BLOCK IMPLEMENTATION OF A SYNCHRONIZED LEARNING ALGORITHM IN ADAPTIVE LATTICE FILTERS

Naoki Tokui

Ishikawa National College of Technology
Dept. of Electrical Engineering
Kitachujo, Tsubata, Ishikawa, 929-0392, JAPAN
e-mail:tokui@ishikawa-nct.ac.jp

Kenji Nakayama, Akihiro Hirano

Faculty of Engineering, Kanazawa University
Dept. of Information and Systems Engineering
2-40-20, Kodatsuno, Kanazawa, 920-8667, JAPAN
e-mail:{nakayama,hirano}@t.kanazawa-u.ac.jp

ABSTRACT

In order to achieve fast convergence and less computation for adaptive filters, a joint method combining a whitening process and the NLMS algorithm is a hopeful approach. However, updating the filter coefficients is not synchronized with the reflection coefficient updating resulting in unstable behavior. We analyzed effects of this, and proposed the “Synchronized Learning Algorithm” to solve this problem. Asynchronous error between them is removed, and fast convergence and small residual error were obtained. This algorithm, however, requires $O(ML)$ computations, where M is an adaptive filter length, and L is a lattice predictor length. It is still large compared with the NLMS algorithm. In order to achieve less computation while the fast convergence is maintained, a block implementation method is proposed. The reflection coefficients are updated at some period, and are fixed during this interval. The proposed block implementation can be effectively applied to parallel form adaptive filters, such as sub-band adaptive filters. Simulation using speech signal shows that a learning curve of the proposed block implementation a little slower than the our original algorithm, but can save the computational complexity.

1. INTRODUCTION

As VLSI technology has been developed, adaptive filters have been applied to audio acoustic processing, control systems, telecommunication systems, and others. Among them, acoustic echo cancellation and noise cancelation are very important.

When very high-order adaptive filters are required, fast convergence and less computation for real signals are very important. The normalized LMS (NLMS) algorithm can be implemented with less computation. However, a very long time is required for convergence. On the contrary, the recursive least squires (RLS) algorithm can converge fast, at the expense of computational complexity.

One method to overcome this problem is to join a whitening process and the NLMS algorithm. The whitening process includes orthogonal transform and linear prediction [1]–[6]. The former method requires many frequency bands in order to realize good orthogonalization [1]–[3]. A lattice predictor is used in the latter method [1],[4],[5]. Order of the predictor is determined by that of an equivalent AR model generating the input signal, which is not so high compared with filter orders. However, in the original adaptive lattice filters, updating the filter coefficients are not synchronized with the reflection coefficient updating, resulting in large residual errors.

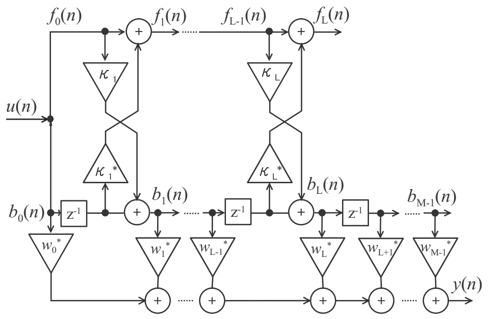

Fig. 1. A joint lattice and transversal filter (lattice predictor of order L and adaptive filter of order M)

This problem was analyzed, and the ”Synchronized Learning Algorithm” was proposed. The filter coefficients are compensated for taking the reflection coefficient updating into account [7],[8]. This method, however, requires $O(ML)$ computations, where M is an adaptive filter length, and L is a lattice predictor length. Although $L \ll M$ is usually satisfied, $O(ML)$ is still unpractical requirement.

In this paper, a block implementation method is proposed in order to save computations. The reflection coefficients are updated at some period, and they are fixed during this interval. Computer simulation using real voice signals will be demonstrated to confirm usefulness of the proposed method.

2. JOINT LATTICE AND TRANSVERSAL ADAPTIVE FILTER

2.1. Update of Reflection Coefficients

Figure 1 shows a block diagram of a joint lattice and transversal filter. The 1st-stage is the lattice predictor and the 2nd-stage is the transversal adaptive filter.

$f_m(n)$ and $b_m(n)$ are the forward and the backward prediction errors, respectively, at the m-th stage and the n-th sample. They are calculated by the following recursive formulas.

$$f_m(n) = f_{m-1}(n) + \kappa_m^*(n) b_{m-1}(n-1) \quad (1)$$

$$b_m(n) = b_{m-1}(n-1) + \kappa_m(n) f_{m-1}(n) \quad (2)$$

$$f_0(n) = b_0(n) = u(n) \quad (3)$$

$$m = 1, \ldots, L$$

$\kappa_m(n)$ is the reflection coefficient at the m-th stage and the n-th sample. $*$ indicate complex conjugate. The reflection coefficient $\kappa_m(n)$ is determined so as to minimize the following prediction errors.

$$\kappa_m(n) = -\frac{2E[b_{m-1}(n-1)f^*_{m-1}(n)]}{E[|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2]} \tag{4}$$

Furthermore, letting the numerator and the denominator be $-2\kappa_{N,m}(n)$ and $\kappa_{D,m}(n)$, respectively, they are approximately updated by

$$\begin{aligned}\kappa_{N,m}(n) &= (1-\gamma)\,(\gamma\kappa_{N,m}(n-1) \\ &\qquad + b_{m-1}(n-1)f^*_{m-1}(n)) \qquad (5)\\ \kappa_{D,m}(n) &= (1-\gamma)\,(\gamma\kappa_{D,m}(n-1) \\ &\qquad + \left(|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2\right)) \qquad (6)\\ 1 > \gamma &> 0\end{aligned}$$

2.2. Update of Filter Coefficients

The input signal for the transversal filter is the backward prediction error $b_m(n)$. Letting $\boldsymbol{b}(n)$, $\boldsymbol{w}(n)$ and $y(n)$ be the backward prediction error, the filter coefficients and the output, respectively, they are related by

$$\boldsymbol{b}(n) = [b_0(n), \ldots, b_{M-1}(n)]^T \tag{7}$$

$$\boldsymbol{w}(n) = [w_0(n), \ldots, w_{M-1}(n)]^T \tag{8}$$

$$y(n) = \boldsymbol{w}^H(n)\boldsymbol{b}(n). \tag{9}$$

T and H indicate Transposition and Hermitian transposition, respectively.

The filter coefficients are updated by the NLMS algorithm as shown in

$$e(n) = d(n) - y(n) \tag{10}$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{\alpha}{\|\boldsymbol{b}(n)\|^2 + \delta}\boldsymbol{b}(n)e(n). \tag{11}$$

α is a step size and δ is a small positive number. The other algorithms can be also employed.

2.3. Relation between Reflection and Filter Coefficient Update

$\kappa_m(n)$ is updated at the n-th sample, and will be used at the $(n+1)$-th sample. $\boldsymbol{b}(n)$ is obtained by using $\kappa_m(n-1)$, that is the previous values. The filter coefficients are updated at the n-th sample using $\boldsymbol{b}(n)$ resulting $\boldsymbol{w}(n+1)$, which will be used at the $(n+1)$-th sample.

In Eq.(11), $e(n)$ and $\boldsymbol{b}(n)$ are obtained using $\kappa_m(n-1)$, not $\kappa_m(n)$. This means the filter coefficients $\boldsymbol{w}(n+1)$ can reduce the cost function in collaboration with $\kappa_m(n-1)$, not with $\kappa_m(n)$. However, at the $(n+1)$-th sample, $\boldsymbol{w}(n+1)$ is combined with $\kappa_m(n)$ to generate the output $y(n+1)$. This means the filter coefficient update is always one sample behind the reflection coefficient update.

3. A SYNCHRONIZED LEARNING ALGORITHM

3.1. Transfer Function Representation

The transfer function of the joint lattice and transversal filter shown in Fig.1 consists of the reflection coefficients and the filter coefficients. In this section, an equivalent transfer function in the time domain is obtained. First, $\boldsymbol{b}(n)$ is expressed by

$$\boldsymbol{b}(n) = \boldsymbol{K}^H(n)\boldsymbol{u}(n) \tag{12}$$

$$\boldsymbol{u}(n) = [u(n), \ldots, u(n-M+1)]^T. \tag{13}$$

Second, $\boldsymbol{f}(n)$ is expressed by

$$\boldsymbol{f}(n) = \boldsymbol{J}^H(n)\boldsymbol{u}(n) \tag{14}$$

Then elements of matrices $\boldsymbol{J}(n)$, $\boldsymbol{K}(n)$ can be calculated easily using the following equations.

$$J_{l,m}(n) = J_{l,m-1}(n) + \kappa^*_m(n)K_{l-1,m-1}(n-1) \tag{15}$$

$$K_{l,m}(n) = \kappa_m(n)J_{l,m-1}(n) + K_{l-1,m-1}(n-1) \tag{16}$$

Here, $\boldsymbol{K}(n)$ has the following structure.

$$\boldsymbol{K}(n) = \\ \begin{bmatrix} 1 & K_{0,1}(n) & \cdots & K_{0,L}(n) & 0 & \cdots \\ 0 & 1 & \ddots & \vdots & K_{0,L}(n-1) & \ddots \\ \vdots & 0 & \ddots & K_{L-1,L}(n) & \ddots & \ddots \\ \vdots & \ddots & \ddots & 1 & K_{L-1,L}(n-1) & \ddots \\ \vdots & \ddots & \ddots & \ddots & 1 & \ddots \\ 0 & \cdots & \cdots & 0 & \cdots & \ddots \end{bmatrix} \tag{17}$$

Using the above expression, the filter output is given by

$$y(n) = \boldsymbol{w}^H(n)\boldsymbol{K}^H(n)\boldsymbol{u}(n). \tag{18}$$

In this expression, $\boldsymbol{w}^H(n)\boldsymbol{K}^H(n)$ represents the equivalent transfer function in the time domain.

3.2. Compensation of Filter Coefficients

From the discussions in [7],[8], $\boldsymbol{w}(n+1)$ is updated using $\boldsymbol{K}(n)$, therefore, the following output at the next sample can reduce the cost function.

$$\hat{y}(n+1) = \boldsymbol{w}^H(n+1)\boldsymbol{K}^H(n)\boldsymbol{u}(n+1). \tag{19}$$

However, $\boldsymbol{K}(n)$ is updated at the $(n+1)$-th sample, then the actual output becomes

$$y(n+1) = \boldsymbol{w}^H(n+1)\boldsymbol{K}^H(n+1)\boldsymbol{u}(n+1). \tag{20}$$

$y(n+1)$ cannot reduce the error well. In order to overcome this mismatch, the filter coefficients are compensated so that the transfer function equivalent Eq.(19). That is,

$$\boldsymbol{K}(n+1)\hat{\boldsymbol{w}}(n+1) = \boldsymbol{K}(n)\boldsymbol{w}(n+1). \tag{21}$$

From this condition, we obtain

$$\hat{\boldsymbol{w}}(n+1) = \boldsymbol{K}^{-1}(n+1)\boldsymbol{K}(n)\boldsymbol{w}(n+1). \tag{22}$$

This compensated filter coefficients will be used at the $(n+1)$th-sample to generate $\boldsymbol{b}(n+1)$ and $y(n+1)$. Figure 2 shows the fact that the equivalent transfer function fluctuated by update of the reflection coefficients can be restored by the compensated filter coefficients.

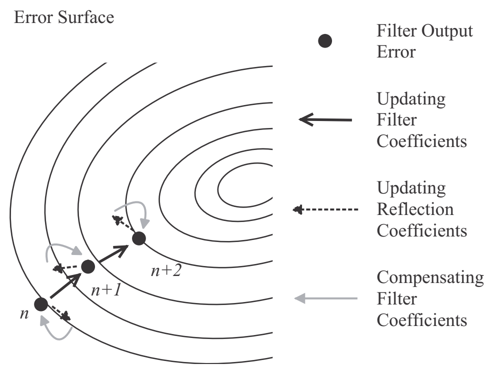
Error Surface
Filter Output Error
Updating Filter Coefficients
Updating Reflection Coefficients
Compensating Filter Coefficients
n
n+1
n+2

Fig. 2. A synchronized learning algorithm in error surface.

Table 1. Comparison of The Joint Lattice and Transversal Filter

	Multiplier	Adder
Synchronized	$2ML + 5M + 9L$	$2ML + 4M + 5L$
Con. Lattice	$5M + 9L$	$4M + 5L$
NLMS	$5M$	$4M$
RLS	$3M^2 + 4M$	$2M^2 + 3M$

3.3. Computational Complexity

Table 1 lists the number of computations of the “Synchronized Learning Algorithm” and the conventional algorithms. “Conventional Lattice” indicates the joint adaptive filter, which has the same structure as the proposed method.

4. BLOCK IMPLEMENTATION OF SYNCHRONIZED LEARNING ALGORITHM

4.1. Reflection Coefficient Update

In the synchronized learning algorithm, the filter coefficients are modified in synchronizing the reflection coefficient update. Usually, the reflection coefficients are updated at every sample. However, if the input signals are stationary or can be handled as stationary signals during some interval, the reflection coefficients will be slightly changed. Furthermore, slight deviation from the ideal reflection coefficients does not affect convergence performance. This means the reflection coefficients can be fixed during some interval. By fixing them, the modification of the filter coefficients given by Eq.(22), which requires a main part of computations, is not required.

Figure 3 shows a time chart of changing the reflection coefficients at every S samples. After $\boldsymbol{\kappa}(n)$ are updated to $\boldsymbol{\kappa}(n+1)$, the matrix $\boldsymbol{K}(n)$ is updated to $\boldsymbol{K}(n+1)$. Using $\boldsymbol{K}(n)$ and $\boldsymbol{K}(n+1)$, the filter coefficients $\boldsymbol{w}(n+1)$ are modified by Eq.(22). This modification is repeated during M samples. Because the signals passing through $\boldsymbol{\kappa}(n+1)$ are transferred through $M-1$ delay elements, and effects of $\boldsymbol{\kappa}(n+1)$ on the filter coefficients continue during M samples. In Fig.3, the hatched blocks occupying M samples indicates this processing. After M samples, the modification by Eq.(22) is stopped, no computations for this purpose is required. Figure 4 shows the part of the matrix $\boldsymbol{K}$, where effects of changing the reflection coefficients apper, and related to the modified filter coefficients $\hat{\boldsymbol{w}}(n)$. This partial modification can save the computations into a half of the original at the most.

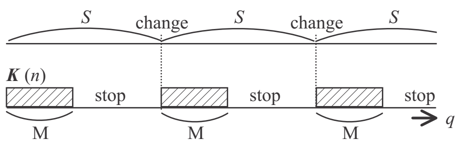

Fig. 3. Block update of reflection coefficients.

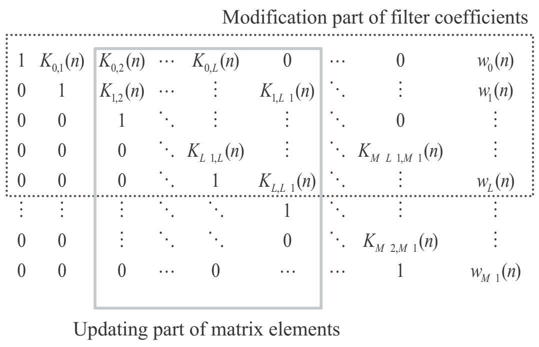
Modification part of filter coefficients
Updating part of matrix elements

Fig. 4. Reflection coefficients update and modification of filter coefficients.

4.2. Double Lattice Predictor Structure

Even though the reflection coefficients can be fixed in some interval for whitening the input signal of the adaptive filters, they must be updated at every sample in order to accurately estimated. Therefore, a double lattice predictor approach is proposed. One of them is used to estimate the reflection coefficients $\boldsymbol{\kappa}(n)$, in which the reflection coefficients are always updated. The other is used in the joint adaptive filter to output the backward prediction errors, in which the reflection coefficients are transferred from the previous lattice predictor at every S samples, and are fixed.

The double lattice predictor structure is shown in Fig. 5. In the upper predictor, κ_i are updated at every sample. They are transferred to the lower predictor, combined with the transversal filter, at every S samples. This process is denoted ”copy” in this figure. $\bar{\kappa_m}(n)$ and $\bar{\kappa}^*_m(n)$ mean the fixed reflection coefficients. $\widehat{f_m}(n)$ and $\widehat{b_m}(n)$ are the forward and backward prediction errors using the fixed reflection coefficients. $\widehat{\boldsymbol{K}}(n)$ is also calculated using the fixed reflection coefficients. Another operations in the joint lattice and transversal filter are the same as the structure shown in Fig.1.

4.3. Computational Complexity

Table 2 lists the number of computations of the “Block Implementation of Synchronized Learning Algorithm”. The computational complexity is different from the sampling points q in Fig.3, where the filter coefficients are modified $(mS \le q \le mS + M)$ or are not modified $(mS + M + 1 \le q \le (m+1)S - 1)$by changing the reflection coefficients. “Maximum” is given for the modified interval and “Minimum” for the other interval. The maximum number of computations in Table 2 is about a half of that of “Synchronized” in Table 1. In the case of sub-band adaptive filters, the proposed method is very useful. If a single DSP is shared by all the sub-bands, then computational requirement can be reduced, that is about $M_{\text{sub}}L$. M_{sub} is a sub-band adaptive filter length, which can be well reduced from M.

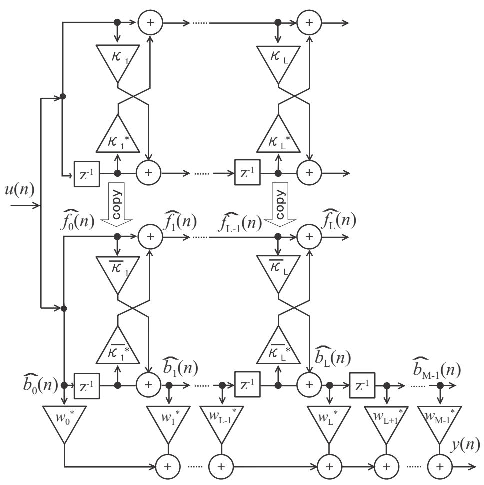

Fig. 5. Double lattice predictor structure

Table 2. Comparison of The Block Implementation of The Joint Lattice and Transversal Filter

	Maximum	Minimum
Multiplier	$ML + L^2/2 + 3M + 11L + 2$	$3M + 11L + 2$
Adder	$2ML - L^2 + 3M + 7L$	$3M + 7L$

5. SIMULATION AND DISCUSSIONS

Simulation was carried out based on system identification. An unknown system is the 10th-order IIR lowpass filter. The impulse response spreads over 50 samples. Therefore, the adaptive filter length M is set to 50 taps.

The voice signal used in the simulation is shown in Fig.6. A sampling frequency is 8kHz, then $20,000$ samples mean 2.5 seconds. Figure 7 shows the learning curves. The proposed method, "Block Implementation (L=20, S=200)", can catch up with the "Synchronized" at 3000 iterations. This means the convergence in early stage is a little slower than the original structure shown in Fig.1, in which the reflection coefficients are updated at every sample. However, the "Block Implementation" can save computational complexity from about $2ML$ to ML compare to the "Synchronized".

From these simulation results, the proposed method is useful for nonstationary processes, such as speech signal.

6. CONCLUSIONS

A block implementation method has been proposed for the joint lattice and transversal filter supervised by the synchronized algorithm. The reflection coefficients are fixed in some interval, where the modification of the filter coefficients can be saved. Computational load of the proposed method is about a half of the original one at the most. The block implementation method can be effectively applied to parallel form adaptive filters, such as sub-band adaptive filters. The computer simulation has shown the proposed method is useful for nonstationary signals such as speech signal.

7. REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, PRENTICE-HALL, 4th Edition, 2002.

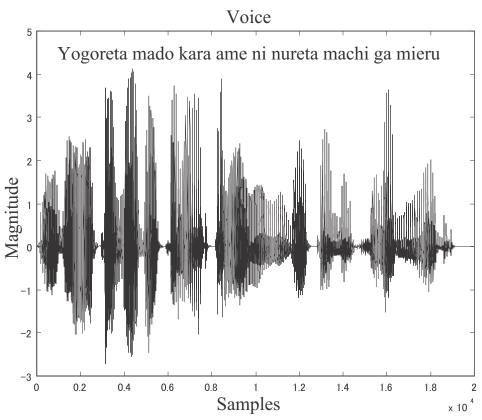

Fig. 6. Input signal of voice.

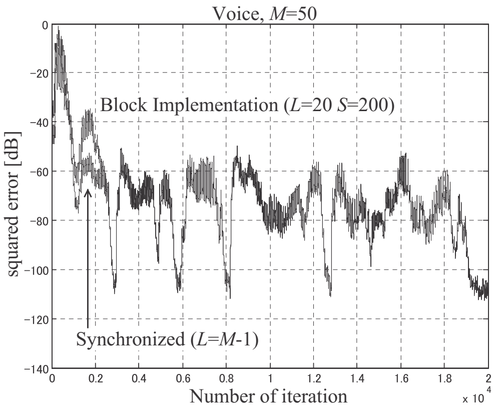

Fig. 7. Learning curves for voice.

[2] J.J. Shynk, "Frequency–domain and multirate adaptive filtering," *IEEE SP MAGAZINE*, pp.14–37, Jan. 1992.

[3] F. Beaufays, "Transform-domain adaptive filters: An analytical approach," *IEEE Trans. Signal Process.*, vol. 43, no. 2, Feb. 1995.

[4] J.H. Yoo, S.H. Cho and D.H. Youn, "A lattice/transversal joint(ltj) structure for an acoustic echo canceller," *1995 ISCAS*, vol. 2., pp.1090–1093, 1995.

[5] S.H. Leung and C.C. Chu, "Adaptive lms filter with lattice prefilter," *Electron. Lett.*, vol. 33, Iss. 1, pp.34–35, 2nd Jan. 1997.

[6] G. Mandyam and N. Ahmed, "The discrete laguerre transform: Derivation and applications," *IEEE Trans. Signal Process.*, vol. 44, no. 12, Dec. 1996.

[7] N. Tokui, K. Nakayama and A. Hirano, "A synchronized learning algorithm for reflection coefficients and tap weights in a joint lattice predictor and transversal filter, " *2001 IEEE ICASSP*, vol. 6, pp.3741–3744 May, 2001.

[8] N. Tokui, K. Nakayama and A. Hirano, "Convergence analysis and a synchronized learning algorithm for a joint lattice predictor and fir adaptive filter, " *IEICE Trans.*, vol. J 85-A, no. 11, pp.1157–1167 Nov., 2002.