Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Fast-Array Adaptive IIR Filter For Active Noise and Vibration Control Systems

Allahyar Montazeri*, Keivan Zavari,*
Javad Poshtan*, Alireza Roosta**

**Electrical Engineering Dept., Iran University of Science and Technology,*
Narmak 16846, Tehran, Iran (e-mail: amontazeri@ iust.ac.ir).
***Shiraz University of Technology*

Abstract: In this paper an RLS-based adaptive controller for active noise and vibration control systems is presented. This adaptive controller is designed based on feed-forward architecture and uses IIR filter as the control filter. Because the derived algorithm for ANVC systems is based on RLS recursion, its computational complexity is of order $O(n^2)$ which is too high for real-time implementations. Besides, it is also vulnerable to round-off and finite precision errors that may occur in real-time implementation of the algorithm. Since the aim is to test the algorithm in an experimental duct a fast array implementation of the algorithm is proposed. This fast array form is used the extended the fast array algorithms for FIR filter which is studied in literature before. The proposed fast array solution of the algorithm not only reduces the computational complexity to the order of $O(n)$ with the same performance, but also because of its matrix nature it has good numerical stability in real-time applications which is a necessity in active noise and vibration control applications.

1. INTRODUCTION

Nowadays noise and vibration is known as one of the main sources of environmental pollution in the entire world. The use of passive methods is shown to be efficient for suppression of high frequencies noise and vibration, while a fast growing research field of active noise and vibration control (ANVC) are proved to be used effectively when the incident disturbance are the result of low frequency noise and vibration. Active noise and vibration control systems have widely been investigated in various applications in recent years and some successful results are reported in real world (Billoud, 2003, Sano *et. al*., 2002). In a general view point, active control is defined as a technique for suppressing unwanted disturbances by the introduction of controlled secondary sources such that their outputs interfere destructively with the incident primary disturbance (Elliott, 2001).

A large group of adaptive algorithms used in ANVC applications make the use of FIR filters as the controller, whose parameters adapt using stochastic gradient descent algorithms (such as LMS algorithms, Newton-LMS, …) or recursive least-squares (RLS) techniques. One of the main problems with the use of basic LMS algorithms in ANVC applications (called FxLMS algorithm) is that the convergence rate of the algorithm depends on the difference between the minimum and maximum value of auto-spectrum of the regression vector. This value is determined by auto-spectrum of the reference signal and the dynamics of the secondary path model which can cause very small convergence rate for broadband noise or vibration signals [Kuo and Morgan, 1996]. The convergence rate can be made independent to the auto-spectrum of the regression vector by using the *Newton-LMS* algorithm [Sayed 2003]. In this way the convergence rate depends just to step size and can be chosen such that fast convergence is obtained. However, since the update of filter coefficients in Newton-LMS algorithm is based on the estimates of the inverse auto-correlation of regressor and cross-correlation between the regressor and disturbance, the computational complexity of the algorithm is too high ($O(n^3)$ where n is the number of control filter weights) for most practical applications of active noise and vibration control problems. One way to reduce the computational cost of the Newton-LMS algorithm (Elliott, 2001) is to pre-compute the inverse of the auto-correlation matrix of the regression vector and use it as a fix matrix in the adaptation algorithms. Such an algorithm would not, however, be able to accommodate the significant changes in the statistical properties of the reference signal. Another way to approximate the Newton-LMS update direction, is to use affine projection algorithm (APA) (Douglas, 1995). Fast implementation of affine projection algorithm in ANC applications is proposed by (Bouchard, 2003, Bouchard and Albu, 2005). Fast affine projection algorithm can provide a good trade-off between the convergence speed and computational complexity of the Newton-LMS algorithm, however, its convergence speed is lower than that of recursive-least-squares algorithm (Diego *et. al*., 2004). Although the RLS algorithm is generally derived from a rather different perspective, the update equation of Newton-LMS algorithm has many similarities with the RLS algorithm. The use of RLS-based algorithms in ANVC applications with FIR filter structure has become more and more common (Auspitzer et. al., 1995, Bouchard

and Quednau, 2000, Bouchard, 2002). This is due to the fact that in contrary to stochastic gradient descent algorithm the convergence behaviour of RLS-type algorithms is quite independent of the statistics of the incident noise or vibration signal. The computational complexity of RLS-type algorithms are of the order $O(n^2)$, where *n* is the length of the control filter. One of the main problems in using plain RLS-type algorithms in ANVC applications is that they suffer from numerical instability due to finite precision computations. To overcome this difficulty it is shown that array-based methods RLS filtering (such as QR algorithm, inverse QR algorithm, …) will be more reliable in finite precision implementations (Sayed, 2003). In order to reduce the computational complexity of RLS-type algorithms used in ANVC applications to the order of $O(n)$ floating point operations (flops) per sampling instant fast transversal filter (FTF) is proposed in (Bouchard and Quednau, 2000) and its numerical stability is improved by using QR decompositions and lattice structures (Bouchard, 2002). However, The comparison study of (Diego *et. al.*, 2004) shows that the performance of FTF implementation of RLS algorithms used in ANVC applications is reduced in comparison with the original RLS algorithm. Array methods (Sayed, 2003) are powerful algorithmic variants that are theoretically equivalent to the recursive least-squares algorithms but they nevertheless perform the required computational in a more reliable manner. By exploiting the structure of data in the regression vector a fast array implementation of RLS algorithm for adapting the FIR filter is derived by (Sayed, 2003).

The use of IIR filters in ANVC applications date back to the development of FuLMS algorithm by (Errikson, 1987) and after that some improvements is proposed in literatures (Snyder, 1994, Mosquera *et. al.*, 1999). By using the *hyperstability* theory, an RLS-type adaptive IIR filter is proposed by the author of the paper in (Montazeri *et. al.*, 2005), in which it is shown the performance of the proposed algorithm is superior than the commonly used FuLMS and SHARF algorithms in ANVC applications. Since the computational complexity of the algorithm proposed by (Montazeri *et. al.*, 2005) is of order $O(n^2)$ and it may be vulnerable to the finite precision implementations and round-off errors, in this paper the fast RLS array implementation of the algorithm is proposed. This array algorithm is developed for IIR filters and is an extension of the algorithms proposed in literatures for FIR filters. It will be shown that the array form of the algorithm exhibits exactly the same performance as the original algorithm while its computational complexity is reduced to the order of $O(n)$. In section 2, the algorithm proposed for adaptation of IIR filter will be briefly introduced, and in section 3 the fast array form of the algorithm is derived. The simulation of the algorithm using identified model of an experimental duct is presented in section 4, and finally some conclusions will wrap up the paper.

2. ADAPTIVE IIR FILTER BY RLS-TYPE ALGORITHM

A typical adaptive feedforward ANVC system is shown in Fig. 1. Here *w(k)* is the incident noise, and *m(k),n(k)* are the measurement noises uncorrelated with *w(k)*. $G_{dw}(q)$ is the transfer function of the *primary path*, $G_{yu}(q)$ is the transfer function of *secondary path*, $G_{rw}(q)$ is the transfer function of *detecting path,* and finally $C(q)$ is the transfer function of the controller in the following form:

$$C(q,k)=\frac{b_0(k)+b_1(k)q^{-1}+b_2(k)q^{-2}+\ldots\ldots+b_{n_B}(k)q^{-n_B}}{1+a_1(k)q^{-1}+a_2(k)q^{-2}+\ldots\ldots+a_{n_A}(k)q^{-n_A}} \quad (1)$$

where n_A, n_B are the orders of denominator and numerator of the control filter respectively. For simplicity it is assumed that $G_{rw}(q)$ is equal to one. The numerator and denominator coefficients of the IIR filter (1) must be updated such that the sum of the squares of residual signal $\varepsilon(k)$ is minimized. The selected performance measure will be:

$$J(n)=\sum_{k=1}^{n}\lambda_1^{n-k}(n)\varepsilon^2(k) \quad (2)$$

$$\varepsilon(k)=d'(k)+G_{yu}(q)\left[\boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(n)\right] \quad (3)$$

where $d'(k)$ is the disturbance to be cancelled at time *k*, $G_{yu}(q)$ is the transfer function of the secondary path, $\boldsymbol{\varphi}(k)$ denote the regression vector, and $\hat{\boldsymbol{\theta}}(n)$ is the vector of coefficients of denominator and numerator of the IIR filter:

$$\boldsymbol{\varphi}(k)=\left[-u(k-1),\ldots,-u(k-n_A),r(k),\ldots,r(k-n_B)\right]^T \quad (4)$$

$$\hat{\boldsymbol{\theta}}(n)=\left[\hat{a}_1(n),\hat{a}_2(n),\ldots,\hat{a}_{n_A}(n),\hat{b}_0(n),\hat{b}_1(n),\ldots,\hat{b}_{n_B}(n)\right]^T \quad (5)$$

By adopting the assumption of slow adaptation of the coefficients of the filters, the transfer function of the secondary path can be combined with the regression vector in (3), and hence the control objective (2) may be formulized as:

$$\min_{\hat{\theta}(n)}J(n)=\min_{\hat{\theta}(n)}\sum_{k=1}^{n}\lambda_1^{n-k}(n)\left[d'(k)+\boldsymbol{\varphi}_f^T(k)\hat{\boldsymbol{\theta}}(n)\right]^2 \quad (6)$$

where $\boldsymbol{\varphi}_f(k)$ is the filtered regression vector whose elements are the filtered version of control and reference signals as follows:

$$r_f(k)=G_{yu}(q)r(k),\quad u'_f(k)=C(q,k)r_f(k) \quad (7)$$

By using this formulation, the RLS-type algorithm proposed by (Montazeri *et. al.*, 2005) for adapting the coefficients of the IIR filter (1) is as follows:

1- $r_f(k)=\hat{G}_{yu}(q)r(k), u'_f(k)=C(q,k-1)r_f(k)$

$\boldsymbol{\varphi}_f(n)=\left[-u'_f(n-1),\ldots,-u'_f(n-n_A),r_f(n),\ldots,r_f(n-n_B)\right]^T$

2- $$g(n+1)=\frac{F(n)\boldsymbol{\varphi}_f(n+1)}{\lambda_1(n)+\lambda_2(n)\boldsymbol{\varphi}_f^T(n+1)F(n)\boldsymbol{\varphi}_f(n+1)} \quad (9)$$

3- $$F(n+1)=\frac{1}{\lambda_1(n)}\left[F(n)-\frac{F(n)\boldsymbol{\varphi}_f(n+1)\boldsymbol{\varphi}_f^T(n+1)F(n)}{\frac{\lambda_1(n)}{\lambda_2(n)}+\boldsymbol{\varphi}_f^T(n+1)F(n)\boldsymbol{\varphi}_f(n+1)}\right] \quad (10)$$

4- $$e(n+1)=d'(n+1)+\boldsymbol{\varphi}_f^T(n+1)\hat{\boldsymbol{\theta}}(n) \quad (11)$$

5- $$v_0(n+1)=e(n+1)+\sum_{j=1}^{n_H}h_j\varepsilon(n+1-j) \quad (12)$$

6- $$\varepsilon(n-j)=d'(n-j)+\boldsymbol{\varphi}_f^T(n-j)\hat{\boldsymbol{\theta}}(n-j)\quad j=0,1,2,\ldots \quad (13)$$

7- $$\hat{\boldsymbol{\theta}}(n+1)=\hat{\boldsymbol{\theta}}(n)-g(n+1)v^0(n+1) \quad (14).$$

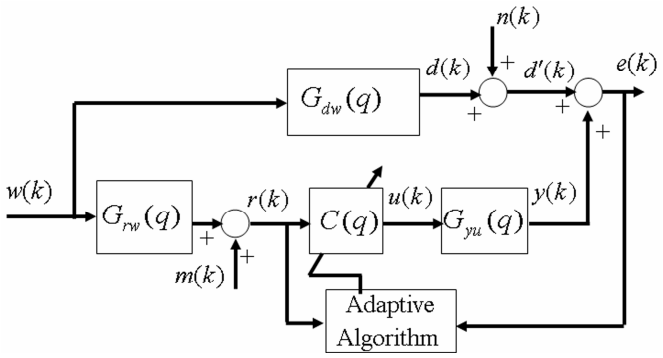

Fig. 1. Block diagram of adaptive feedforward ANVC system

In the above equations $F(n)$ in (10) is the adaptation gain matrix or the covariance of parameter estimation error, $\lambda_1(n)$ and $\lambda_2(n)$ are two weighting sequences in the range $0 << \lambda_1(n) \le 1,\ 0 < \lambda_2(n) \le 2$ determining how adaptation gain evolves in time, $e(n)$ and $\varepsilon(n)$ are *a priori* and *a posteriori* residual error sensed by error microphone, $v_0(n)$ and $v(n)$ are filtered *a priori* and *a posteriori* error used by the algorithm. In (15) $\hat{G}_{yu}(q)$ is an estimation of secondary path impulse response, and h_j's the coefficients of an FIR filters used to filter *a posteriori* error for stabilizing the adaptive algorithm by satisfying some SPR condition.

3. FAST RLS ARRAY ADAPTIVE IIR FILTER

3.1 Preliminary Results

Considering the update equation (8), it requires the gain vector $g(n)$ to compute $\hat{\boldsymbol{\theta}}(n)$. In turn, the evaluation of $g(n)$ requires the matrix $F(n)$, and updating $F(n)$ needs $O(n^2)$ operations per iteration. Besides, the computation of the denominator of $g(n)$ also requires $O(n^2)$ operation. Since these update steps are the main computational bottleneck in the algorithm, the effort is to develop a time-update for $g(n)$ directly based on $g(n-1)$. In order to implement a fast array form of the algorithm proposed in section 2, the first thing that must be noted is the structure of data in the regression vector $\boldsymbol{\varphi}_f(n)$. By splitting the regression vector into samples of the previous outputs of the filter $\boldsymbol{\varphi}_{1f}(n)$, and the samples of the reference signal $\boldsymbol{\varphi}_{2f}(n)$, the shift structure of the regressor for two successive sampling times will be captured by noting the following relation:

$$\begin{aligned}\left[-u'_f(n)\ \boldsymbol{\varphi}_{1f}^T(n)\ r_f(n+1)\ \boldsymbol{\varphi}_{2f}^T(n)\right] = \\ \left[\boldsymbol{\varphi}_{1f}^T(n+1) - u'_f(n-n_A)\ \boldsymbol{\varphi}_{2f}^T(n+1)\ r_f(n-n_B)\right]\end{aligned} \tag{16}$$

where

$$\boldsymbol{\varphi}_{1f}(n) = \left[-u'_f(n-1), \cdots\cdots, -u'_f(n-n_A)\right]^T$$

$$\boldsymbol{\varphi}_{2f}(n) = \left[r_f(n), \cdots\cdots, r_f(n-n_B)\right]^T.$$

Here it is assumed that λ_1 and λ_2 are constant and independent of time index *n*.

By defining:

$$\gamma(n+1) = \frac{\lambda_1^{-1}}{1 + \frac{\lambda_2}{\lambda_1}\boldsymbol{\varphi}_f^T(n+1)F(n)\boldsymbol{\varphi}_f(n+1)} \tag{17}$$

$$g(n+1) = \frac{\lambda_1^{-1}\ F(n)\boldsymbol{\varphi}_f(n+1)}{1 + \frac{\lambda_2}{\lambda_1}\boldsymbol{\varphi}_f^T(n+1)F(n)\boldsymbol{\varphi}_f(n+1)} \tag{18}$$

$$g'(n+1) = \frac{\lambda_2}{\lambda_1}\ g(n+1) \tag{19}$$

$$\gamma'(n+1) = \lambda_1\ \gamma(n+1) \tag{20}$$

we can write:

$$g'(n+1)\gamma'^{-1}(n+1) = \frac{\lambda_2}{\lambda_1^2}F(n)\boldsymbol{\varphi}_f(n+1) \tag{21}.$$

and

$$\gamma'^{-1}(n+1) = 1 + \frac{\lambda_2}{\lambda_1}\boldsymbol{\varphi}_f^T(n+1)F(n)\boldsymbol{\varphi}_f(n+1) \tag{22}.$$

By partitioning the adaptation gain matrix $F(n)$ as follows:

$$F(n) = \begin{bmatrix} F_{11}(n)_{n_A \times n_A} & F_{12}(n)_{n_A \times (n_B+1)} \\ F_{21}(n)_{(n_B+1)\times n_A} & F_{22}(n)_{(n_B+1)\times(n_B+1)} \end{bmatrix} \tag{23}$$

The time-update equation for the left-hand side of (21) will be rewritten as follows (the derivation is omitted here for the lack of space):

$$\begin{bmatrix} g'_1(n+1)\gamma'^{-1}(n+1) \\ 0 \\ g'_2(n+1)\gamma'^{-1}(n+1) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ g'_1(n)\gamma'^{-1}(n) \\ 0 \\ g'_2(n)\gamma'^{-1}(n) \end{bmatrix} + \lambda_1^{-1}\delta F(n) \begin{bmatrix} -u'_f(n) \\ \boldsymbol{\varphi}_{1f}(n) \\ r_f(n+1) \\ \boldsymbol{\varphi}_{2f}(n) \end{bmatrix} \tag{21}$$

where:

$$\delta F(n) = \frac{\lambda_2}{\lambda_1}\left(\begin{bmatrix} F_{11}(n) & 0 & F_{12}(n) & 0 \\ 0 & 0 & 0 & 0 \\ F_{21}(n) & 0 & F_{22}(n) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & F_{11}(n-1) & 0 & F_{12}(n-1) \\ 0 & 0 & 0 & 0 \\ 0 & F_{21}(n-1) & 0 & F_{22}(n-1) \end{bmatrix} \right)$$

Since the vector $g(n)$ is required to update the coefficients of the IIR filter, it is required to recover $g_1(n)$ and $g_2(n)$ from the time-update (21). To this end if $\gamma'^{-1}(n+1)$ is computed from (22) and subtracted from $\gamma'^{-1}(n)$, the following time-update relation will be obtained:

$$\gamma'^{-1}(n+1)=\gamma'^{-1}(n)+\begin{bmatrix}-u'_f(n)\\ \varphi_{1f}(n)\\ r_f(n+1)\\ \varphi_{2f}(n)\end{bmatrix}^T \delta F(n)\begin{bmatrix}-u'_f(n)\\ \varphi_{1f}(n)\\ r_f(n+1)\\ \varphi_{2f}(n)\end{bmatrix} \quad (22).$$

Equations (21) and (22) show that the update of the gain vector $g'(n)\gamma'^{-1}(n)$ as well as $\gamma'^{-1}(n)$ depends only to the value of $\delta F(n)$, and hence for the fast implementation of the algorithm it is required to compute the time-update of $\delta F(n)$ with the order of $O(n)$ operations. For this purpose we start by the initialization of the algorithm with the following parameters:

$$g'(0)=g(0)=0,\gamma(0)=\lambda_1^{-1},\gamma'(0)=1,F(0)=\Pi^{-1}=\begin{bmatrix}\Pi_{11}^{-1} & \Pi_{12}^{-1}\\ \Pi_{21}^{-1} & \Pi_{22}^{-1}\end{bmatrix},$$

$$F(-1)=\lambda_1 F(0)$$

$$\Pi_{11}^{-1}=\eta\begin{bmatrix}\lambda_1^2 & \cdots & 0\\ \vdots & \ddots & 0\\ 0 & 0 & \lambda_1^{n_A+1}\end{bmatrix},\Pi_{22}^{-1}=\eta\begin{bmatrix}\lambda_1^{n_A+2} & \cdots & 0\\ \vdots & \ddots & 0\\ 0 & 0 & \lambda_1^{n_A+n_B+2}\end{bmatrix} \quad (23)$$

$$\Pi_{12}^{-1}=\Pi_{21}^{-1}=0.$$

In this case, $\delta F(0)$ will be initialized with an *M+2* by *M+ 2* ($M=n_A+n_B+1$) matrix with rank four which has two positive eigenvalues and two negative eigenvalues as follows:

$$\delta F(0)=\eta\lambda_1\lambda_2\begin{bmatrix}1 & \cdots & 0 & 0 & \cdots & 0\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ 0 & \cdots & -\lambda_1^{n_A} & 0 & \cdots & 0\\ 0 & \cdots & 0 & \lambda_1^{n_A} & \cdots & 0\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ 0 & \cdots & 0 & 0 & \cdots & -\lambda_1^{M}\end{bmatrix} \quad (24)$$

Since $\delta F(0)$ is a matrix with rank four it can be factored as:

$$\delta F(0)=\bar{L}_0 S_0 \bar{L}_0^T$$

$$\bar{L}_0=\sqrt{\eta\lambda_1\lambda_2}\begin{bmatrix}1 & 0 & 0 & 0\\ \vdots & \vdots & \vdots & \vdots\\ 0 & \lambda_1^{\frac{n_A}{2}} & 0 & 0\\ 0 & 0 & \lambda_1^{\frac{n_A}{2}} & 0\\ \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & \lambda_1^{\frac{M}{2}}\end{bmatrix},S_0=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix} \quad (25)$$

where $\bar{L}_0$ is a matrix with the size of $(M+2)\times 4$ and S_0 is the signature matrix. It can be proved that (the proof is omitted here for the lack of space) if $\delta F(n)$ is factorized as $\delta F(n)=\lambda_1\bar{L}_n S_n \bar{L}_n^T$ ($S_n,\bar{L}_n$ has some known property like $S_0,\bar{L}_0$), and $F(n)$ is updated according to (10), then $\delta F(n+1)$ can also be factored as $\delta F(n+1)=\lambda_1\bar{L}_{n+1}S_{n+1}\bar{L}_{n+1}^T$ with $S_{n+1}=S_n$.

3.2 Farst Array Algorithm

By using the initialization (24) and (25), and the statement mentioned above, $\delta F(n)$ in (21) and (22) can be replaced by its factored form, and hence the following equations will be obtained:

$$\begin{bmatrix}g'_1(n+1)\gamma'^{-1}(n+1)\\ 0\\ g'_2(n+1)\gamma'^{-1}(n+1)\\ 0\end{bmatrix}=\begin{bmatrix}0\\ g'_1(n)\gamma'^{-1}(n)\\ 0\\ g'_2(n)\gamma'^{-1}(n)\end{bmatrix}+\lambda_1^{-1}\bar{L}_n S_n\bar{L}_n^T\begin{bmatrix}-u'_f(n)\\ \varphi_{1f}(n)\\ r_f(n+1)\\ \varphi_{2f}(n)\end{bmatrix}$$

$$\gamma'^{-1}(n+1)=\gamma'^{-1}(n)+\begin{bmatrix}-u'_f(n)\\ \varphi_{1f}(n)\\ r_f(n+1)\\ \varphi_{2f}(n)\end{bmatrix}^T\bar{L}_n S_n\bar{L}_n^T\begin{bmatrix}-u'_f(n)\\ \varphi_{1f}(n)\\ r_f(n+1)\\ \varphi_{2f}(n)\end{bmatrix}.$$

A close inspection of the above equations reveals that they can be written in the following norm preserving and inner-product matrix form (Sayed, 2003):

$$\begin{aligned}CC^T&=AA^T+BSB^T\\ FC^T&=DA^T+ESB^T\end{aligned} \quad (26)$$

with: $A=\gamma'^{-\frac{1}{2}}(n),\ C=\gamma'^{-\frac{1}{2}}(n+1),\ S=S_n$

$$E=\bar{L}_n,\ B=\begin{bmatrix}-u'_f(n)\\ \varphi_{1f}(n)\\ r_f(n+1)\\ \varphi_{2f}(n)\end{bmatrix}^T\bar{L}_n$$

and

$$D=\begin{bmatrix}0\\ g'_1(n)\gamma'^{-\frac{1}{2}}(n)\\ 0\\ g'_2(n)\gamma'^{-\frac{1}{2}}(n)\end{bmatrix},\ F=\begin{bmatrix}g'_1(n+1)\gamma'^{-\frac{1}{2}}(n+1)\\ 0\\ g'_2(n+1)\gamma'^{-\frac{1}{2}}(n+1)\\ 0\end{bmatrix}.$$

By putting (26) in a more general matrix form it can be rewritten as:

$$\begin{bmatrix}C & 0\\ F & Z\end{bmatrix}\begin{bmatrix}C^T & F^T\\ 0 & Z\end{bmatrix}=\begin{bmatrix}A & B\\ D & E\end{bmatrix}\begin{bmatrix}I & 0\\ 0 & S\end{bmatrix}\begin{bmatrix}A^T & D^T\\ B^T & E^T\end{bmatrix} \quad (27)$$

Therefore, there is a $J=I\oplus S$ unitary transformation Θ such that:

$$\begin{bmatrix}A & B\\ D & E\end{bmatrix}\Theta=\begin{bmatrix}C & 0\\ F & Z\end{bmatrix} \quad (28)$$

and $\Theta\begin{bmatrix}I & 0\\ 0 & S\end{bmatrix}\Theta^T=\begin{bmatrix}I & 0\\ 0 & S\end{bmatrix}$.

It can be shown (the proof is omitted here for the sake of brevity) that $Z=\sqrt{\lambda_1^{-1}}\bar{L}_{n+1}$, and hence the array form (28)

can be used to update $g'(n)\gamma'^{-1}(n)$, $\gamma'^{-1}(n)$, and $\overline{L}_n$. The algorithm proposed in section 2 by (8)-(15) can be summarized in the array form following the steps below:

1- Initialize the algorithm with the following parameters:

$$\overline{L}_0 = \sqrt{\eta\lambda_1\lambda_2}\begin{bmatrix} 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \lambda_1^{\frac{n_A}{2}} & 0 & 0 \\ 0 & 0 & \lambda_1^{\frac{n_A}{2}} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \lambda_1^{\frac{M}{2}} \end{bmatrix}, S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad (29)$$

2- By measuring the new samples of the reference signal $r(n+1)$, update the regression vector in pre-array with these new calculated samples:

$$r_f(n+1) = \hat{G}_{yu}(q)r(n+1),\ u'_f(n) = C(q,n)r_f(n) \quad (30)$$

3- Find a J-unitary $(J = I \oplus S)$ transformation matrix Θ_n such that the first element of the first row of the post-array matrix will be positive and its last four elements are equal to zero:

$$\begin{bmatrix} \gamma'^{-\frac{1}{2}}(n) & \left[-u'_f(n)\ \boldsymbol{\varphi}_{1f}^T(n)\ r_f(n+1)\ \boldsymbol{\varphi}_{2f}^T(n)\right]\overline{L}_n \\ \begin{bmatrix} 0 \\ g'_1(n)\gamma'^{-\frac{1}{2}}(n) \\ 0 \\ g'_2(n)\gamma'^{-\frac{1}{2}}(n) \end{bmatrix} & \lambda_1^{-1}\overline{L}_n \end{bmatrix}\Theta_n =$$

$$\begin{bmatrix} \gamma'^{-\frac{1}{2}}(n+1) & \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} g'_1(n+1)\gamma'^{-\frac{1}{2}}(n+1) \\ 0 \\ g'_2(n+1)\gamma'^{-\frac{1}{2}}(n+1) \\ 0 \end{bmatrix} & \sqrt{\lambda_1^{-1}}\overline{L}_{n+1} \end{bmatrix} \quad (31)$$

4- Extract the required elements of the first column of post-array matrix to calculate $g'(n+1)$, $\gamma'(n+1)$, and $g(n+1)$, $\gamma(n+1)$ subsequently for updating the coefficients of the filters:

$$g'(n+1) = \begin{bmatrix} g'_1(n+1)\gamma'^{-\frac{1}{2}}(n+1) \\ g'_2(n+1)\gamma'^{-\frac{1}{2}}(n+1) \end{bmatrix}(\gamma'^{\frac{1}{2}}(n+1)) \quad (32)$$

$$\gamma'(n+1) = (\gamma'^{\frac{1}{2}}(n+1))(\gamma'^{\frac{1}{2}}(n+1)) \quad (33)$$

$$g(n+1) = \frac{\lambda_1}{\lambda_2}g'(n+1) \quad (34)$$

$$\gamma(n+1) = \frac{1}{\lambda_1}\gamma'(n+1) \quad (35)$$

5- Measure *a priori* residual error:

$$e(n+1) = d'(n+1) + \boldsymbol{\varphi}_f^T(n+1)\hat{\boldsymbol{\theta}}(n) \quad (36)$$

6- Calculate filtered *a priori* error used by algorithm:

$$v_0(n+1) = e(n+1) + \sum_{j=1}^{n_H} h_j \varepsilon(n+1-j) \quad (37)$$

7- Calculate *a posteriori* residual error by:

$$\varepsilon(n+1) = (1 - \frac{\gamma^{-1}(n+1) - \lambda_1}{\lambda_2}.\gamma(n+1))\, e(n+1) \quad (38)$$

8- Update the coefficient vector with:

$$\hat{\boldsymbol{\theta}}(n+1) = \hat{\boldsymbol{\theta}}(n) - g(n+1)v^0(n+1) \quad (39)$$

9- Repeat steps 2 to 9 until the algorithm converges to the optimal weights of the IIR filter.

The computational complexity of both algorithms (number of multiplications in each flops) are shown and compared in Table 1. It is assumed that the secondary path is modelled with an FIR filter of order n_s, and M is the number of coefficients of IIR control filter.

Table 1. Computational complexity of both algorithms

	Proposed algorithm	Fast array implementation
Step 1	n_s+M	-
Step 2	$2M^2$+M+2	n_s+M
Step 3	$6M^2$+M+1	25(M+3)
Step 4	M	2(M+3)
Step 5	n_H	M
Step 6	M	n_H
Step 7	M	4
Step 8	-	M
Total	$8M^2$+$6M$+n_s+n_H+3	30M+n_s+n_H+85

5. SIMULATION RESULTS

The performance of the proposed fast-array adaptive IIR algorithm in ANVC applications, is evaluated in an experimental duct setup. Fig. 2 depicts the experimental duct with primary and secondary speakers, and the filter box designed to connect the duct to the computer data acquisition card. The proposed algorithm is implemented in MATLAB/Simulink environment, and real-time window target toolbox is used to generate C++ codes requires to run the algorithm in real-time manner. The order of the numerator and denominator of the IIR filter is chosed by identification of primary and secondary path using subspace method and SLICOT software. The order obtained for numerator and denominator of IIR filter are 18 and 11 respectively. In order to stabilize the algorithm $H(q)$ is found by trial and error a polynomial of order 11 so that the SPR condition requires for the stability of the algorithm satisfied. The algorithm is started with zeros initial conditions for the weights, $\lambda_1 = 1, \eta = 10^8$ and λ_2 a very small number. The primary noise is chosed to be white noise (Fig. 3 up) and the error signal is measured at error microphone for 5 second. As can be seen the error signal is converged to zero after about 0.7 second. Fig. 4 shows the convergence behaviour of some of the filter weights to their optimal value.

Fig. 2. Acoustical experimental duct

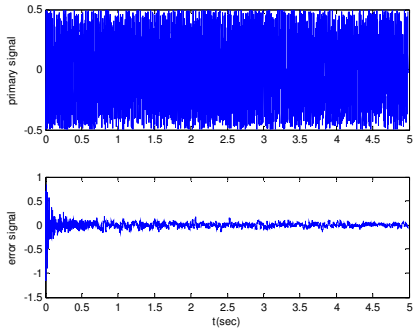

Fig. 3: Primary signal (up) and error signal (down)

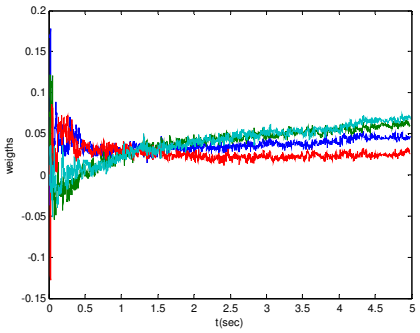

Fig. 3: Convergence behaviour of some of the filter weight

6. CONCLUSIONS

In this paper a fast-array adaptive algorithm for IIR filters in ANVC applications is derived. The algorithm is an extended form of the array algorithm used for RLS-based adaptive FIR filters. The computational complexity of the algorithm is of order $O(n)$ which is very lower than the original algorithm which is of order $O(n^2)$. The performance of proposed algorithm is shown in an experimental duct using MATLAB/Simulink and real-time windows target toolbox.

REFERENCES

Auspitzer, Th., D. Guicking and S. J. Elliott (1995). Using a fast-recursive-least-squared algorithm in a feedback-controller. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics,* pages: 61 – 64.

Billoud, G. (2003). Active control at Lord Corporation – A reality. *Technical Report,* Lord Corporation, http://www.lord.com/Portals/_default/LordDocuments/root /technical_papers/LL-6508.pdf.

Bouchard M., F. Albu (2005). The Gauss–Seidel fast affine projection algorithm for multichannel active noise control and sound reproduction systems. *Int. J. Adapt. Control Signal Process.* **vol. 19**, pages:107–123.

Bouchard M. (2003). Multichannel affine and fast affine projection algorithms for active noise control and acoustic equalization systems. *IEEE Transactions on Speech and Audio Processing,* **vol. 11, no. 1,** pages: 54–60.

Bouchard M. (2002). Numerically stable fast convergence least-squares algorithms for multichannel active sound cancellation systems and sound deconvolution systems. *Signal Processing,* **vol. 82, no. 5,** pages: 721–736.

Bouchard M. and S. Quednau (2000). Multichannel recursive-least-squares algorithms and fast-transversal-filter algorithms for active noise control and sound reproduction systems. IEEE Transactions on Speech and Audio Processing, **vol. 8, no. 5,** pages: 606-618.

.Diego M., A. Gonzalez, M. Ferrer and G. Pinero (2004). An adaptive algorithms comparison for real multichannel active noise control. *In Proceeding of European Signal Processing Conference*.

Douglas SC. (1995). The fast affine projection algorithm for active noise control. *In Proceedings of the 29th Asilomar Conference on Signals, Systems and Computers,* **vol. 2**, CA, U.S.A., pages: 1245–1249.

Elliot, S.J. (2001). *Signal processing for active control.* Academic Press.

Eriksson, L.J., M.C. Allie, R.A. Greiner (1987). "The selection and application of an IIR adaptive filter for use in active sound attenuation. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* **vol. 35, no. 4,** pages: 433-437.

Kuo, S.M, D.R. Morgan (1996). *Active noise control systems: algorithms and DSP implementations. John Wiley & Sons*.

Montazeri A., M.H. Kahaei, J. Poshtan (2005). A new stable adaptive IIR filter for active noise control systems. *In Proceeding of* 13th *European Signal Processing Conference (EUSIPCO2005),* Turkey.

Mosquera C., J.A. Gomez, F. Perez and M. Sobreira (1999). Adaptive IIR Filters for Active Noise Control. *6th International Congress on Sound and Vibration.*

Sano, H., T. Yamashita and N. Nakamura (2002). Recent application of active noise and vibration control to automobiles. *In Proc. Of Active 2002*, Southampton, UK.

Sayed, A.H. (2003). *Fundamentals of Adaptive Filtering. John Wiley & Sons*.

Snyder, S.D. (1994). Active control using IIR filters-A second look", *IEEE International Conference on Acoustics, Speech, and Signal Processing,* **vol.2,** pages: 241-244.