SELECTIVE BLOCK UPDATE OF NLMS TYPE ALGORITHMS

Thomas Schertler

Fachgebiet Theorie der Signale
Darmstadt University of Technology
schert@nesi.tu-darmstadt.de

ABSTRACT

Adaptive filters for the cancellation of acoustic echoes, as applied in hands-free telephone sets, require about a thousand coefficients and more to get a significant echo reduction. This leads to a very high computational effort and cannot be realized on most low-cost DSPs.

One common proposition to decrease the computational load is to update only a portion of the coefficients at a time. This decreases not only the computational load but also the convergence speed. To reduce this drawback, it has been suggested that only the most significant coefficients be updated. This improves the convergence speed considerably. Unfortunately, it requires additional memory of twice the filter length.

In our proposal, we present a modified version of the mentioned algorithm which has almost the same adaptation speed but consumes significantly less memory.

1. INTRODUCTION

Adaptive filters are very often used to cancel acoustical echos. Due to the reverberation time of rooms (several hundred milliseconds), a huge number of filter taps is required to achieve sufficient echo reduction. Most echo cancellers make use of the well-known NLMS algorithm which has a rather low complexity [3]. Nevertheless, especially in real-time implementations for consumer products, this complexity may still be too high. Therefore, several algorithms with reduced complexity, based on partial update of the coefficients, have been introduced.

The coefficients to be updated can be selected in several different ways: the periodic NLMS algorithm [6] and other partial update algorithms like the sequential NLMS [2] or the sequential block NLMS for example are using a static scheme to update some pre-defined coefficients of the filter vector. The reduction in complexity also accounts for a reduction of convergence speed. In fact, by reducing the number of coefficients updated in every sample period by a certain factor, the adaptation performance in terms of convergence speed decreases by approximately the same factor in comparison to the conventional NLMS algorithm. This can be an important drawback, especially for acoustic echo cancellers, if insufficient convergence is noticeable.

Another algorithm tries to preserve the performance of the regular NLMS algorithm by updating those coefficients with the largest update [1]. The results obtained with this algorithm are very close to the full update, but this dynamic update scheme has a high demand for memory, namely twice the filter length. Since low memory requirements can be very important for commercial products, this can be a crucial disadvantage for use in industrial hands-free telephone sets.

In this contribution we present an algorithm based on some ideas of [1] which is more convenient for real-time implementations as used in commercial products. Instead of selecting single filter coefficients for the update, it looks for entire blocks of filter coefficients. By using this approach, we lose some convergence speed but we reduce the costs in terms of memory. The computational complexity of the proposed algorithm can be adjusted to available processing power and can be lower than [1].

2. DESCRIPTION OF CURRENT ALGORITHMS

Let N be the length of the filter vector and M the number of filter taps to be updated at each iteration. In order to simplify the description, N/M shall be an integer, although this is not necessary for an implementation. The coefficient vector $\underline{w}(k)$ and the excitation vector $\underline{x}(k)$ of the adaptive filter can be partitioned into B_c subdivisions of length B_l:

$$\begin{aligned}\underline{w}(k) &= [w_0(k),\, w_1(k),\, \ldots,\, w_{N-1}(k)]^T \\ &= [\underline{w}_0^T(k),\, \underline{w}_1^T(k),\, \ldots,\, \underline{w}_{B_c-1}^T(k)]^T \end{aligned} \quad (1)$$

$$\begin{aligned}\underline{x}(k) &= [x(k),\, x(k-1),\, \ldots,\, x(k-N+1)]^T \\ &= [\underline{x}_0^T(k),\, \underline{x}_1^T(k),\, \ldots,\, \underline{x}_{B_c-1}^T(k)]^T \end{aligned} \quad (2)$$

with

$$\underline{w}_i(k) = [w_{iB_l}(k),\, \ldots,\, w_{(i+1)B_l-1}(k)]^T$$

and

$$\underline{x}_i(k) = [x(k-iB_l),\, \ldots,\, x(k-(i+1)B_l+1)]^T\,.$$

First, we describe the sequential block NLMS algorithm (seqB-NLMS) as an example of algorithms with a pre-fixed partial update ($M = B_l$). Using the definitions above, the sequential block NLMS can be denoted as:

$$\underline{w}_i(k+1) = \begin{cases} \underline{w}_i(k) + \mu \dfrac{e(k)\,\underline{x}_i(k)}{\underline{x}^T(k)\,\underline{x}(k)} \\ \qquad \text{if } ((k+i)\ mod\ B_c) = 0 \\ \underline{w}_i(k) \quad \text{otherwise} \end{cases} \tag{3}$$

with the adaptation error $e(k) = y(k) - \underline{w}^T(k)\,\underline{x}(k)$ and the step size parameter μ. One block of coefficients $\underline{w}_i(k)$ is updated at each iteration. The other blocks remain unchanged. This decreases the computational load of the algorithm but also cuts the speed of convergence by factor $B_c = N/M$. Since the update scheme is determined in a static way, the overhead for this algorithm is only marginal.

Another way to reduce computational load without expanding the convergence time by a factor B_c is to update only the most important coefficients or, in other words, those coefficients with larger gradient components on the error surface [1]. These coefficients are easy to detect by sorting the magnitude values of the vector $\underline{x}(k)$ and updating those coefficients associated with the M largest values of the sorted list. This leads to the following equation, here referred to as selective coefficient NLMS (selC-NLMS):

$$w_i(k+1) = \begin{cases} w_i(k) + \mu \dfrac{e(k)\,x_i(k)}{\underline{x}^T(k)\,\underline{x}(k)} \\ \qquad \text{if } i \text{ belongs to the first } M \\ \qquad\quad \text{maxima of } |\underline{x}(k)| \\ w_i(k) \quad \text{otherwise.} \end{cases} \tag{4}$$

The sorting procedure that is used to find the M largest magnitude values of the excitation vector $\underline{x}(k)$ is a running ordering algorithm called SORTLINE [5]. Since there is only one new magnitude value entering the observation window and one value leaving it at each iteration, a special sorting algorithm can be used which requires at most $2\log_2 N + 2$ comparisons per iteration.

3. IMPLEMENTATIONAL ASPECTS AND COSTS OF CURRENT ALGORITHMS

Table 1 shows the complexity of the original NLMS algorithm with full update. The complexity is expressed in terms of additions/subtractions, multiplications, data read/writes, and memory requirements. The line "Initialization" means preparing pointers or index registers for the convolution $\underline{w}^T(k)\,\underline{x}(k)$ and for the coefficient update. The complexity depends on the desired DSP and is denoted by s. Most DSPs are able to compute at least one addition and multiplication in parallel (MAC instruction). Some of them even provide memory transfers in a cycle together with arithmetic commands. The total complexity of the NLMS in terms of clock-cycles is $2N$ for the DSP 96000 (Motorola), $3N$ for the ADSP 2106x (Analog Devices), but $5N$[1] for the Pine DSP (DSP Group) and DSP 16xx (Lucent Technology).

	add	mul	mov	mem
Normalization	2	2	4	1
Calc. $e(k)$	N	N	$2N^*+2$	$2N$
Initialization	–	–	$2s$	–
Update	N	$N+1$	$2N^*+2$	–

Table 1: Complexity of NLMS algorithm

On most DSPs, memory accesses are very efficiently realized for successive data values or data values with a fixed distance. Those memory accesses are marked with an asterisk in the tables. Random access of data values usually produces overhead.

	add	mul	mov	mem
Normalization	2	2	4	1
Calc. $e(k)$	N	N	$2N^*+2$	$2N$
Initialization	–	–	$2s$	–
Update	M	$M+1$	$2M^*+2$	–

Table 2: Complexity of seqB-NLMS algorithm

The complexity of the seqB-NLMS algorithm and the selC-NLMS are shown in table 2 and table 3. The sorting

	add	mul	mov	mem
Normalization	2	2	4	1
Calc. $e(k)$	N	N	$2N^*+2$	$2N$
Initialization	–	–	$(M+1)s$	–
Update	M	$M+1$	$2M+2$	–
Sorting	$2\log_2 N + 2$		$2\lfloor (N-1)/2 \rfloor$	$2N$

Table 3: Complexity of selC-NLMS algorithm

algorithm SORTLINE in selC-NLMS requires N memory locations for the sorted list of absolute values. It adds at most $2\log_2 N + 2$ comparisons to the complexity and in the worst case up to $N-1$ memory transfers by deleting the oldest value of the sorted list and inserting the new value at its appropriate location [5]. The number of transfers can be reduced to $\lfloor (N-1)/2 \rfloor$ using a circular buffer. In order to find the actual location of the excitation values and coefficients corresponding to the M largest absolute values, one could either store links to these locations together with the sorted list (doubles memory and transfers) or select them

[1] includes rounding due to fixpoint arithmetics

during the convolution of $\underline{x}(k)$ and $\underline{c}(k)$ (adds N calculations of magnitude and N comparisons). In table 3, the first possibility was chosen because the second seems to increase the computational complexity too much.

4. SELECTIVE BLOCK UPDATE ALGORITHM

The proposed algorithm tries to combine the advantages of the seqB-NLMS algorithm and the selC-NLMS. It divides the excitation vector and the coefficient vector into B_c blocks of length $B_l = N/B_c$ (see equ. 1 and 2). Instead of looking for the M largest magnitude values it selects $M_b = M/B_l$ blocks with the largest excitation power $\underline{x}_i^T(k)\underline{x}_i(k)$ and adapts those blocks.

$$\underline{w}_i(k+1) = \begin{cases} \underline{w}_i(k) + \mu \dfrac{e(k)\,\underline{x}_i(k)}{\underline{x}^T(k)\,\underline{x}(k)} \\ \qquad \text{if } i \text{ belongs to the first } M_b \text{ maxima} \\ \qquad\quad \text{of } \underline{x}_i^T(k)\underline{x}_i(k),\ i \in (0,\ B_c - 1) \\ \underline{w}_i(k) \quad \text{otherwise} \end{cases} \tag{5}$$

By doing so, it preserves a large part of the convergence speed of the selC-NLMS algorithm but also the computational advantages of the seqB-NLMS. The excitation power can be calculated recursively, adding $2B_c$ additions/subtraction, $B_c - 1$ multiplications and B_c memory locations.

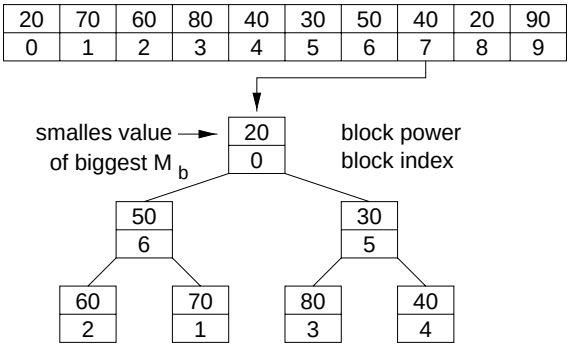

Figure 1: Sorting algorithm using HEAPSORT

The sorting of the block power values can be done using the HEAPSORT algorithm[4]. In order to do so, we build a heap of size M_b and fill it sequentially with data records, consisting of the block power value (sorting "key") and the block index number. Since all block power values change at every new iteration, we have to build up the heap from scratch in every sample period.

An example of HEAPSORT for $M_b = 7$ and $B_c = 10$ is shown in figure 1. The example demonstrates the sorting algorithm before handling the block with index number 7. Since the topmost element of the heap is also its smallest element, only one comparison is required to decide whether the new element belongs to the heap or not. If the new value is larger than the topmost element, at most $2\lfloor \log_2 (M_b + 1) \rfloor$ comparisons and the same amount of memory transfers are required to rebuild the heap.

Since there are B_c block power values, the complexity in terms of comparisons and memory transfers is (in the worst case) $2B_c\lfloor \log_2 (M_b + 1) \rfloor$ and the memory requirements are $2M_b$ as shown in table 4. Figure 2 demonstrates the contrast between the raised memory requirements of the selC-NLMS and the selB-NLMS.

	add	mul	mov	mem
Norm.	$2B_c+2$	B_c+1	$4B_c^*+2$	B_c+1
$e(k)$	N	N	$2N^*+2$	$2N$
Init.	–	–	$(B_c+1)\,s$	–
Upd.	M_bB_l	M_bB_l+1	$M_bB_l^*+2$	–
Sort.	$B_c(1+2\lfloor \log_2 M_b+1 \rfloor)$		$2B_c\lfloor \log_2 M_b+1 \rfloor$	$2M_b$

Table 4: Complexity of selB-NLMS algorithm

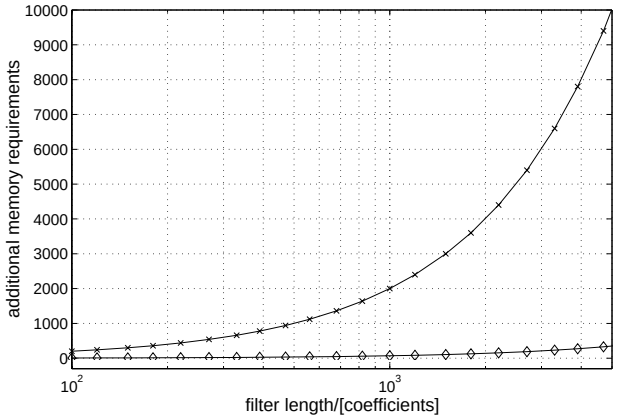

Figure 2: Additional memory requirements due to partial update; selC-NLMS ($\times$), $M = N/5$; selB-NLMS ($\diamond$), $B_l = 20,\ M_b = N/100$

5. SIMULATIONS

Simulations have been carried out in order to demonstrate and compare the performance of the existing algorithms and the selB-NLMS. The simulations have been divided into two groups: simulations with long filters (1200 coefficients) using white gaussian noise as excitation and simulations with short filters (150 coefficients) using digitized speech signals als excitation. The sampling frequency has

been 8 kHz for the excitation signal as well as for the measured room impulse responses.

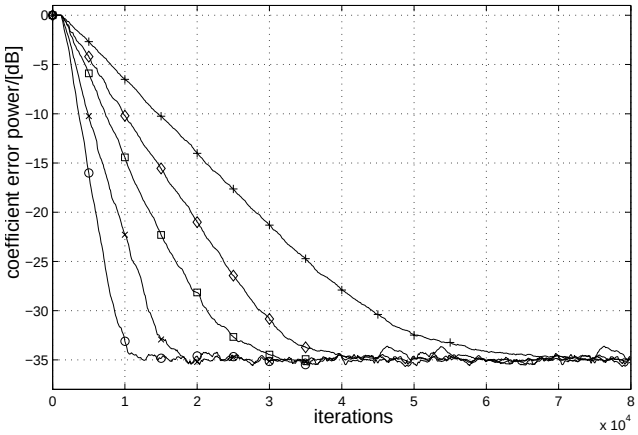

Figure 3: System mismatch, white gaussian noise, room impulse response: 2044 taps, N=1200, μ=1; seqB-NLMS (+), $M = 240$; selB-NLMS ($\diamond$), $B_l = 20,\ B_c = 12$; selB-NLMS ($\square$), $B_l = 5,\ B_c = 48$; selC-NLMS ($\times$), $M = 240$; NLMS ($\circ$), full update

Figure 3 shows the system mismatch in terms of coefficient error power for white gaussian noise. The performance loss of the selC-NLMS compared to the NLMS is marginal. As expected, the selB-NLMS does not perform as well as the selC-NLMS, because there are too much small excitation values in each selected block. The smaller the blocks are, the better the results (for $B_l = 1$, the selB-NLMS and the selC-NLMS are identical). On the other hand, the smaller the blocks are, the bigger the computational load.

In figure 4, speech (male and female speakers) has been used as excitation. In order to reduce the effects of instationarity of the speech data, the figure shows an ensemble average of 15 independent simulations with different speech signals. The loss of convergence speed of the selC-NLMS and the selB-NLMS compared to the full update is smaller with correlated input. This is an expected result, because speech data has a lot of periods without sufficient excitation (like pauses between words). Thus, by stopping the adaptation of blocks belonging to those periods, the convergence speed gets not decreased.

6. CONCLUSIONS

In this paper, an algorithm with low memory requirements has been presented that updates only selected portions of the filter vector. In contrast to the selC-NLMS, its additional memory requirements are low, so that it is better suited for real-time implementations on low-cost DSPs with restricted

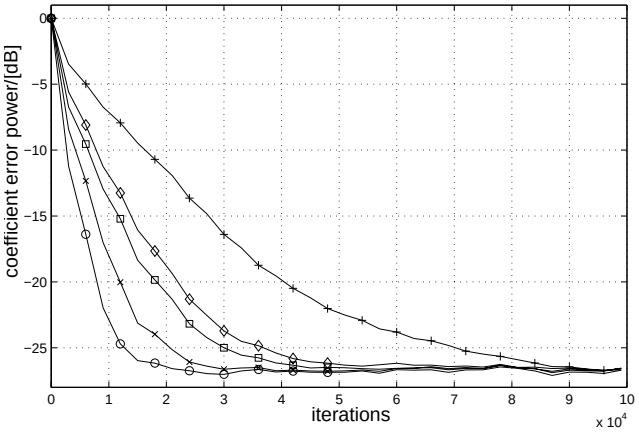

Figure 4: System mismatch, speech, ensemble average over 15 simulations, room impulse response: 151 taps, N=150, $\mu = 0.3$; seqB-NLMS (+), $M = 24$; selB-NLMS ($\diamond$), $B_l = 6,\ B_c = 4$; selB-NLMS ($\square$), $B_l = 3,\ B_c = 8$; selC-NLMS ($\times$), $M = 24$; NLMS ($\circ$), full update

resources. Since the presented selB-NLMS is a trade-off between selC-NLMS and seqB-NLMS, one can implement a long adaptive filter and adjust the parameters of the algorithm according to the requirements for memory, performance and load. The performance of the selB-NLMS has been demonstrated by simulations.

7. REFERENCES

[1] T. ABOULNASR AND K. MAYYAS. Selective coefficient update of gradient-based adaptive algorithms. In "Proc. ICASSP 1997, München, Germany", pp. 1929–1932 (1997).

[2] S. C. DOUGLAS. Adaptive filters employing partial updates. *IEEE Trans. on Circuits and Systems* **44**(3), 209–216 (Mar. 1997).

[3] S. HAYKIN. "Adaptive Filter Theory". Information and System Sciences Series. Prentice-Hall International, Inc., 3rd ed. (1996).

[4] D. E. KNUTH. "Sorting and Searching", vol. 3 of "The Art of Computer Programming". Addision-Wesley, 1st ed. (1973).

[5] I. PITAS. Fast algorithms for running ordering and max/min calculation. *IEEE Trans. on Circuits and Systems* **36**(6), 795–804 (June 1989).

[6] J. TREICHLER, J. JOHNSON, AND M. G. LARIMORE. "Theory and Design of Adaptive Filters". Wiley-Interscience, New York (1987).