the DBFXRLS-SEQ offers a compromised solution with the reduced complexity (order of N^3 for each iteration), and its control performance is better than the one from the DBFXLMS algorithm shown in Figure 4a. Note that the DBFXLMS algorithm has much lower computational complexity with an order of N^2 for each iteration. Furthermore, the DBFXRLS algorithm shows the fastest convergence rate. However, among these three algorithms, the DBFXRLS-SEQ algorithm gains the benefit from its lower computational load.

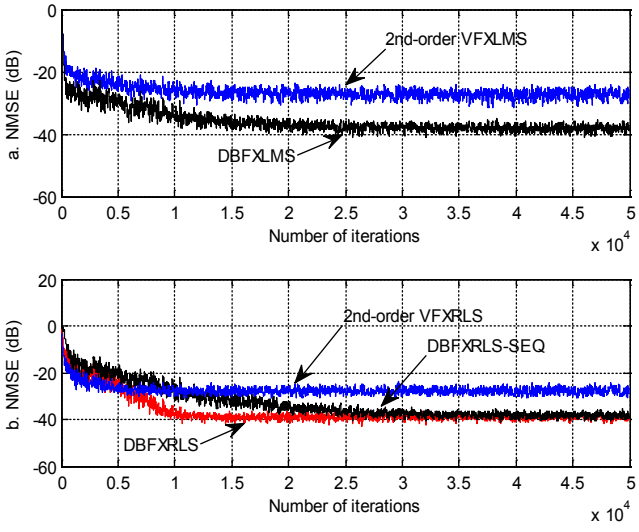

Figure 4. Performance comparisons: (a) FXLMS based algorithms (2nd-order VFXLMS: $\mu_L = 0.001$, $\mu_Q = 0.0005$; DBFXLMS: $\mu_a = 0.0005$, $\mu_b = 0.0002$, $\mu_c = 0.0002$; (b) FXRLS based algorithms (2nd-order VFXRLS: $\lambda = 0.9995$, $\delta = 1/1000$; DBFXRLS $\lambda = 0.9995$, $\delta = 1/1000$; DBFXRLS-SEQ: $\lambda = 0.997, \delta = 1/1000$)

3.2. Linear Primary Path with Saturated Narrowband Noise

In the simulations, the measured primary path and secondary path provided in [2] are used. Their frequency and phase responses are displayed in Figure 5.

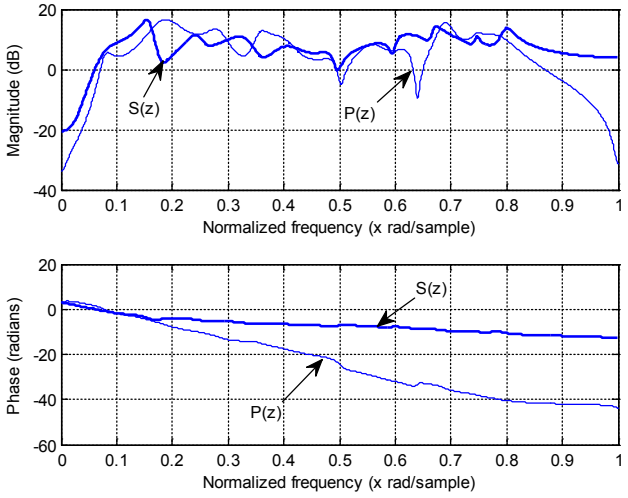

Figure 5. Frequency responses for the primary path and secondary path used in simulations

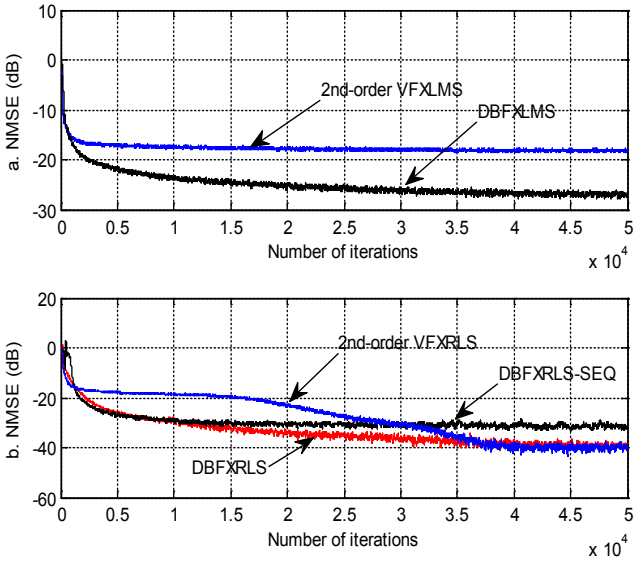

Figure 6. Performance comparisons: (a) FXLMS based algorithms (2nd-order VFXLMS: $\mu_L = 0.001$, $\mu_Q = 0.0005$; DBFXLMS: $\mu_a = 0.0005$, $\mu_b = 0.00002$, $\mu_c = 0.00002$); (b) FXRLS based algorithms (2nd-order VFXRLS: $\lambda = 0.9996$, $\delta = 1/1000$; DBFXRLS $\lambda = 0.9998$, $\delta = 1/1000$; DBFXRLS-SEQ: $\lambda = 0.999, \delta = 1/10000$)

Figure 6 shows the control performance for narrow band noise as used in [14]. The reference signal consists of the normalized frequencies of 0.01, 0.02, and 0.08 and is normalized to have a unit power. The SNR at the noise cancelling point is set to 40 dB. Furthermore, the reference signal is assumed to have strong saturation. Hence, for all the simulations, the reference signal is clipped with a threshold at 50% of the maximum signal value before it is used for driving the adaptive filters and for inputting the filter of the secondary path estimate. As shown in Figure 6a, the DBFXLMS algorithm converges significantly faster the second-order VFXLMS algorithm. This conclusion is reported in [14]. In Figure 6b, both the second-order VFXRLS and DBFXRLS algorithms reach the same control performance (-40 dB) after algorithm convergence. However, the DBFXRLS algorithm has the fastest convergence rate. It is observed that the DBRLS-SEQ algorithm has some performance degradation (-33 dB) but it still offers better control performance than that of the DBFXLMS algorithm (-27 dB) at the moderate computational cost.

4. Conclusions

This paper presented a novel adaptive bilinear filter based on a diagonal-channel structure for nonlinear active noise control. Using the proposed diagonal-channel structure, the diagonal-channel bilinear filtered-X least mean square (DBFXLMS) and recursive lest square (DBFXRLS) algorithms are derived. In order to reduce the computational cost for the DBFXRLS algorithm, the DBFXRLS algorithm using a sequential channel update (DBFXRLS-SEQ) is developed. Computer simulations verify the effectiveness of the proposed algorithms.

REFERENCES

[1] S. J. Elliott, Signal Processing for Active Control. London, UK: Academics, 2001.

[2] S. M. Kuo and D. R. Morgan, Active Noise Control Systems: Algorithms and DSP Implementations. New York: Wiley, 1996.

[3] S. J. Elliott and P. A. Nelson, "Active noise control," IEEE Signal Process. Mag., vol. 10, no. 4, pp. 12-35, Oct. 1993.

[4] S. M. Kuo and D. R. Morgan, "Active noise control: a tutorial review," Proc. IEEE, vol. 87, no. 6, pp. 943-973, Jun. 1999.

[5] L. J. Eriksson, M. C. Allie, and R. A. Greiner, "The selection and application of an IIR adaptive filter for use in active sound attenuation," IEEE Trans. Acoust., Speech, Signal Processing, vol. 35, No. 4, pp. 433-437, March 1987.

[6] S. D. Snyder and N. Tanaka, "Active control of vibration using a neural network," IEEE Trans. Neural Network, vol. 6, No. 4, pp. 819-829, July 1995.

[7] M. Bouchard, B. Pailard, and C. T. L. Dinh, "Improved training of neural networks for nonlinear active control of sound and vibration," IEEE Trans. Neural Networks, vol. 10, No. 2, pp. 391-401, March 1999.

[8] P. Strauch, and B. Mulgrew, "Active control of nonlinear noise processes in a linear duct," IEEE Trans. Signal

Processing, vol. 46, pp. 2404-2412, Sept. 1998.

[9] L. Tan and J. Jiang, "Filtered-X second-order Volterra adaptive algorithms," Eletron. Lett., vol. 33, No. 8, pp. 671-672, Apr. 1997.

[10] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," IEEE Trans. Signal Processing, vol 49, No.8, pp. 1667-1676, August 2001.

[11] D. P. Das and G. Panda, "Active mitigation of nonlinear noise process using a novel filtered-s LMS algorithm," IEEE Trans. Speech, Audio Processing, vol. 12, no. 3, pp. 313-322, May 2004.

[12] G. L. Sicuranza and A. Carini, "Nonlinear multichannel active control using partial updates," Proc. ICASSP-2005, pp. 109-112, 2005.

[13] D. Zhou and V. DeBrunner, "Efficient adaptive nonlinear filters for nonlinear active noise control," IEEE Trans. Circuits Syst. I, vol. 54, no. 3, pp. 669-681, Mar. 2007.

[14] S. M. Kuo and H. T. Wu, "Nonlinear adaptive bilinear filters for active noise control systems," IEEE Trans. Circuits Syst. I, vol. 52, no. 3, pp. 617-624, March 2005.

[15] L. Tan, J. Jiang, "Nonlinear active noise control using diagonal-channel LMS and RLS bilinear filters," IEEE 57th International Midwest Symposium on Circuits & Systems, pp.789-792, College Station, Texas, August 2014.

[16] X.-G. Yan, C. Edwards, "Fault estimation for single output nonlinear systems using an adaptive sliding mode estimator," IET Control Theory & Applications, Vol. 2, No. 10, pp. 841-850, 2008.

[17] X.-G. Yan, S. K. Spurgeon, C. Edwards, "State and parameter estimation for nonlinear delay systems using sliding mode techniques," IEEE Transactions on Automatic Control, Vol. 58, No. 4, 1023-1029, April, 2013.

[18] V. J. Mathews, "Adaptive polynomial filters," IEEE Signal Processing Mag., vol. 8, No. 3, pp. 10-26, July 1991.

[19] L. Tan and J. Jiang, "Adaptive Second-Order Volterra Filtered-X RLS Algorithms with Sequential and Partial Updates for Nonlinear Active Noise Control," IEEE 4th Int. Conf. on Industrial Electronics and Applications, pp.1625-1630, Xi'an, China, May 2009.

[20] L. Tan and J. Jiang, "Active Noise Control Using the Fil1tered-X RLS Algorithm with Sequential Updates," Journal of Communication and Computer, vol. 6, no. 5, pp. 9-14, May 2009.