The Gauss-Seidel fast affine projection algorithm for multichannel active noise control and sound reproduction systems

Martin Bouchard
School of Information Technology and Engineering, University of Ottawa

Felix Albu
Faculty of Electronics & Telecommunications, Politehnica University of Bucharest

Short title: "Multichannel Gauss-Seidel Fast Affine Projection ..."

SUMMARY

In the field of adaptive filtering, the fast implementations of affine projection algorithms are known to provide a good tradeoff between convergence speed and computational complexity. Such algorithms have recently been published for multichannel active noise control systems. Previous work reported that these algorithms can outperform more complex recursive least-squares algorithms when noisy plant models are used in active noise control systems. This paper proposes a new fast affine projection algorithm for multichannel active noise control or sound reproduction systems, based on the Gauss-Seidel solving scheme. The proposed algorithm has a lower complexity than the previously published algorithms, with the same convergence speed and the same good performance with noisy plant models, and a potential for better numerical stability. It provides the best performance/cost ratio. Details of the algorithm and its complexity are presented in the paper, with simulation results to validate its performance.

Keywords:

- multichannel active noise control
- adaptive filtering
- fast affine projection algorithms

This work was supported by a NSERC grant

Sponsor:

Corresponding author :

Martin Bouchard
School of Information Technology and Engineering,
University of Ottawa, 800 King Edward
Ottawa (Ontario), K1N 6N5, Canada
Tel.: (613) 562-5800 ext 6190
Fax: (613) 562-5187
e-mail : bouchard@site.uottawa.ca

1. INTRODUCTION

Active noise control (ANC) systems [1],[2] work on the principle of destructive interference between an original "primary" disturbance sound field measured at the location of K "error" sensors (typically microphones), and a "secondary" sound field that is generated by J control actuators (typically loudspeakers). In ANC systems, a common approach is to use adaptive FIR filters, in either feedforward or feedback control configurations. A similar problem is found in sound reproduction systems [3],[4], where the acoustic response of a room between J actuators and K sensors needs to be inverted and compensated. Some applications are transaural audio or multichannel sound wavefield synthesis systems, where given waveforms have to be reproduced at some sensor locations. Figures 1 and 2 show block-diagrams of monochannel implementations of feedforward active noise control and sound reproduction systems, using adaptive FIR filters. The systems in Figs. 1 and 2 are delay-compensated, i.e. the stabilization time on the error signals caused by updates to the adaptive FIR filter coefficients has been eliminated by minimizing an alternative error signal [5],[6]. This structure has sometimes been called the “modified filtered-x” structure [7], and the use of this structure will be assumed in the rest of this paper. Also, the algorithms to be introduced in this paper are for feedforward adaptive active noise control, although it is a simple task to adapt them to either feedback adaptive active noise control (with internal model control (IMC) structures) [8] or sound reproduction systems [7].

In the field of adaptive filtering it is well known that fast affine projection algorithms can produce a good tradeoff between convergence speed and computational complexity. Although these algorithms typically do not provide the same convergence speed as recursive least-squares algorithms, they can provide a much improved convergence speed compared to stochastic gradient descent algorithms, without the high increase of the computational load or the instability often found in recursive least-squares algorithms, especially for multichannel systems [7],[9]. Many multichannel

fast affine projection algorithms have been previously published for acoustic echo cancellation, but the problem of active noise control or sound reproduction is a very different one. Indeed active noise control and sound reproduction systems are obviously control or inverse problems, while acoustic echo cancellation is an identification problem (with of course its own additional constraints such as double-talk, etc.). This leads to different structures (such as the filtered-x structure in the filtered-x LMS [10] instead of the standard adaptive FIR filter structure, or other structures such as adjoint [7], filtered-ε [10] or inverse filtered-x [11]), to different dimensions for the different signals, and obviously to different multichannel algorithms.

Adaptive filtering algorithms based on fast affine projection algorithms have recently been published for multichannel active noise control [12], partly based on earlier work in [13]. It was shown that they can provide the expected tradeoff between convergence speed and computational complexity for multichannel ANC systems. It was also reported that in the realistic case where noisy plant models are used (i.e. for algorithms using filtered-x types of structures), fast affine projection algorithms can be much more robust to plant model noise than more complex algorithms based on recursive least-squares, and therefore they can achieve a better convergence performance at a lower cost. The previously published fast affine projection algorithm for multichannel active noise control had a built-in sliding window recursive least-squares algorithm, because other fast affine projection algorithms with built-in fast transversal filters algorithms (or fast recursive least-squares algorithms) were found to be numerically unstable for the multichannel active noise control implementation. In Section 2 of this paper, a new fast affine projection algorithm using the Gauss-Seidel inversion scheme is introduced for multichannel active noise control. It is an extension of the Gauss-Seidel Fast Affine Projection algorithm (or GS-FAP), published in [14], to the problem of multichannel active noise control. The complexity of the new algorithm is evaluated in Section 3, where it is shown that it is significantly lower than the previously published multichannel ANC fast affine projection algorithm, and typically of the same order than least-mean-squares algorithms.

Simulation results in Section 4 show that the convergence performance of the new proposed algorithm is identical to the previously published multichannel ANC fast affine projection algorithm, including its superior performance with noisy plant models. It is also shown that overall the new proposed algorithm produces the best performance/cost ratio.

2. GAUSS-SEIDEL FAST AFFINE PROJECTION ALGORITHM FOR MULTICHANNEL ACTIVE NOISE CONTROL

In this section the equations of the fast affine projection algorithm for multichannel ANC based on the Gauss-Seidel solving scheme are presented. The original fast affine projection algorithm can be found in [15],[16], and the modifications required in order to adapt the algorithm to the problem of active noise control (or sound reproduction) are discussed in [13]. These modifications are required because of the computation of "auxiliary" coefficients by fast affine projection algorithms, instead of the "normal" time domain coefficients which would directly link the signals from reference input sensors to the output actuators in ANC systems. Since the new proposed algorithm is based on the GS-FAP algorithm [14] and it is developed for multichannel ANC systems using a delay-compensated or modified filtered-x (MFX) structure, it is therefore called the MFXGS-FAP for multichannel ANC. The previously published fast affine projection algorithm for multichannel ANC was using a similar delay-compensated structure, but had a built-in recursive least-squares (RLS) scheme instead of the Gauss-Seidel scheme, therefore it was called the MFXFAP-RLS algorithm for multichannel ANC [12]. In the equations below, an emphasis will be put on how the different signals in multichannel ANC must be structured so that the fast affine projection algorithm can be used. The dimensions of the different resulting signals will also be emphasized. To describe the MFXGS-FAP algorithm, the following notation is first defined (refer to Fig. 1):

I number of reference sensors in an ANC system

J	number of actuators in an ANC system
K	number of error sensors in an ANC system
L	length of the adaptive FIR filters
N	affine projection order
M	length of (fixed) FIR filters modeling the plant (transfer functions between the actuators and the error sensors) in an ANC system
$x_i(n)$	value at time n of the i^{th} reference signal
$y_j(n)$	value at time n of the j^{th} actuator signal
$d_k(n)$	value at time n of the primary sound field at the k^{th} error sensor
$e_k(n)$	value at time n of the k^{th} error sensor
$\hat{d}_k(n)$	estimate of $d_k(n)$, computed in delay-compensated modified filtered-x structures
$\hat{e}_k(n)$	value at time n of the alternative error signal for the k^{th} sensor, computed in delay-compensated modified filtered-x structures
$h_{j,k,m}$	value of the m^{th} coefficient in the (fixed) FIR filter modeling the plant between $y_j(n)$ and $e_k(n)$
$v_{i,j,k}(n)$	value at time n of the filtered reference signal, i.e. the signal obtained by filtering the $x_i(n)$ signal with the plant model $\boldsymbol{h}_{j,k}$ filter

$\hat{w}_{i,j,l}(n)$ value at time n of the l^{th} auxiliary coefficient in the adaptive FIR filter linking $x_i(n)$ and $y_j(n)$. These auxiliary coefficients are the coefficients computed by fast affine projection algorithms.

$\boldsymbol{R}(n)$ $KN \times KN$ auto-correlation matrix. $\boldsymbol{R}(n)$ is initialized as an identity matrix multiplied by the scalar δ, where δ is a regularization factor to be adjusted.

$\overline{\boldsymbol{R}}(n)$ the top left $K(N-1) \times K(N-1)$ values of $\boldsymbol{R}(n)$

$\boldsymbol{P}(n)$ an inverse correlation matrix of size $KN \times K$ (first K columns of the inverse of $\boldsymbol{R}(n)$)

$\underline{\boldsymbol{b}}$ a matrix of size $KN \times K$ whose elements are zeros except for the top $K \times K$ values which are set to an identity matrix

$\boldsymbol{r}_j(n)$ correlation vector of size $K(N-1) \times 1$ associated with the j[th] actuator, initialized with zero values

$\boldsymbol{r}(n)$ correlation matrix of size $K(N-1) \times K$, initialized with zero values

$\bar{\boldsymbol{r}}(n)$ correlation matrix of size $K \times K$, initialized with zero values

$\mathfrak{n}(n)$ $1 \times KN$ error vector

$\overline{\mathfrak{n}}(n)$ the first $K(N-1)$ columns of $\mathfrak{n}(n)$

$\mathfrak{n}_{N-1}(n)$ the last K columns of $\mathfrak{n}(n)$

$\boldsymbol{x}''_j(n)$ vector of size $IJ \times 1$ sparsely filled with the I values of $x_i(n)$. The rows to be filled with the $x_i(n)$ values are the same I rows associated with the j[th] actuator in the signal $v(n)$.

$$\boldsymbol{h}_{j,k} = \left[h_{j,k,1},\ h_{j,k,2},\ \cdots\ h_{j,k,M} \right]^T \qquad (\text{size } M \times 1)$$

$$\boldsymbol{v}_{i,j,k}(n) = \left[v_{i,j,k}(n),\ v_{i,j,k}(n-1),\ \cdots\ v_{i,j,k}(n-L+1) \right]^T \qquad (\text{size } L \times 1)$$

$$\boldsymbol{x}_i(n) = \left[x_i(n),\ x_i(n-1),\ \cdots\ x_i(n-L+1) \right]^T \qquad (\text{size } L \times 1)$$

$$\boldsymbol{x}'_i(n) = \left[x_i(n),\ x_i(n-1),\ \cdots\ x_i(n-M+1) \right]^T \qquad (\text{size } M \times 1)$$

$$\boldsymbol{y}_j(n) = \left[y_j(n),\ y_j(n-1),\ \cdots\ y_j(n-M+1) \right]^T \qquad (\text{size } M \times 1)$$

$$\hat{\boldsymbol{d}}(n) = \left[\hat{d}_1(n), \hat{d}_2(n), \cdots \hat{d}_K(n) \right] \qquad (\text{size } 1 \times K)$$

$$\hat{\boldsymbol{e}}(n) = \left[\hat{e}_1(n), \hat{e}_2(n), \cdots \hat{e}_K(n) \right] \qquad (\text{size } 1 \times K)$$

$$v(n) = \begin{bmatrix} v_{1,1,1}(n) & \cdots & v_{1,1,K}(n) \\ \vdots & \ddots & \vdots \\ v_{I,J,1}(n) & \cdots & v_{I,J,K}(n) \end{bmatrix} \qquad (\text{size } IJ \times K)$$

$$\boldsymbol{v}(n) = \begin{bmatrix} v(n) \\ \vdots \\ v(n-L+1) \end{bmatrix} \qquad (\text{size } IJL \times K)$$

$$\tilde{\boldsymbol{a}}(n) = \left[v(n-1) \quad \cdots \quad v(n-N+1) \right]^T \ (\text{size } K(N-1) \times IJ)$$

$$\hat{\boldsymbol{w}}(n) = \left[\left[\hat{w}_{1,1,1}(n) \quad \cdots \quad \hat{w}_{I,J,1}(n) \right] \quad \cdots \quad \left[\hat{w}_{1,1,L}(n) \quad \cdots \quad \hat{w}_{I,J,L}(n) \right] \right]^T \ (\text{size } IJL \times 1)$$

$$\boldsymbol{R}(n) = \begin{bmatrix} \bar{\boldsymbol{r}}(n) & \boldsymbol{r}^T(n) \\ \boldsymbol{r}(n) & \bar{\boldsymbol{R}}(n-1) \end{bmatrix} \qquad (\text{size } KN \times KN).$$

The interlaced notation used for $\boldsymbol{v}(n)$ or $\boldsymbol{w}(n)$ is not the only possible notation, but it is required in order to make $\boldsymbol{v}(n)$ a "block time series", for which it is possible to develop multichannel fast algorithms for ANC systems such as fast recursive least-squares algorithms [7],[9], or fast affine projection algorithms [12]. Using the above notation, the MFXGS-FAP for multichannel active noise control can be described by equations (1)-(10) below. Equation (2) is the equation generating the actual actuator output, it includes both the FIR filtering with adaptive filter $\hat{\boldsymbol{w}}_{i,j}(n)$ and some compensation term $\boldsymbol{r}_j^T(n)\overline{\mathfrak{n}}^T(n-1)$, which is needed because as previously mentioned fast affine projection algorithms compute auxiliary coefficients instead of the "normal" time domain coefficients. The auxiliary coefficients could be converted to "normal" time domain coefficients, but at the cost of unnecessary computations. Equation (1) is needed for the computation of a part of that compensation term. Equation (3) performs the filtering by the plant model which is found in all algorithms based on the "filtered-x" structure. Equation (4) computes estimates $\hat{\boldsymbol{d}}(n)$ of the disturbance acoustic signals to be canceled, which allows to fully inverse the order of the plant model and the adaptive filter in Fig. 1. As a consequence, the effect of the plant delay on the adaptive algorithm is removed [7],[8]. Also related to this and to Fig. 1 is (7), where alternative error signals are computed from the estimated disturbance signals $\hat{\boldsymbol{d}}(n)$ and from the adaptive filter coefficients $\hat{\boldsymbol{w}}(n)$. As in (2), a compensation term $\boldsymbol{r}^T(n)\overline{\mathfrak{n}}^T(n-1)$ is added in (7), because of the fact that the $\hat{\boldsymbol{w}}(n)$ are auxiliary coefficients and not the "normal" time coefficients. The decorrelation of the error signals as in all affine projection algorithms is performed in (8), while the update of the adaptive filter coefficients is actually done in (10). Equations (5) and (6) are used to update the $\boldsymbol{R}(n)$ matrix, for which the inverse matrix $\boldsymbol{P}(n)$ needs to be computed.

$$\boldsymbol{r}_j(n) = \boldsymbol{r}_j(n-1) + \tilde{\boldsymbol{a}}(n)\boldsymbol{x}''_j(n) - \tilde{\boldsymbol{a}}(n-L)\boldsymbol{x}''_j(n-L) \qquad (1)$$

(size: $K(N-1)\times 1 = (K(N-1)\times 1) + (K(N-1)\times IJ)(IJ\times 1) - (K(N-1)\times IJ)(IJ\times 1)$)

$$y_j(n) = \sum_{i=1}^{I} \hat{\boldsymbol{w}}_{i,j}^T(n)\boldsymbol{x}_i(n) - \boldsymbol{r}_j^T(n)\bar{\mathfrak{n}}^T(n-1) \quad (2)$$

(size: $1\times 1 = (1\times L)(L\times 1) - (1\times K(N-1))(K(N-1)\times 1)$)

$$v_{i,j,k}(n) = \boldsymbol{h}_{j,k}^T \boldsymbol{x}_i'(n) \quad (3)$$

(size $1\times 1 = (1\times M)(M\times 1)$)

$$\hat{d}_k(n) = e_k(n) - \sum_{j=1}^{J} \boldsymbol{h}_{j,k}^T \boldsymbol{y}_j(n) \quad (4)$$

(size: $1\times 1 = (1\times 1) + (1\times M)(M\times 1)$)

$$\boldsymbol{r}(n) = \boldsymbol{r}(n-1) + \tilde{\boldsymbol{a}}(n)v(n) - \tilde{\boldsymbol{a}}(n-L)v(n-L) \tag{5}$$

(size: $K(N-1)\times K = (K(N-1)\times K) + (K(N-1)\times IJ)(IJ\times K) - (K(N-1)\times IJ)(IJ\times K)$)

$$\bar{\boldsymbol{r}}(n) = \bar{\boldsymbol{r}}(n-1) + v^T(n)v(n) - v^T(n-L)v(n-L) \tag{6}$$

(size : $K\times K = K\times K + (K\times IJ)(IJ\times K) - (K\times IJ)(IJ\times K)$)

$$\hat{\boldsymbol{e}}^T(n) = \hat{\boldsymbol{d}}^T(n) + \boldsymbol{v}^T(n)\hat{\boldsymbol{w}}(n) - \boldsymbol{r}^T(n)\bar{\mathfrak{n}}^T(n-1) \tag{7}$$

(size: $K\times 1 = K\times 1 + (K\times IJL)(IJL\times 1) - (K\times K(N-1))(K(N-1)\times 1)$)

$$\varepsilon^T(n) = \boldsymbol{P}(n)\hat{\boldsymbol{e}}^T(n) \tag{8}$$

(size: $KN\times 1 = (KN\times K)(K\times 1)$)

$$\mathfrak{n}(n) = u\,\varepsilon(n) + \left[\boldsymbol{0} \quad \bar{\mathfrak{n}}(n-1)\right] \tag{9}$$

(size: $1\times KN = (1\times KN) + \left[(1\times K),(1\times K(N-1))\right]$)

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) - \boldsymbol{v}(n-N+1)\mathfrak{n}_{N-1}^T(n) \tag{10}$$

(size: $IJL\times 1 = (IJL\times 1) - (IJL\times K)(K\times 1)$)

In (9), μ is a normalized convergence gain $0 \le \mu \le 1$, typically set to 1. To get $\boldsymbol{P}(n)$ needed in (8), the equation $\boldsymbol{R}(n)\boldsymbol{P}(n) = \underline{\boldsymbol{b}}$ is solved using one single iteration of the Gauss-Seidel method [17] for each column of $\boldsymbol{P}(n)$ and $\underline{\boldsymbol{b}}$. As an example, Table 1 shows a possible Matlab™ implementation for a single iteration of the Gauss-Seidel method in the MFXGS-FAP algorithm. It shows that in each single iteration of the Gauss-Seidel method, the previous value of $\boldsymbol{P}(n)$ is used as initial conditions, and also the previous state of some internal variables is also used as initial conditions for those variables. For a proper initialization of the MFXGS-FAP algorithm (as for the MFXFAP-RLS algorithm), at the first iteration of the algorithm $v(n)$ should be non-zero and $v(n-1)\ldots v(n-L+1)$ (or $v(n-1)\ldots v(n-N+1)$) should all be zero. The regularization factor δ for the initialization of the matrix $\boldsymbol{R}(n)$ should also be tuned carefully. A value of δ larger than necessary will lead to sub-

optimal convergence performance, while a value of δ too small will lead to increased numerical noise and weaker convergence performance (even if no divergence is observed).

It is possible to have two versions if the MFXGS-FAP algorithm. The first one is called the "scalar version" and it uses the standard equations for one iteration of the Gauss-Seidel algorithm. The second version makes use of the fact that $\boldsymbol{R}(n)$ and $\underline{\boldsymbol{b}}$ have an inner structure made of blocks of size K by K, thus by replacing each scalar operation in the Gauss-Seidel algorithm with a matrix operation of size K by K, a "block version" of the MFXGS-FAP algorithm is obtained. As it will be shown in Section 3, for small values of K the complexity of both versions is about the same. The block version has the potential for better performance, because the convergence of the Gauss-Seidel iterative scheme depends on the dominance of the main diagonal elements of $\boldsymbol{R}(n)$. In the case of the block version, this condition becomes the dominance of the main block diagonal. Since the main block diagonal of $\boldsymbol{R}(n)$ is always more dominant than only the main diagonal of $\boldsymbol{R}(n)$, the convergence of the Gauss-Seidel scheme should be better in the block version. However, the proposed MFXGS-FAP algorithm uses only one single iteration of the typically iterative Gauss-Seidel algorithm, since there is experimental evidence that this is enough for the MFXGS-FAP algorithm to achieve the same performance as the theoretically more exact MFXFAP-RLS, as it will be shown in Section 4. To conclude, the best convergence of the block version over the scalar version for this single iteration of the Gauss-Seidel algorithm means that a better inverse $\boldsymbol{P}(n)$ is theoretically found by the block version. But simulation results in Section 4 will show that the two versions actually produce nearly identical results.

The MFXGS-FAP computes $\boldsymbol{P}(n)$ directly from the correlation matrix $\boldsymbol{R}(n)$ for each iteration, unlike the MFXFAP-RLS [12] or purely RLS-based algorithms [7],[9] which compute recurrently an inverse of $\boldsymbol{R}(n)$ (although $\boldsymbol{R}(n)$ is a different matrix in the case of the purely RLS-based

algorithms). This has two implications. First, the MFXGS-FAP has the potential for an inherently better numerical stability because it avoids the numerically sensitive process of inverting recursively $\boldsymbol{R}(n)$, and this will be addressed in the simulations of Section 4. Second, in the MFXGS-FAP algorithm it may not be required to invert $\boldsymbol{R}(n)$ for each iteration of the algorithm. Of course $\boldsymbol{R}(n)$ has to be always updated but its inverse does not necessarily have to be computed for each iteration. This can not be done in the MFXFAP-RLS or purely RLS-based algorithms because the recurrent scheme of inverting $\boldsymbol{R}(n)$ can not miss any update without having undesirable effects caused by discontinuities. Therefore, there is a potential for further reduction of the computational load of the MFXGS-FAP algorithm, and this will be addressed in Section 3 and also in the simulations of Section 4.

Although this has not been observed in our simulations of Section 4, it is possible on the long run in a practical implementation that the correlation values computed in equations (1), (5) and (6) become increasingly numerically noisy, because of the rectangular sliding window form of these equations. To address this potential problem, these equations could either be implemented in a moving average form (with "new average" = k $\times$ "old average" + (1-k) $\times$ "new data", where k is a "forgetting factor" between 0.9 and 1.0), or some periodic re-initialization of the algorithm could be considered.

3. COMPUTATIONAL COMPLEXITY OF THE ALGORITHMS

The computational complexity of the MFXGS-FAP algorithm introduced in Section 2 is estimated by the number of multiplies for one iteration of the algorithm. The breakdown of the estimated complexity of the MFXGS-FAP is shown is Table 2, for each equation of the algorithm. The total estimated complexity is given by:

$$IJK(2L+M+2(N-1)(K+J)+K+1)+K^2(2N-1) + J(KM+K(N-1)+IL)+KN+K^3N^2 \quad (11).$$

This is the complexity of the "scalar version" of the MFXGS-FAP. For the "block version" of the MFXGS-FAP, if it is assumed that matrix inversions are performed with standard LU decompositions, which require $O\{X^3/2\}$ multiplies for square matrices of size X, then an additional $O\{K^3/2\}$ multiplies is required. This additional load can often be neglected for practical systems. If the Gauss-Seidel scheme to compute the matrix $\boldsymbol{P}(n)$ from $\boldsymbol{R}(n)\boldsymbol{P}(n)=\underline{\boldsymbol{b}}$ is performed at a reduced rate $1/p$ (p is then the update period), as suggested in Section 2, then the complexity of the last component K^3N^2 in (11) is reduced by a factor p to become K^3N^2/p. Note that to achieve this reduction by a factor p in a practical system, the Gauss-Seidel iteration to obtain $\boldsymbol{P}(n)$ has to be continuously computed at a reduced rate, compared to the other equations of the MFXGS-FAP. For example, $\boldsymbol{P}(n)$ could be computed off-line, as opposed to an on-line implementation for the other equations of the algorithm. Otherwise, if the computation of $\boldsymbol{P}(n)$ occurs at every p iteration but needs to be computed within the time allowed for a single iteration, then there would be no real computational savings (only a small power consumption saving could still be achieved).

In the first columns of Table 3, the number of multiplies required by the MFXGS-FAP is compared to other previously published adaptive FIR filtering algorithms for multichannnel active noise control or sound reproduction, based on least-mean-squares (LMS) algorithms [7], recursive least-squares algorithms [7],[9], and affine projection/fast affine projection algorithms [12]. Two cases are considered in Table 3: a monochannel system with $I=1, J=1, K=1, L=100, M=64$ and a multichannel system with $I=1, J=3, K=2, L=100, M=64$. For the affine projection and fast-affine projection algorithms, projection orders of $N=5$ and $N=10$ are considered. From Table 3,

it is clear that except for the multichannel filtered-x LMS algorithm, the proposed MFXGS-FAP algorithm has the lowest computational complexity, and even more if an update rate of $1/20$ ($p = 20$) is used. The simulations of Section 4 will provide some evaluation of the MFXGS-FAP performance with such values of p. The reduction of the complexity over the previously published fast affine projection algorithm for multichannel active noise control (MFXFAP-RLS) is also very significant, for example with $N = 10$: from 36% of reduction (monochannel p=1) to 65% of reduction (multichannel p=20), on top of the expected improved numerical stability.

4. SIMULATIONS OF THE MULTICHANNEL ANC ALGORITHMS

In order to compare the convergence of the MFXGS-FAP algorithm with other algorithms for multichannel ANC, simulations were performed using both Matlab™ and C code (making sure that identical results were obtained), with experimentally measured acoustic impulse responses. The C code implementation was chosen to allow an easy switch between double precision floating point resolution (64 bits) and single precision floating point resolution (32 bits). The acoustic impulse responses used in the simulations were obtained from measurements made at different positions inside a duct (with a diameter of about 10 cm), at a sampling rate of 3kHz which was further downsampled by a factor of 2 in the simulations. Figure 3 shows the resulting frequency responses of the different secondary paths used in the simulations. The primary paths and the paths between the acoustic source and the reference sensors (Fig. 1) also have similar shapes for the frequency responses. The impulse responses of the primary paths have longer delays than the other paths, in order for a causal solution to exist for the simulated feedforward system. White noise was used for the excitation (acoustic source), and the resulting block diagram for the simulation setup is shown in Fig. 4. It should be noted from Fig. 4 that the reference signals (input signals of the adaptive filtering algorithm) and the disturbance signals to be cancelled are not white noise signals, even though the

excitation is a white noise signal. Other excitation signals such as tonal or multi-tonal signals could also have been considered, however in this case the number of coefficients required by the adaptive controller is typically much less (2 to 4 coefficients per tone to be cancelled), and the difference in performance between the different algorithms becomes less important, although it can still be significant.

The algorithms that were implemented for a comparison with the proposed algorithm of Section 2 are the multichannel modified filtered-x LMS and RLS algorithms [7], and the MFXFAP-RLS algorithm [12]. The RLS algorithm was modified to force the symmetry of the inverse correlation matrix, since this greatly helps the numerical stability of the algorithm [9]. The simulated system had the dimensions I=1, J=3 and K=2. This is to reflect the well known principle that an additional actuator can greatly help to find a causal solution for a broadband acoustic control system or an inverse acoustic system (MINT algorithm [18]). However, a system with I=1, J=3 and K=2 is typically underdetermined, therefore the global correlation matrix to be recurrently inverted by the modified filtered-x RLS algorithm is singular, and in order to avoid instability some noise (1%) was added to the signals used to compute this inverted correlation matrix. The experimentally measured impulse responses used for the secondary paths had 64 samples each ($M = 64$), while the adaptive filters had 150 coefficients each ($L = 150$). The forgetting factor coefficient in the recursive least-squares algorithm was set to λ=0.995. The step size μ and the regularization factor δ used by some algorithms were adjusted by trial and error, and the values producing the fastest convergence speed were selected. For all the affine projection algorithms, a value of 1.0 was used for the step size μ, which corresponds to the use of FAP algorithms without relaxation [15], leading to the fastest convergence (even under noisy plant model conditions). This was also experimentally validated by trial and error. Note that it would be possible in practice to decrease μ in order to

reduce the misadjustment when a steady state solution is reached (i.e. to "fine tune" the solution found by the adaptive filtering algorithm).

For all the setups considered in the simulations, it was found that the MFXGS-FAP in its two versions (scalar and block versions) and the MFXFAP-RLS produced three nearly identical convergence curves, for double precision and single precision floating point arithmetic, for ideal plant models and noisy plant models, for monochannel and multichannel systems, for different affine projection order, etc. The convergence curves were identical up to the second digit in dB, therefore no graph comparing the convergence of those three algorithms will be shown. This verifies the assumption that only a single iteration of the traditionally iterative Gauss-Seidel scheme is required in order to compute a good estimate of the matrix $\boldsymbol{P}(n)$, since results nearly identical to an algorithm computing exactly $\boldsymbol{P}(n)$ (the MFXFAP-RLS) were obtained. The estimate of $\boldsymbol{P}(n)$ is thus good enough to provide the convergence speed gain that the affine projection scheme can offer.

Figure 5 shows the performance of the MFXGS-FAP algorithm for different affine projection orders, from $N=1$ to $N=100$. In this figure and the subsequent figures, the attenuation is defined as the ratio of the sum of the error signals power over the sum of primary field (i.e. the disturbance signals) power. As can be seen from Fig. 5, a projection order of $N=10$ can produce a significantly improved convergence performance over a projection order of $N=1$ (the $N=1$ case is a normalized stochastic gradient descent algorithm, or NLMS). It could also be argued that $N=5$ is sufficient to obtain a significantly improved performance, at a lower cost than $N=10$. Figure 6 compares the performance of the MFXGS-FAP algorithm for projection order $N=10$ with the multichannel modified filtered-x LMS and RLS algorithms. As expected, the convergence performance of the MFXGS-FAP algorithm is found between the convergence performance of the LMS-based algorithm and the RLS-based algorithm. The convergence speed gain of the MFXGS-FAP over the multichannel modified filtered-x LMS is considerable.

Some simulations were performed to evaluate the impact of reducing the update rate of $\boldsymbol{P}(n)$ in the MFXGS-FAP algorithm. It was found that for both the scalar version and the block version of the MFXGS-FAP, update periods up to p=25 could be used without having any significant effect on the convergence curves. If update periods of more than p =25 were used, then some signs of instability started to show in the convergence curves. Therefore, the value p=20 used in Table 3 for the complexity of the MFXGS-FAP algorithm is a realistic one, and the complexity of the algorithm can be significantly reduced over the case where p=1.

Another aspect which was addressed by the floating point simulations (Matlab and C) was the numerical stability of the fast affine projection algorithms. The MFXFAP-RLS was initially found to be stable in double precision floating point format and unstable after 50000 iterations of convergence in single precision format. However, as reported in [12], a simple trick to improve the numerical robustness of the MFXFAP-RLS is to force the symmetry of the two inverse correlation matrices in the built-in sliding window recursive least-squares algorithm. With this simple modification, the MFXFAP-RLS proved to be experimentally as numerically robust as the MFXGS-FAP algorithm (no divergence, even with the single precision format and high affine projection orders). Still, it is expected that because the MFXGS-FAP algorithm avoids the recurrent process of computing the inverse of the $\boldsymbol{R}(n)$ matrix, it will prove to be more stable than the MFXFAP-RLS algorithm in most numerical environments (with typically less precision than 32 bits floating point). And since the MFXGS-FAP has a lower complexity and a nearly identical convergence performance to the MFXFAP-RLS, its use should be preferred for most systems.

The last column of Table 3 compute a performance/cost ratio, obtained from the attenuation achieved by the different algorithms after 50,000 iterations (averaged over the last 5000 iterations), divided by the number of multiplies per iteration required by each algorithm. The last column of

Table 3 does not consider the numerical stability of the different algorithms. It uses the fact that except for numerical effects all the RLS-based algorithms of Table 3 produce the same convergence performance and all the AP/FAP-based algorithms of Table 3 also produce the same convergence performance. It can be shown that the proposed MFXGS-FAP provides the best performance/cost ratio, first for $N = 5$ and then for $N = 10$. Using an update rate of $p = 20$ further improves the performance/cost ratio, since it can produce the same performance at a lower cost. The next best algorithms in terms of performance/cost ratio are the MFXFAP-RLS algorithm and the multichannel filtered-x LMS algorithm. The multichannel filtered-x fast transversal filter (FTF) algorithm has the next best performance/cost ratio, however it is usually numerically unstable. All the other algorithms (RLS-based or AP-based) have performance/cost ratios 10 to 100 times weaker than the proposed MFXGS-FAP algorithms.

As in [12], simulations with noisy acoustic plant models (“h model” in Figs. 1 and 2) were also performed to compare the robustness of the different algorithms to plant models inaccuracy. It should be noted here that the term "noisy" is used to describe a limited accuracy of the plant model, and not to describe time variations of the plant. So far ideal plant models had been assumed. The noise added to the ideal plant models was added on a frequency by frequency basis, where a random complex value with a magnitude of 20 dB or 10 dB less that the original magnitude was added to each frequency in the frequency response. Figure 7 first shows the performance when plant models with a 20 dB SNR were used. In this case the performance of all algorithms was similar to the case when ideal plant models were used (Fig. 6), except for the initial convergence of the multichannel modified filtered-x RLS algorithm which becomes slower. However, when 10 dB SNR models were used, the multichannel modified filtered-x RLS required a much smaller step size u in order to converge, which greatly slowed down its convergence speed. In this case the MFXGS-FAP algorithm greatly outperformed the multichannel modified filtered-x RLS, on top of also having a much lower computational load. This is shown in Fig. 8. Since in practice it may not

always be possible to have plant models that are very accurate, the fact that the MFXGS-FAP seems more robust to plant model noise is another reason to consider this algorithm for practical implementations. Simulations were also run using synthetic transfer functions (order 6 or 8 ARMA models) instead of the experimentally measured acoustic ones, and similar convergence results were obtained for noisy plant models (i.e. strong sensibility to plant model noise for RLS-based algorithms and better performance for the MFXGS-FAP algorithm).

Considering that the MFXGS-FAP algorithm can provide a good improvement of the convergence speed over the multichannel modified filtered-x LMS algorithm, with an increase of the computational complexity acceptable for many practical systems (i.e. the best performance/cost ratio as shown in Table 3), and considering that in the case of noisy plant models fast affine projection algorithms can even outperform the more complex algorithms based on recursive least-squares, the MFXGS-FAP is therefore an attractive algorithm for practical real-time implementations. This is particularly true since the multichannel recursive least-squares algorithms for ANC listed in Table 3 either have a much higher computational load, or have serious numerical instability problems [7],[9].

5. CONCLUSION

The multichannel MFXGS-FAP algorithm was introduced as a good alternative for practical active noise control systems using FIR adaptive filtering. Two versions of the MFXGS-FAP algorithm were presented (scalar and block), and the possibility of updating the inverse correlation matrix at a reduced rate was described and tested. The performance of the MFXGS-FAP algorithm is significantly better than multichannel LMS algorithms for ANC, with an increase of the complexity which can be acceptable for many applications. The MFXGS-FAP provides the best

performance/cost ratio. In the cases where noisy plant models are used in ANC systems, the performance of fast affine projection algorithms can be even better than the more complex RLS-based algorithms.

REFERENCES

1. Elliott S. *Signal processing for active control*. Academic Press: London, 2001
2. Kuo SM, Morgan DR. Active noise control: a tutorial review. *Proc. of the IEEE* 1999; **87**(6): 943-973.
3. Bauck J, Cooper DH. Generalized transaural stereo and applications. *J. Audio Eng. Soc.* 1996; **44**(9):683-705.
4. Nelson PA, Orduna-Bustamante F, Hamada H. Inverse filter design and equalization zones in multi-channel sound reproduction. *IEEE Trans. Speech Audio Process.* 1995; **3**(3):1-8.
5. Bjarnason E. Active noise cancellation using a modified form of the filtered-x LMS algorithm. *Proc. 6th Eur. Signal Processing Conf.*, Munich, Germany, 1992; **2**:1053–1056.
6. Kim IS, Na HS, Kim KJ, Park Y. Constraint filtered-x and filtered-u least-mean-square algorithms for the active control of noise in ducts. *J. Acoust. Soc. Amer.* 1994; **95**(6):3379–3389.
7. Bouchard M, Quednau S. Multichannel recursive least-squares algorithms and fast-transversal-filter algorithms for active noise control and sound reproduction systems. *IEEE Trans. Speech Audio Processing* 2000; **8**(5):606-618.
8. Elliott SJ, Sutton TJ, Rafaely B, Johnson M. Design of feedback controllers using a feedforward approach. *Proc. ACTIVE 95*, Newport Beach (CA), USA, 1995; 863–874.
9. Bouchard M. Numerically stable fast convergence least-squares algorithms for multichannel active sound cancellation systems and sound deconvolution systems. *Signal Processing* 2002; **82**(5):721-736.
10. Widrow B, Walach E. *Adaptive inverse control*. Prentice Hall: Upper Saddle River (NJ), 1996.

11. Bouchard M, Yu F. Inverse structure for active noise control and combined active noise control/sound reproduction systems. *IEEE Trans. on Speech and Audio Process.* 2001; **9**(2):141-151.
12. Bouchard M. Multichannel Affine and Fast Affine Projection Algorithms for Active Noise Control and Acoustic Equalization Systems. *IEEE Trans. on Speech and Audio Processing* 2003; **11**(1):54-60.
13. Douglas SC. The fast affine projection algorithm for active noise control. *Proc. 29th Asilomar Conf. Sign., Syst., Comp.*, Pacific Grove (CA), USA, 1995; **2**:1245-1249.
14. Albu F, Kadlec J, Coleman N, Fagan A. The Gauss-Seidel Fast Affine Projection Algorithm. *Proc. of SIPS 2002*, San Diego (CA), USA, 2002; 109-114.
15. Gay SL, Tavathia S. The fast affine projection algorithm. *Proceedings of ICASSP 1995*, Detroit (MI), USA, 1995; **5**:3023-3026.
16. Tanaka M, Kaneda Y, Makino S, Kojima J. Fast projection algorithm and its step size control. *Proceedings of ICASSP 1995*, Detroit (MI), USA, 1995; **2**:945-948.
17. Barret R, Berry M, Chan T, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C, Van Derr Vost H. *Templates for the solutions of linear systems: Building blocks for iterative methods* 2nd ed. SIAM: Philadelphia (PA), 1994.
18. Miyoshi M, Kaneda Y. Inverse filtering of room acoustics. *IEEE Trans. Acoustics, Speech, and Signal Processing* 1988; **36**(2):145-52.

FIGURES

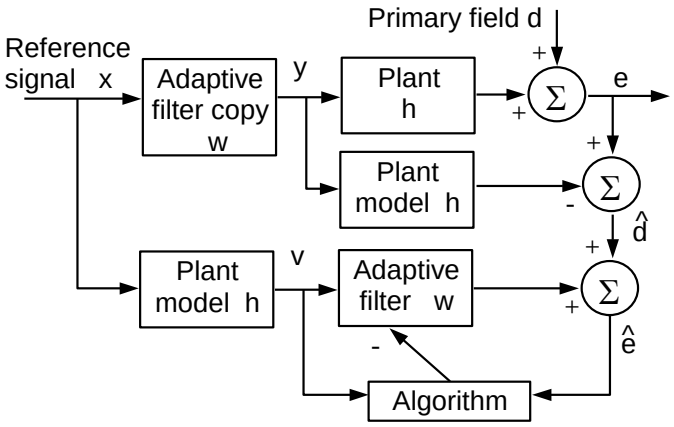

Fig. 1 (Bouchard)

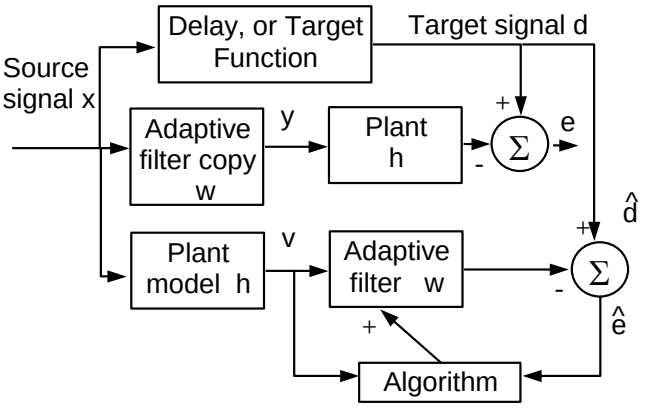

Fig. 2 (Bouchard)

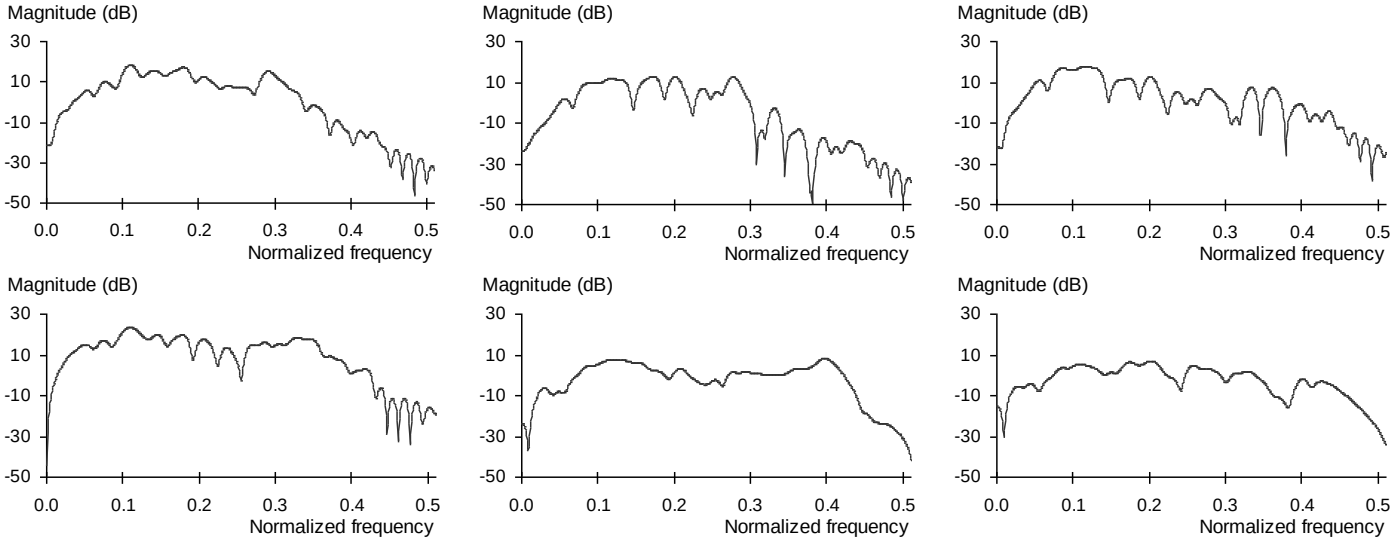

Fig. 3 (Bouchard)

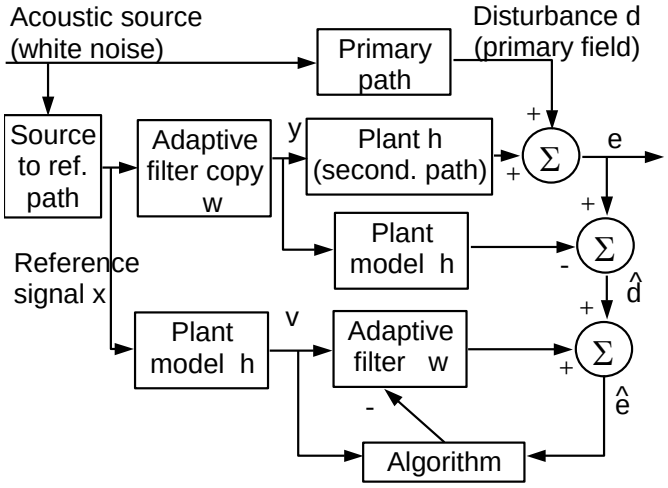

Fig. 4 (Bouchard)

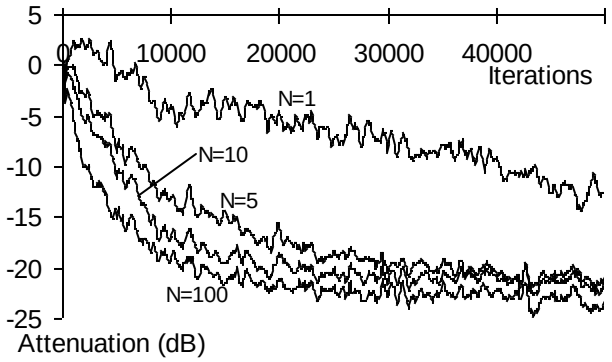

Fig. 5 (Bouchard)

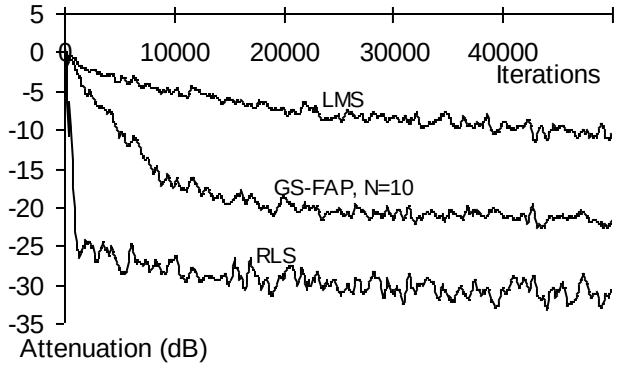

Fig. 6 (Bouchard)

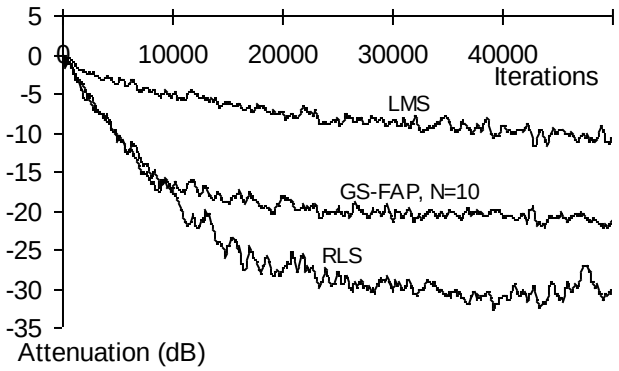

Fig. 7 (Bouchard)

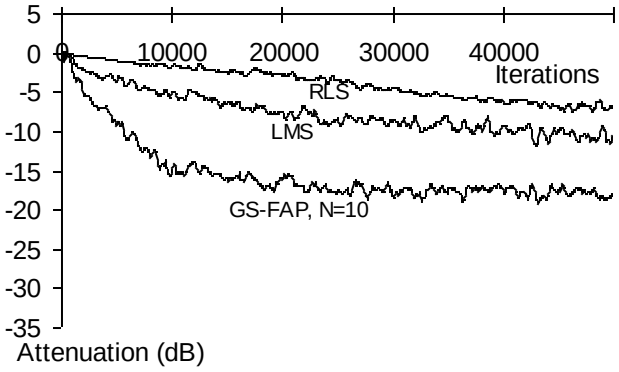

Fig. 8 (Bouchard)

FIGURE CAPTIONS

Figure 1: A delay compensated modified filtered-x structure for active noise control

Figure 2: A delay compensated modified filtered-x structure for sound reproduction systems

Figure 3: Frequency responses of the six secondary paths used in the simulations.

Figure 4: Block diagram of the simulated active noise control system.

Figure 5: Convergence curves for the MFXGS-FAP, with different affine projection orders ($N=1$, $N=5$, $N=10$ and $N=100$).

Figure 6: Convergence curves for the MFXGS-FAP algorithm and multichannel delay-compensated modified filtered-x LMS and RLS algorithms, with ideal plant models.

Figure 7: Convergence curves for the MFXGS-FAP algorithm and multichannel delay-compensated modified filtered-x LMS and RLS algorithms, with 20 dB SNR plant models.

Figure 8: Convergence curves for the MFXGS-FAP algorithm and multichannel delay-compensated modified filtered-x LMS and RLS algorithms, with 10 dB SNR plant models.

TABLES

```
inv_vector(K+1:K*N)=inv_vector(1:K*(N-1));
inv_vector(1:K)=1./diag( R(1:K,1:K) );
for k=1:1:K
   for i=1:N*K
       tmp=0;
       for j=1:N*K
          if j~=i
             tmp = tmp -R(i,j) * P(j,k);
          end
       end
        P(i,k)=(b(i,k)+ tmp)*inv_vector(i);
    end
end
```

Table 1 (Bouchard)

Equation	Number of multiplies per iteration
$\boldsymbol{r}_j(n) = \boldsymbol{r}_j(n-1) + \tilde{\boldsymbol{a}}(n)\boldsymbol{x}''_j(n) - \tilde{\boldsymbol{a}}(n-L)\boldsymbol{x}''_j(n-L)$	$2IJ^2K(N-1)$
$y_j(n) = \sum_{i=1}^{I} \hat{\boldsymbol{w}}_{i,j}^T(n)\boldsymbol{x}_i(n) - \boldsymbol{r}_j^T(n)\bar{\mathfrak{n}}^T(n-1)$	$IJL + JK(N-1)$
$v_{i,j,k}(n) = \boldsymbol{h}_{j,k}^T \boldsymbol{x}'_i(n)$	$IJKM$
$\hat{d}_k(n) = e_k(n) - \sum_{j=1}^{J} \boldsymbol{h}_{j,k}^T \boldsymbol{y}_j(n)$	JKM
$\boldsymbol{r}(n) = \boldsymbol{r}(n-1) + \tilde{\boldsymbol{a}}(n)v(n) - \tilde{\boldsymbol{a}}(n-L)v(n-L)$	$2IJK^2(N-1)$
$\bar{\boldsymbol{r}}(n) = \bar{\boldsymbol{r}}(n-1) + v^T(n)v(n) - v^T(n-L)v(n-L)$	$IJ(K^2+K)$ (symmetry)
$\hat{\boldsymbol{e}}^T(n) = \hat{\boldsymbol{d}}^T(n) + \boldsymbol{v}^T(n)\hat{\boldsymbol{w}}(n) - \boldsymbol{r}^T(n)\bar{\mathfrak{n}}^T(n-1)$	$IJKL + K^2(N-1)$
$\varepsilon^T(n) = \boldsymbol{P}(n)\hat{\boldsymbol{e}}^T(n)$	K^2N
$\mathfrak{n}(n) = u\,\varepsilon(n) + [\boldsymbol{0} \quad \bar{\mathfrak{n}}(n-1)]$	KN
$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) - \boldsymbol{v}(n-N+1)\mathfrak{n}_{N-1}^T(n)$	$IJKL$
solve for $\boldsymbol{R}(n)\boldsymbol{P}(n) = \underline{\boldsymbol{b}}$ with Gauss-Seidel	K^3N^2
Total:	$IJK(2L+M+2(N-1)(K+J)+K+1)+K^2(2N-1)$ $+J(KM+K(N-1)+IL)+KN+K^3N^2$

Table 2 (Bouchard)

Multichannel algorithm for ANC/sound reproduction (all using the modified filtered-x structure)	Multiplies per iteration for I=1, J=1, K=1, L=100, M=64	Multiplies per iteration for I=1, J=3, K=2, L=100, M=64	Performance/cost ratio (after 50000 iterations, for I=1, J=3, K=2, L=100, M=64, ideal plant models)
Proposed GS-FAP (scalar version), $N=10$	604 for p=1 509 for p=20	3,776 for p=1 3,016 for p=20	-5.7E-03 dB/multiply -7.1E-03 dB/multiply
Proposed GS-FAP (scalar version), $N=5$	489 for p=1 465 for p=20	2,796 for p=1 2,606 for p=20	-7.4E-03 dB/multiply -8.0E-03 dB/multiply
FAP-RLS, $N=10$[12]	943	8,505	-2.5E-03 dB/multiply
FAP-RLS, $N=5$ [12]	583	4,165	-5.0E-03 dB/multiply
affine projection, $N=10$[12]	12,838	137,488	-1.6E-04 dB/multiply
affine projection, $N=5$ [12]	3,821	37,678	-5.5E-04 dB/multiply
least-mean-squares (LMS) [7]	428	2,268	-4.5E-03 dB/multiply
fast-transversal-filter (FTF) [7]	1,226	16,007	-1.9E-03 dB/multiply
QRD-LSL, time-domain coefficients updated every 100 iterations [9]	4,752	54,616	-5.7E-04 dB/multiply
symmetry-preserving RLS [9]	20,729	320,122	-9.7E-05 dB/multiply
inverse QR-RLS [9]	31,628	510,019	-6.1E-05 dB/multiply

Table 3 (Bouchard)

TABLE CAPTIONS

Table 1: Matlab™ code for a single iteration of the Gauss-Seidel algorithm to compute $\boldsymbol{P}(n)$ from $\boldsymbol{R}(n)$, from $\underline{\boldsymbol{b}}$, and from an internal variable *inv_vector*, shown here for the scalar version of the MFXGS-FAP. The internal variable *inv_vector* is of size $KN \times 1$ and it is initialized with zero values.

Table 2: Complexity breakdown of the proposed MFXGS-FAP algorithm, estimated by the number of multiplies per iteration.

Table 3: Comparison of the computational load of the MFXGS-FAP algorithm with other multichannel delay-compensated modified filtered-x algorithms for ANC, and evaluation of a performance/cost ratio.

The matlab code for the proposed algorithm can be obtained from http://falbu.50webs.com/gs.html

The reference for the paper is:
M. Bouchard, F. Albu, "The Gauss-Seidel fast affine projection algorithm for multichannel active noise control and sound reproduction systems ", Special Issue on Adaptive Control of Sound and Vibration, Int. Journal of Adaptive Control and Signal Processing, vol. 19, nr. 2-3, pp. 107-123, March-April 2005