Robust fast affine projection algorithm for acoustic echo cancellation

Ville Myllylä

Darmstadt University of Technology, Institute of Communication Technology,
Merckstrasse 25, 64283 Darmstadt, Germany
myllylav@nesi.tu-darmstadt.de, ville.myllyla@nokia.com

Abstract— The NLMS algorithm is probably still the most used algorithm for acoustic echo cancellation systems due to its well-known useful characteristics: simplicity and robust performance even if implemented in fixed-point arithmetic. However, it suffers from a slow convergence rate with speech signals. On the other hand, the affine projection (AP) algorithm, which is a generalization of the NLMS algorithm, has better convergence properties but is computationally demanding and has some numerical problems. There exists a computationally more efficient version of the AP algorithm, namely, the fast affine projection (FAP) algorithm. But it still has numerical instability problems. In this paper these problems are hindered by introducing some modifications to the FAP. Eventually, this new algorithm comes very close to the adaptive decorrelation NLMS approach, but it is computationally less demanding.

I. INTRODUCTION

Typically, in acoustic echo control approaches an adaptive filter algorithm is used to estimate the unknown acoustic echo path and to form an echo replica that is then subtracted from the microphone signal. As a result, the acoustic echo is reduced to some extent. This is briefly depicted in Fig. 1., where, also, some notations are introduced. For further details see, e.g., [1]. So far, in these implementations the NLMS algorithm has been the most popular adaptation algorithm since it is robust and simple. However, it convergences slowly with correlated signals such as speech, which is in echo control applications a reality.

The aim of this work was to derive an algorithm that has a better convergence rate than the NLMS algorithm and would still be economical and robust to implement, i.e., the algorithm should be computationally efficient and should not have fixed-point instability problems. The AP algorithm [2] was chosen as a starting point, since it has good convergence characteristics [2, 3] and does not need as much computation as, e.g., the RLS algorithm. Like in the RLS case, there exist computationally fast versions for the AP algorithm [4, 5].

The FAP algorithm provides significant computational savings, so that it requires only slightly more computational power than the NLMS. It uses a sliding-window fast RLS method to calculate the inverse of the data correlation matrix of order p, where p is the projection order of the FAP algorithm and it is much smaller than the filter length N. As pointed out in [6], the use of the fast RLS introduces some problems: The program code for the sliding-window fast RLS algorithm is difficult to implement, memory-intensive, and

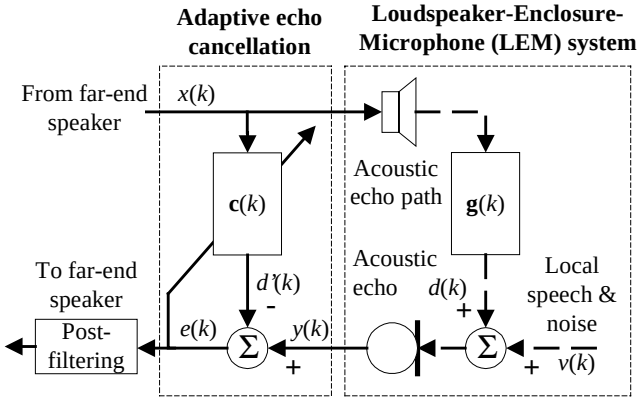

Figure 1: Application environment, Loudspeaker-Enclosure-Microphone system, and the adaptive echo cancellation approach.

potentially numerically-unstable unless special care is taken in its implementation.

Because of these problems some other solutions for the inversion of the data correlation matrix have been suggested. In [6] orthogonal transforms are used to approximately calculate the affine projection avoiding the use of the fast sliding-windowed RLS. But this solution has still error accumulation problems if implemented in fixed-point arithmetic. As a solution, the use of periodic restart or leaky integration has been suggested [6]. Both [7] and [8] use a standard sliding-window RLS-type approach by utilising the matrix inversion lemma twice. However, as it is well known, this approach is also subject to error accumulation.

In [9], the data correlation matrix was approximated to have Toeplitz form. Since there exist efficient algorithms to compute the inverse of Toeplitz matrices, this solution was supposed to maintain the computational efficiency of the FAP algorithm and, in addition, to be robust in the fixed-point environment. However, the paper does not show how the actual inverse is achieved.

This is the framework where we begin. At first, we make one more simplification in the algorithm of [9], namely, the error vector is reduced to a scalar, which provides some computational savings and makes the inversion even easier to compute. Then, the Levinson-Durbin algorithm is applied for the inversion.

This new algorithm, the robust FAP, is derived and its properties are discussed in the next chapter. Then, some simulation results are shown in order to verify the theoretical derivations. Finally, conclusions are given.

II. ROBUST FAST AFFINE PROJECTION ALGORITHM

In the consequent text, the following notation is used: italics for scalar variables, bold face for vectors and matrices, of which matrices are shown in capital letters. The size of the matrix or vector is also shown in a subscript, where N denotes the adaptive filter length and p the projection order of the affine projection algorithms. In connection with scalars, a subscript denotes a location index in a vector, e.g., $\mathbf{s}_p(k) = [s_1(k), s_2(k),\ldots, s_p(k)]^T$. k is used as a time index and a superscript T denotes the transpose. Normally scalars are indexed with the time index like in the excitation vector $\mathbf{x}_N(k) = [x(k), x(k-1),\ldots, x(k-N+1)]^T$.

A. Fast affine projection algorithm

At first, the AP algorithm is introduced:

$$\mathbf{e}_p(k) = \mathbf{y}_p(k) - \mathbf{X}_{N,p}^T(k)\mathbf{c}_N(k-1), \quad (1)$$

$$\mathbf{R}_{p,p}(k) = \left[\mathbf{X}_{N,p}^T(k)\mathbf{X}_{N,p}(k) + \delta\mathbf{I}_{p,p}\right], \quad (2)$$

$$\mathbf{c}_N(k) = \mathbf{c}_N(k-1) + \mu(k)\mathbf{X}_{N,p}(k)\mathbf{R}_{p,p}^{-1}(k)\mathbf{e}_p(k), \quad (3)$$

where

$$\mathbf{X}_{N,p}(k) = \left[\mathbf{x}_N(k), \mathbf{x}_N(k-1),\ldots,\mathbf{x}_N(k-p+1)\right] \quad (4)$$

is the N by p excitation matrix, or data sample matrix. In addition, $\delta\mathbf{I}$ is a regularization term, where δ is a regularization parameter and $\mathbf{I}$ is an identity matrix, and $\mu(k)$ is a step-size parameter. As can be seen from the equations, the algorithm is quite costly: roughly $2Np + O(p^3)$. Certainly, there is a need for a computationally faster version, i.e., for the FAP algorithm.

In Table 1 a so called simplified version of the FAP algorithm is presented as suggested in [9]. The difference compared to the original FAP algorithm is that the inverse of the data correlation matrix, Eq.(2), is calculated directly in step (A.4), not utilising RLS-like methods. The residual error vector calculation, Eq.(1), is now done in steps (A.1), (A.2) and (A.3) utilising a recursive approach and an alternate filter weight vector $\mathbf{z}_N(k)$.

Table 1. Simplified FAP algorithm.

$$\mathbf{r}_{p-1}(k) = \mathbf{r}_{p-1}(k-1) + x(k)\mathbf{x}_{p-1}(k-1) - x(k-N)\mathbf{x}_{p-1}(k-N-1) \quad (A.1)$$

$$e(k) = y(k) - \mathbf{x}_N^T(k)\mathbf{z}_N(k) - \mathbf{r}_{p-1}^T(k)\mathbf{s}_{p-1}(k-1) \quad (A.2)$$

$$\mathbf{e}_p(k) = \begin{bmatrix} e(k) \\ (1-\mu(k))\mathbf{e}_{p-1}(k-1) \end{bmatrix} \quad (A.3)$$

$$\mathbf{g}_p(k) = \mathbf{R}_{p,p}^{-1}(k)\mathbf{e}_p(k) \quad (A.4)$$

$$\mathbf{s}_p(k) = \begin{bmatrix} 0 \\ \mathbf{s}_{p-1}(k-1) \end{bmatrix} + \mu(k)\mathbf{g}_p(k) \quad (A.5)$$

$$\mathbf{z}_N(k+1) = \mathbf{z}_N(k) + \mathbf{x}_N(k-p+1)s_p(k) \quad (A.6)$$

The filter update, Eq. (3), is done according to the steps (A.4), (A.5) and (A.6). The actual echo path estimate filter, $\mathbf{c}_N(k)$, is not computed, but an approximation of it, $\mathbf{z}_N(k)$, instead. In step (A.2), while calculating the filter error, this approximation is countered by a correction term. This technique saves us performing the costly matrix vector multiplications in Eq.(3).

B. A new formulation

In order to compute the inverse of the data correlation matrix efficiently in step (A.4) it is possible to use RLS-like methods or to approximate the data correlation matrix as a Toeplitz form as suggested in [9]. If we choose, for example, a projection order of 4 and forget the regularization for a moment, Eq.(2) becomes:

$$\mathbf{R}_{4,4}(k) = \left[\mathbf{X}_{N,4}(k)^T\mathbf{X}_{N,4}(k)\right] = \begin{bmatrix} r_0(k) & r_1(k) & r_2(k) & r_3(k) \\ r_1(k) & r_0(k-1) & r_1(k-1) & r_2(k-1) \\ r_2(k) & r_1(k-1) & r_0(k-2) & r_1(k-2) \\ r_3(k) & r_2(k-1) & r_1(k-2) & r_0(k-3) \end{bmatrix}, \quad (5)$$

where $r_\tau(k) = \sum_{i=0}^{N-1} x(k-i)x(k-i-\tau)$ is an estimate of the autocorrelation at lag τ and at time instant k based on the past N input data. If $N >> p$, as it usually is, the following holds rather well:

$$r_\tau(k) \approx r_\tau(k-1) \approx \ldots \approx r_\tau(k-p+1). \quad (6)$$

Utilizing this approximation, the right hand side of Eq.(5) becomes a Toeplitz matrix, i.e., it has constant values along its main diagonals:

$$\hat{\mathbf{R}}_{4,4}(k) = \begin{bmatrix} r_0(k) & r_1(k) & r_2(k) & r_3(k) \\ r_1(k) & r_0(k) & r_1(k) & r_2(k) \\ r_2(k) & r_1(k) & r_0(k) & r_1(k) \\ r_3(k) & r_2(k) & r_1(k) & r_0(k) \end{bmatrix}. \quad (7)$$

Furthermore, it is symmetric. These properties make the computation of its inverse very efficient. However, before proceeding with the inverse, one more simplification is made.

As suggested by many authors, the error vector can be reduced to a scalar. As seen in step (A.3), this is totally valid for a step-size value of 1. For other step-sizes it becomes an approximation that holds quite well for step-size values close to 1. This approximation provides considerable computational savings. At first, step (A.3) can be omitted. In step (A.4), we note that only the first column of the data correlation matrix inverse is needed. This fact leads to a very efficient inversion method.

The Levinson-Durbin algorithm (LDA), see e.g. [10], solves the following equation for $\mathbf{a}$ and E:

$$\mathbf{T}\begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = [E,0,\ldots,0]^T, \quad (8)$$

where $\mathbf{a}$ is a predictor coefficient vector, E is the square of the prediction error, and $\mathbf{T}$ is a symmetric Toeplitz matrix composed of the autocorrelation coefficients, i.e., like Eq.(7). Dividing both sides of Eq.(8) by E, results in:

$$\mathbf{T}\frac{1}{E}\begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = [1,0,\ldots,0]^T. \quad (9)$$

In other words $\frac{1}{E}\begin{bmatrix}1\\ \mathbf{a}\end{bmatrix}$ is the first column of the inverse matrix $\mathbf{T}^{-1}$. Thus, the Levinson-Durbin algorithm is all that is needed for the "inversion" of the data correlation matrix. Furthermore, almost all the autocorrelation estimate values needed to specify the matrix are already available after step (A.1). Only the autocorrelation value at zero lag is missing but can be easily added. The resulting algorithm is presented in Table 2.

Table 2. Robust FAP algorithm.

I Error calculation

Note: $\mathbf{r}_{2\ldots p}(k) = \left[r_2(k), r_3(k), \ldots, r_p(k)\right]^T$

$$\mathbf{r}_p(k) = \mathbf{r}_p(k-1) + x(k)\mathbf{x}_p(k) - x(k-N)\mathbf{x}_p(k-N) \quad \text{(B.1)}$$

$$e(k) = y(k) - \mathbf{x}_N^T(k)\mathbf{z}_N(k) - \mathbf{r}_{2\ldots p}^T(k)\mathbf{s}_{p-1}(k-1) \quad \text{(B.2)}$$

II Levinson-Durbin

Initialilize: $E_0(k) = r_1(k)$, $C_0(k) = r_2(k)$

Note: $\hat{\mathbf{a}}_p(k) = \left[a_p, a_{p-1}, \ldots, a_1\right]^T$ means a reverse of $\mathbf{a}_p(k)$.

For i = 1 to p-1

$$K_i(k) = -C_{i-1}(k)/E_{i-1}(k) \quad \text{(B.3.1)}$$

$$\mathbf{a}_i(k) = \begin{bmatrix}\mathbf{a}_{i-1}(k) + K_i(k)\hat{\mathbf{a}}_{i-1}(k)\\ K_i(k)\end{bmatrix} \quad \text{(B.3.2)}$$

$$E_i(k) = E_{i-1}(k) + K_i(k)C_{i-1}(k) \quad \text{(B.3.3)}$$

$$C_i(k) = \mathbf{r}_{2\ldots i+2}(k)\begin{bmatrix}\hat{\mathbf{a}}_i(k)\\ 1\end{bmatrix} \quad \text{(B.3.4)}$$

end

III Update of the approximate weight vector

$$\mathbf{g}_p(k) = \frac{e(k)}{E_{p-1}(k)}\begin{bmatrix}1\\ \mathbf{a}_{p-1}(k)\end{bmatrix} \quad \text{(B.4)}$$

$$\mathbf{s}_p(k) = \begin{bmatrix}0\\ \mathbf{s}_{p-1}(k-1)\end{bmatrix} + \mu(k)\mathbf{g}_p(k) \quad \text{(B.5)}$$

$$\mathbf{z}_N(k+1) = \mathbf{z}_N(k) + \mathbf{x}_N(k-p+1)s_p(k) \quad \text{(B.6)}$$

C. Algorithm properties

1) Stability issues

The Levinson-Durbin algorithm (Table 2, part II) provides good properties against instability. At first, the reflection coefficient, or so called PARCOR coefficient, K (Table 2, step B.3.1) can be used to monitor the stability of the algorithm (to be exact, the definition of the PARCOR coefficient has the opposite sign). The necessary and sufficient condition for the Toeplitz matrix in Eq.(8) to be positive definite, i.e., to have an inverse, is $|K| < 1$, (see, e.g., [11]). Thus, if in any stage it becomes close to one, the LDA can be terminated and, for that iteration, the step-size can be simply assigned to zero to avoid further problems. It is not necessary to use any regularization. However, less iterations will be lost if a relatively small value is added to the first autocorrelation coefficient $r_1(k)$. Alternatively, an ordinary NLMS-update can be performed in case of inversion problems.

A second advantage of the use of the LDA is that it is recursive only within the projection order p and not with the time instant k. That is, there is no error accumulation between time instants since all values within the LDA are computed from scratch for each time instant k.

In the error calculation part, step (B.1) should be implemented with an appropriate rounding. Otherwise, this part does not permit error accumulation. Similarly, the last part, the update of the approximate weight vector, is apparently not a source for instability problems.

2) Complexity

The robust FAP algorithm is rather efficient. The first part, or the error calculation, takes $N + 3p$ multiplications. The second part, the LDA, uses about $p^2 + 0.5p$ multiplications and p - 1 divisions. Finally, the update of the approximate weight vector consumes about $N + p$ multiplications and 1 division. Altogether, this makes $2N + p^2 + 4.5p$ multiplications and p divisions, i.e., roughly $2N + p^2 + 6p$ operations. In Table 3, the robust FAP is compared to other FAP-like algorithms.

Table 3. Computational complexity (number of divisions and multiplications) of different FAP-like algorithms. The additional (NLMS) cost $2N$ is omitted.

Algorithm	**Upper limit**	**Detailed**	$p = 4$	$p = 10$
Simplified FAP [9]	$O(p^3)$	-	-	-
FAP-RLS [7]	$O(5p^2)$	$5p^2$	80	500
FAP-RLS [8]	$O(3p^2)$	$3p^2 + 12p$	96	420
Toeplitz FAP [9]	$O(2p^2)$	-	-	-
Robust FAP	$\boldsymbol{O(p^2)}$	$\boldsymbol{p^2 + 6p}$	**40**	**160**
FAP [4, 5]	$O(20p)$	$20p$	80	200
Approximate impl. [6]	$O(16p)$	$9p + p\log_2 p - 3$	41	121
FAP with scalar error	$O(14p)$	$14p$	56	140

3) Comparison to pre-whitening NLMS

The new algorithm becomes very close to adaptive decorrelation NLMS approaches [12, 13], where the predictor coefficients have to be computed, too. Utilising these coefficients, the adaptation signals for NLMS are then whitened. Finally, the echo replica is generated by filtering the excitation signal with the estimated echo path impulse response. In this approach, the echo path impulse response filtering has to be conducted twice, namely, for

speech and whitened speech. While the filter length N is the dominating part in the algorithm complexity, this makes the pre-whitening approach more demanding even if the predictor coefficients are updated only periodically.

4) Discussion

There are several points in the algorithm where different choices can be made. At first, different windowing approaches can be used for recursive updating of the autocorrelation coefficients in step (B.1) instead of a rectangular window. However, this might require additional computations, since in step (B.2), rectangular windowing is needed to maintain the FAP structure. Nevertheless, for example, exponential windowing can be used to enhance the stability properties of the coefficients. Furthermore, the updating of the predictor coefficients may be periodic in order to save computations. Finally, a full inverse matrix can be calculated quite efficiently from the predictor coefficients by utilising the Gohberg-Semencul relation [14] in case the use of the full vector error is desired.

III. Simulation Results

Simulations were conducted to get an idea about the algorithm behavior. The performance of the NLMS, the AP and the robust FAP algorithm were compared in the ideal adaptation case, i.e., there was neither near-end speech nor background noise and the used echo path length was the same as the filter length, or 300 in 8 kHz sampling frequency. In this case, a fixed step-size of 1 was utilized and the projection order p for the projection algorithms was 4. Twelve different speech sentences (6 male, 6 female), each of length 5 seconds, were put together to form a 60 seconds long excitation signal. There were two real impulse responses measured from a car that were switched after each sentence, i.e., in every 5 seconds, to simulate echo path changes. Then the average of the system error norm,

$$\Delta g_{dB}(k) = 10\log\left(\left\|\mathbf{c}(k) - \mathbf{g}(k)\right\|^2\right), \qquad (10)$$

was computed over these twelve sequences resulting 5 seconds long average. Thus, the average contains one initial convergence curve and 11 convergence curves after echo path changes. The first two seconds of these averages are shown in Fig. 2.

From the figure, it can be seen that the new algorithm performs better than the NLMS, but is yet a bit slower than the original AP algorithm. The main reason for this is the fact that the inversion in the AP algorithm was performed with the built-in Matlab® *inv*(.)-function, which handles instability problems very well. In contrast, in the new algorithm version, an ordinary NLMS step has to be performed every now and then because of the instability problems in the LDA.

IV. Conclusions

In this paper, a robust FAP algorithm was formulated, which is supposed to be robust even if implemented in fixed-point arithmetic. Furthermore, it is computationally very efficient. This was achieved by reducing the filter error vector to a scalar, simplifying the data correlation matrix to become of a Toeplitz form, and applying the Levinson-Durbin algorithm on its partial inverse. The new algorithm performs better than the NLMS and is only slightly slower than the original affine projection algorithm due to simplifications in its derivation.

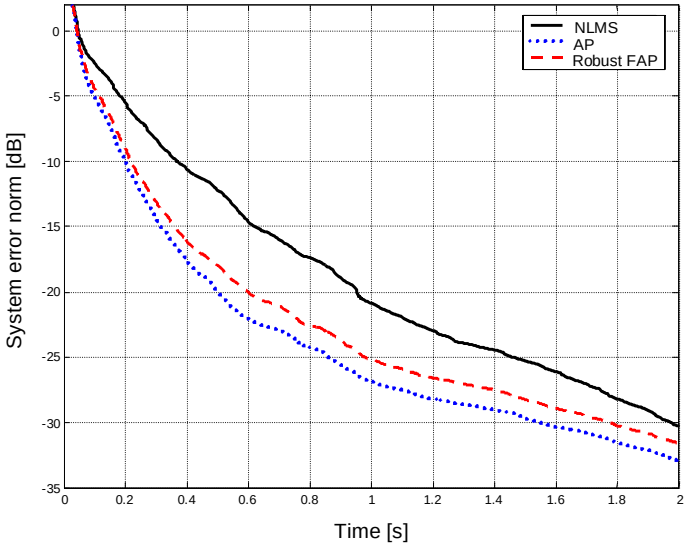

Figure 2: Comparison of the different algorithms in an ideal adaptation case.

V. References

[1] Breining C., Dreiseitel P., Hänsler E., et al, "Acoustic echo control - an application of very-high-order adaptive filters", *IEEE Signal Processing magazine*, Vol. 16, No. 4, 1999.

[2] Ozeki K., Umeda T., "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties", *Electronics and Communications in Japan*, Vol. 67-A, No. 5, 1984.

[3] Sankaran S., Beex A., "Convergence behavior of affine projection algorithms", *IEEE Transactions on signal processing*, Vol. 48, No. 4, 2000.

[4] Tanaka M., Kaneda Y., Makino S., and Kojima J., "A fast projection algorithm for adaptive filtering", *IEICE Trans. fundamentals*, Vol. E78-A, No. 10, 1995.

[5] Gay S., Tavadia S., "The fast affine projection algorithm", Proc. *ICASSP-95*, 1995.

[6] Douglas S., "Efficient approximate implementations of the fast affine projection algorithm using orthogonal transforms", Proc. *ICASSP-96*, 1996.

[7] Kaneda Y., Tanaka M., and Kojima J., "An adaptive algorithm with fast convergence for multi-point sound control", *Proc. ACTIVE-95*, 1995.

[8] Liu Q., Champagne B., Ho K., "On the use of a modified fast affine projection algorithm in subbands for acoustic echo cancellation", *Proc. 7th IEEE DSP Workshop 96*, 1996.

[9] Oh S., Linebarger D., Priest B., and Ragothaman B., "A fast affine projection algorithm for an acoustic echo canceller using a fixed-point DSP processor", *Proc. ICASSP-97*, 1997.

[10] Strobach P., "New forms of Levinson and Schur algorithms", *IEEE Signal Processing magazine*, Vol. 8, No. 1, 1991.

[11] Wang Y., Krishna H., Krishna B., "Split Levinson algorithm is weakly stable", *Proc. ICASSP-89*, 1989.

[12] Yamamoto S., Kitayama J., Tamura J., and Ishigami H., "An adaptive echo canceller with linear predictor", *Trans. IECE of Japan*, Vol. 62, No. 11, 1991.

[13] Frenzel R., Hennecke M., "Using prewhitening and stepsize control to improve the performance of the LMS algorithm for acoustic echo compensation", *Proc. ISCAS-92*, Vol. 4, 1992.

[14] Cernuschi-Frias B., "A derivation of the Gohberg-Semencul relation", *IEEE Transactions on Signal Processing*, Vol. 39, No. 1, 1991.