INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
*Int. J. Adapt. Control Signal Process.* (2015)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/acs.2574

# A numerically stable, finite memory, fast array recursive least squares filter for broadband active noise control

S. van Ophem[1,‡] and A. P. Berkhoff[1,2,*,†]

[1] *Faculty of Engineering Technology, University of Twente P.O. Box 217, 7500 AE Enschede, The Netherlands*
[2] *TNO Technical Sciences, Acoustics and Sonar, P.O. Box 96864, 2509 JG The Hague, The Netherlands*

SUMMARY

For broadband active noise control applications with a rapidly changing primary path, it is desirable to find algorithms with a rapid convergence, a fast tracking performance, and a low computational cost. Recently, a promising algorithm has been presented, called the fast-array Kalman filter, which uses rotation matrices to calculate the filter parameters. However, when this algorithm is implemented, it can show unstable behavior because of finite precision error propagation. In this paper, a novel algorithm is presented, which exhibits the fast convergence and tracking properties and the linear calculation complexity of the fast-array Kalman filter but does not suffer from the mentioned numerical problems. This is accomplished by running two finite length growing memory recursive least squares filters in parallel and using a convex combination of the two filters when the control signal is calculated. A reset of the filter parameters with proper re-initialization is enforced periodically. The mixing parameters will be chosen in such a way that the total available information used for the calculation of the control signal will be approximately equal at every time instance. The performance of the filter is shown in numerical simulations and real-time lab experiments. The numerical experiments show that the algorithm performs better numerically than the fast-array sliding window recursive least squares filter, while achieving a comparable convergence rate and tracking performance. The real-time lab experiments confirm the behavior shown in the simulations. Copyright © 2015 John Wiley & Sons, Ltd.

Received 18 July 2014; Revised 2 April 2015; Accepted 25 April 2015

KEY WORDS: active noise control; recursive least squares; primary path tracking; round-off errors

## 1. INTRODUCTION

Filtered reference least means square (fxLMS) and filtered error LMS algorithms are very popular in the context of feed-forward broadband Active Noise Control (ANC). The implementation on digital signal processors is straightforward and uses little floating point operations. However, they also have a number of less desirable properties, such as a low rate of convergence and a relatively high mean squared error (MSE) in the steady-state situation. These properties are especially problematic for multiple input and multiple output systems.

The main reason for the low convergence rate of the fxLMS algorithm is the assumption that both the adaptive filter and secondary path estimate are linear time-invariant and therefore can be interchanged [1]. This assumption only holds if the adaptive filter changes slowly in comparison with that of the secondary path dynamics. Nevertheless, to improve the convergence rate, multiple changes to the fxLMS algorithm have been proposed in the literature, for example, the modified

*Correspondence to: A. P. Berkhoff, University of Twente, Faculty of Engineering Technology, P.O. Box 217, 7500 AE Enschede, The Netherlands.
†E-mail: a.p.berkhoff@utwente.nl
‡Current address: KU Leuven, Department of Mechanical Engineering Division PMA, Celestijnenlaan 300b Box 2420, 3001 Heverlee, Belgium

fxLMS [2], fast affine projections, and preconditioned LMS [1]. Another way to improve the rate of convergence can be obtained by reformulating the ANC problem as a state estimation problem, as proposed by Sayyarrodsari *et al.* [3].

The assumption of the linear time-invariant adaptive filter and secondary path also potentially influences the tracking performance of primary path changes. These changes will occur when the primary noise source is moving relative to the ANC system or the reference microphone is moving. Some examples of moving noise sources are airplanes and cars. With such noise sources, the primary path can change rapidly, violating the assumption of a system with slowly variating dynamics. Some examples of ANC with moving noise sources are given by Omoto and Fujiwara [4], Berkhoff [5], and Van Ophem and Berkhoff [6].

The availability of affordable, high-performance digital signal processors makes it possible to implement more advanced signal processing algorithms with a potentially better performance than the fxLMS algorithms. Therefore, recent research has been focused on alternatives, such as the modified filtered-reference Recursive Least Squares (RLS) algorithm. This algorithm has a high rate of convergence, but is computationally demanding. It was shown by Fraanje *et al.* [7] that the modified filtered-RLS algorithm is a special case of a Kalman filter, which allows an efficient implementation in the form of a fast-array implementation, as earlier described by Sayed [8]. A real-time implementation of this fast-array Kalman filter was presented by Van Ophem and Berkhoff [6]. In this implementation, an output normal parameterization of the estimated secondary path was used to reduce the amount of floating point operations and to remove redundancy from the state space model.

Although the fast-array Kalman filter shows the desired high rate of convergence, it was shown by Van Ophem and Berkhoff [6] that the tracking performance is diminishing with progressing time. The reason for this behavior is that the Kalman filter uses all old data to calculate the estimate of the filter coefficients. Therefore, a logical way to improve tracking would be a mechanism that throws away old data in the recursions. In Fraanje *et al.* [7] a forgetting factor, which exponentially weights the data, was proposed to improve the tracking performance. Although the improved tracking was observed by Van Ophem and Berkhoff [6], a disastrous instability caused by the round-off errors in digital systems was also observed by the authors [6]. An alternative solution for improving the tracking performance has been proposed by Sayed [8] in the form of a sliding window. A description of this filter in fast-array form has been described by Park *et al.* [9].

The sliding window RLS algorithm works by running two RLS filters in parallel. The first filter works as a standard growing memory RLS filter, but the second filter throws away old information, which results in a finite memory filter. It was found [10] that this filter also suffers from round-off errors, especially in single precision floating point arithmetic, but with a linear error growth, as opposed to the exponential error growth with a forgetting factor. This growth was evident, even when both the orthogonal-diagonal method [11] and the update method described by Stewart and Van Dooren [12] were applied for the hyperbolic rotations. In the latter case, the growth of the error is better predictable, and therefore, by adding a third filter in parallel with the two filters already used for the sliding window algorithm, it is possible to achieve a numerically stable filter. The big disadvantage of this approach is an increase in floating point operations.

The contributions of this paper are as follows: A single input single output ANC algorithm is derived, having a rate of convergence and tracking performance similar to that of a fast-array sliding window RLS filter, but without the numerical error growth. The resulting filter also uses a similar amount of floating point operations per iteration as the fast-array sliding window RLS filter, but does not need a third parallel filter for correct numerical operation. Therefore, the amount of needed floating points operation will be less than the needed floating point operations for a fast-array sliding window RLS filter. Results of the algorithm are given in both simulations and in real-time experiments. Furthermore, the algorithm is compared with the sliding window RLS filter, both with respect to performance and calculation complexity.



DOI: 10.1002/acs

## 2. METHODS

First, the modified filtered-RLS structure for ANC will be shown. All the simulations and real-time experiments of the different RLS implementations in this paper will use this structure. The reason that the modified filtered-RLS structure is chosen is because it does not make the assumption that the primary path dynamics are slowly changing [1]. It is worth mentioning in advance that the novel RLS filter, which will be presented in Section 2.2, is not necessarily linked to this modified filtered-RLS structure, but can be applied more generally.

### *2.1. Modified filtered-recursive least squares*

Consider single input–single output ANC with a modified structure; see Figure 1. The goal of the adaptive filter is to find a set of Finite Impulse Response (FIR) filter coefficients $\hat{w}_i \in \mathbb{R}^{n_w}$, which minimize the modified error $\epsilon_i$. This modified error will be calculated by summing the estimated disturbance $\hat{d}_i$ and the output of the adaptive filter $\tilde{y}_i$:

$$\epsilon_i = \hat{d}_i + \tilde{y}_i. \tag{1}$$

The output of the adaptive filter is calculated by multiplying the filtered reference signal $\hat{r}_i$ with the filter coefficients $\hat{w}_i$

$$\tilde{y}_i = -\hat{r}_{n_w,i}^T \hat{w}_i. \tag{2}$$

$\hat{r}_{n_w,i}$ is a vector with the last $n_w$ values of the filtered reference signal:

$$\hat{r}_{n_w,i} = \begin{bmatrix} \hat{r}_i & \hat{r}_{i-1} & \cdots & \hat{r}_{i-n_w+1} \end{bmatrix}^T. \tag{3}$$

The filtered reference signal is calculated by filtering the measured reference signal with the estimated secondary path state space model:

$$\theta_{i+1}^r = A_s \theta_i^r + B_s x_i, \tag{4}$$

$$\hat{r}_i = C_s \theta_i^r + D_s x_i, \tag{5}$$

in which $\theta_i^r$ is the internal path state and $A_s$, $B_s$, $C_s$, and $D_s$ are the estimated secondary path state matrices.

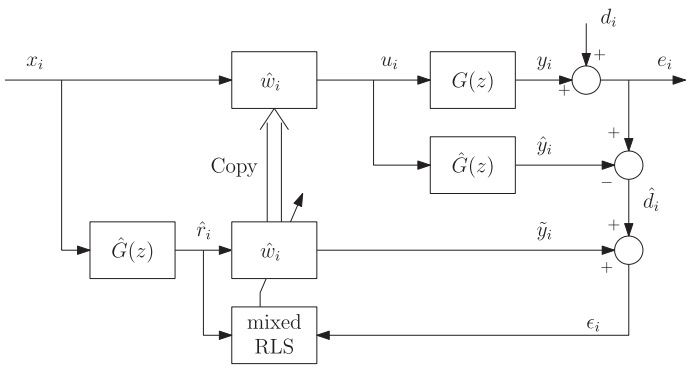


Figure 1. Modified filtered-recursive least squares.



DOI: 10.1002/acs

The estimated value of the disturbance is calculated by subtracting the estimated output $\hat{y}_i$ of the secondary path $\hat{G}(z)$ from the measured error $e_i$:

$$\hat{d}_i = e_i - \hat{y}_i. \tag{6}$$

The estimated output of the secondary path is calculated by filtering the control signal $u_i$ with the estimated state space model of the secondary path:

$$\hat{\theta}_{i+1} = A_s\hat{\theta}_i + B_s u_i, \tag{7}$$

$$\hat{y}_i = C_s\hat{\theta}_i + D_s u_i, \tag{8}$$

The control signal $u_i$ is calculated by filtering the reference signal $x_i$ with the adaptive filter, as follows

$$u_i = x_{n_w}^T \hat{w}_i, \tag{9}$$

$$x_{n_w} = \begin{bmatrix} x_i & x_{i-1} & \cdots & x_{i-n_w+1} \end{bmatrix}^T. \tag{10}$$

This filter structure is well known in the context of ANC and has been applied both to filtered-reference LMS and RLS algorithms [1, 7].

### 2.2. *Mixed windowed recursive least squares*

For the adaption of the filter coefficients, we propose a filter that behaves like a constant length finite memory RLS algorithm with a linear calculation complexity of $O(n_w)$, equivalent to the Chandrasekhar form of the sliding window RLS filter [9], but does not exhibit the round-off error propagation. To achieve this, a convex mixing approach is used to emulate the sliding window RLS filter.

Convex combinations of filters have been a popular topic in recent years; see Refs. [13–15]. An example of convex filters in the context of ANC is given by Ferrer [16]. The main difference between these approaches and the proposed filter in this paper is the way the convex combination is applied. In the literature, the convex combinations of the filters are used to mix two filters, which have different filter parameters, such as forgetting factors and convergence coefficients. The optimal mixing parameters, which give the lowest MSE, then will be determined by an extra adaptive filter.

The proposed implementation will use two filters with identical filter parameters and predetermined time-varying mixing coefficients. This means that it is not necessarily the convex combination with the lowest MSE that will be found. Instead, it simulates a filter with a constant memory length, such as the sliding window RLS filter.

The filter works as follows: Two parallel growing memory filters are mixed in such a way that the total available information used for calculating the least squares solution will be equal at every time instance. This is made under the assumption that the input signal is persistently exciting.

First, the needed equations for a recursive update of the mixed solution are derived. The mixing parameters are constrained as follows: $0 \leqslant \alpha_i \leqslant 1, 0 \leqslant \beta_i \leqslant 1$, and they sum up to one:

$$\alpha_i + \beta_i = 1, \ \forall i. \tag{11}$$

Consider two parallel RLS filters, both with a growing data window bounded to $W$ entries. The first filter will be activated at time instance $U$, and the second filter will be activated after $V = U + W/2$ iterations. The filters have the following data matrices, $H_i$, $H_{V:i}$, and measurement vectors $y_i$, $y_{V:i}$, with $V < i < W$:



DOI: 10.1002/acs

$$H_i = \begin{bmatrix} \hat{r}^T_{n_w,U} \\ \hat{r}^T_{n_w,U+1} \\ \vdots \\ \hat{r}^T_{n_w,i} \end{bmatrix}, \; H_{V:i} = \begin{bmatrix} \hat{r}^T_{n_w,V} \\ \hat{r}^T_{n_w,V+1} \\ \vdots \\ \hat{r}^T_{n_w,i} \end{bmatrix}. \tag{12}$$

$$y_i = \begin{bmatrix} \hat{d}_U \\ \hat{d}_{U+1} \\ \vdots \\ \hat{d}_i \end{bmatrix}, \; y_{V:i} = \begin{bmatrix} \hat{d}_V \\ \hat{d}_{V+1} \\ \vdots \\ \hat{d}_i \end{bmatrix}. \tag{13}$$

The cost functions of the parallel RLS filters are [8]:

$$\min_{w_i} \left[ w_i^T \Pi w_i + \| y_i - H_i w_i \|^2 \right], \tag{14}$$

$$\min_{w_{V:i}} \left[ w_{V:i}^T \Pi w_{V:i} + \| y_{V:i} - H_{V:i} w_{V:i} \|^2 \right], \tag{15}$$

in which the matrix $\Pi \in \mathbb{R}^{n_w \times n_w}$ is a positive definite regularization matrix. Minimization of Equations (14) and (15), with respect to $w_i$ and $w_{V:i}$, respectively shows that

$$w_i = \left( \Pi + H_i^T H_i \right)^{-1} H_i^T y_i, \tag{16}$$

$$w_{V:i} = \left( \Pi + H_{V:i}^T H_{V:i} \right)^{-1} H_{V:i}^T y_{V:i}. \tag{17}$$

We assume that the solution of the RLS problem is a convex combination of the two parallel filters:

$$\hat{w}_{mix,i} = \alpha_i \left( \Pi + H_i^T H_i \right)^{-1} H_i^T y_i + \beta_i \left( \Pi + H_{V:i}^T H_{V:i} \right)^{-1} H_{V:i}^T y_{V:i}. \tag{18}$$

The resulting update equations are:

$$\hat{w}_{mix,i} = \alpha_i \left( \hat{w}_{i-1} + K_i R_i^{-1} \epsilon_i \right) + \beta_i \left( \hat{w}_{V:i-1} + K_{V:i} R_{V:i}^{-1} \epsilon_{V:i} \right). \tag{19}$$

In the Appendix, a derivation of Equation (19) is given. To solve Equation (19), the weighting factors $\alpha_i$ and $\beta_i$ have to be known. In Section 2.3, a particular choice for these factors will be given. Besides these weighting factors, a way to also calculate the variables $K_i$, $K_{V:i}$ $R_i$, $R_{V:i}$, $\epsilon_i$, and $\epsilon_{V:i}$ has to be specified. In Section 2.4 an algorithm with $O(n_w)$ complexity will be presented.

### *2.3. Convex mixing values*

Again consider the mixing problem at time instance $i$. If we assume that the input signal is persistently exciting, the amount of data present in both of the data matrices $H_i$, $H_{V:i}$ will grow when the data window grows. We will name the filter, which uses $H_i$ to calculate the solution, *filter 1* and the filter which uses $H_{V:i}$, *filter 2*. Consider the following points of interest:

The first point of interest is at exactly $i = V$. In this case $H_{V:i}$ contains no information at all. Therefore, it makes sense to set the weighting factor of *filter 1* to $\alpha_i = 1$ and the weighting factor of *filter 2* to $\beta_i = 0$. The second point of interest is when $i$ is somewhere in between $V$ and $W$. In this case, both the filters have information present in their data matrices. The amount of data the filters have is proportional to the window length, so to compensate for this, the weight of the



DOI: 10.1002/acs

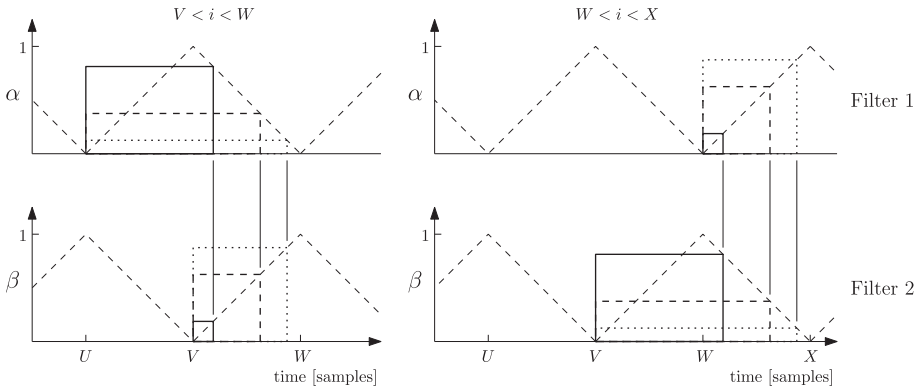


Figure 2. Weighting of the data windows as function of time.

least squares solution of *filter 1* will be decreased when $i$ grows larger and the weight of the solution of *filter 2* will be increased, until the situation $i = W$ is reached and $\alpha_i = 0$ and $\beta_i = 1$. A schematic drawing of this mixing scheme is shown in Figure 2. This figure shows the mixing parameters and the data windows at different time instances. The data windows have been scaled with $\alpha_i$ and $\beta_i$.

At time instance $i = W$, the first filter will be reset and the whole weighting scheme will repeat, but then the other way round, so *filter 2* uses the data matrix $H_{V:i}$ and *filter 1* will use $H_{W:i}$. This is shown at the right hand side in Figure 2. It is also possible to use more filters in parallel. The advantage of more parallel filters is that a more smoothly changing weighting scheme can be used. However, the disadvantage is that the amount of floating point operations will increase.

### 2.4. *Fast-array formulation*

The parameters in Equation (19) will be updated with two parallel growing memory (forgetting factor $\lambda = 1$) fast-array RLS algorithms. The complexity of these algorithms grows linearly with $n_w$. This linear complexity is achieved by updating the difference of the state error covariance matrices $P_i$ between time instances $i - 1$ and $i$. It assumes that this difference can be factorized as follows [8]:

$$dP_i = P_i - \Psi P_{i-1} \Psi^T = L_i M_i L_i^T. \tag{20}$$

In this equation, $\Psi$ is a first diagonal shift matrix. For a system with a shift-invariant input signal, like the proposed ANC system, the rank of the matrix $M$ can be as low as $\text{rank}(M) = 2$. Because an extensive derivation for the update equations of a fast-array RLS algorithm is available in the literature (see Refs. [7, 8]) we will simply state the results from the literature and the resulting filter equations. The filter updates will be given in terms of Kalman variables, in which $K_i$ is the Kalman gain, $\epsilon_i$ is the innovation, and $R_i$ is the expected value of the innovation. This is performed because in this way, the equations stated in this paper conform to the literature on this subject and also to express the possibility of extending these equations to a more general Kalman filtering problem, where uncertainties exist in the secondary path state.

The filter parameters of the two filters will be calculated completely in parallel: No interaction will take place between the filters. Both filters will be reset every time a window length $W$ has passed. The full filter, including the modified RLS structure, is shown in Table I. It can be seen that the convex combination takes place when the estimation of the secondary path states $\hat{\theta}_i$ has to be performed. The mixing parameters $\alpha_i$ and $\beta_i$ can be implemented with a Bartlett window.

The rotation matrices $\Theta_{i-1}^{(1)}$ and $\Theta_{i-1}^{(2)}$ can be calculated with Givens rotations or hyperbolic rotations. Details of the implementation of these matrices can be found in Sayed [8].



DOI: 10.1002/acs

Table I. Update procedure for the parallel recursive least squares filters.

Global variables

Initialization:

$\theta_0^r = \hat{\theta}_0 = \mathbf{0}_{n_s \times 1}$

$x_{n_w,-1} = \hat{r}_{n_w,-1} = \mathbf{0}_{n_w \times 1}$

Iterate for $i \geqslant 0$:

$\theta_{i+1}^r = A_s \theta_i^r + B_s x_i$

$\hat{r}_i = C_s \theta_i^r + D_s x_i$

$\hat{r}_{n_w,i} = \begin{bmatrix} \hat{r}_i & \hat{r}_{n_w-1,i-1}^T \end{bmatrix}^T$

$x_{n_w,i} = \begin{bmatrix} x_i & x_{n_w-1,i-1}^T \end{bmatrix}^T$

if $i \geqslant W/4$ :

$\alpha_i = \text{Bartlett}[(i + W/4) \bmod W]$

$\beta_i = \text{Bartlett}[(i + 3W/4) \bmod W]$

$\hat{\theta}_{i+1} = A_s \hat{\theta}_i + B_s \left( \alpha_i \hat{r}_{n_w,i}^T \hat{w}_i^{(1)} + \beta_i \hat{r}_{n_w,i}^T \hat{w}_i^{(2)} \right)$

$\hat{y}_i = C_s \hat{\theta}_i + D_s \left( \alpha_i \hat{r}_{n_w,i}^T \hat{w}_i^{(1)} + \beta_i \hat{r}_{n_w,i}^T \hat{w}_i^{(2)} \right)$

else:

$\hat{\theta}_{i+1} = A_s \hat{\theta}_i + B_s \hat{r}_{n_w,i}^T \hat{w}_i^{(1)}$

$\hat{y}_i = C_s \hat{\theta}_i + D_s \hat{r}_{n_w,i}^T \hat{w}_i^{(1)}$

$\text{Bartlett}[i] = \begin{cases} \frac{2i}{W-1}, & 0 \leqslant i < W/2 \\ 1 - \frac{2i}{W-1}, & W/2 \leqslant i \leqslant W \end{cases}$

| Filter 1 | Filter 2 |
|---|---|
| Initialization ($i = 0$) or reset: | |
| $\hat{w}_i^{(1)} = \mathbf{0}_{n_w \times 1}$ or $\hat{w}_i^{(1)}$ | $\hat{w}_i^{(2)} = \mathbf{0}_{n_w \times 1}$ or $\hat{w}_i^{(2)}$ |
| $\bar{K}_{i-1}^{(1)} = \mathbf{0}_{n_w \times 1}$ | $\bar{K}_{i-1}^{(2)} = \mathbf{0}_{n_w \times 1}$ |
| $\hat{r}_{n_w,i-1}^{(1)} = \mathbf{0}_{n_w \times 1}$ | $\hat{r}_{n_w,i-1}^{(2)} = \mathbf{0}_{n_w \times 1}$ |
| $R_{i-1}^{(1)} = 1$ | $R_{i-1}^{(2)} = 1$ |
| $L_{i-1}^{(1)} = \sqrt{\delta} \begin{bmatrix} 1 & 0 \\ \mathbf{0}_{n_w-1 \times 1} & \mathbf{0}_{n_w-1 \times 1} \\ 0 & 1 \end{bmatrix}$ | $L_{i-1}^{(2)} = \sqrt{\delta} \begin{bmatrix} 1 & 0 \\ \mathbf{0}_{n_w-1 \times 1} & \mathbf{0}_{n_w-1 \times 1} \\ 0 & 1 \end{bmatrix}$ |
| Iterate for $i \geqslant 0$: | Iterate for $i \geqslant W/4$: |
| Reset if $(i + W/4) \bmod W = 0$ | Reset if $(i + W/4) \bmod W = W/2$ |
| $\tilde{y}_i^{(1)} = -\hat{r}_{n_w,i}^T \hat{w}_i^{(1)}$ | $\tilde{y}_i^{(2)} = -\hat{r}_{n_w,i}^T \hat{w}_i^{(2)}$ |
| $\epsilon_i^{(1)} = e_i - \hat{y}_i + \tilde{y}_i^{(1)}$ | $\epsilon_i^{(2)} = e_i - \hat{y}_i + \tilde{y}_i^{(2)}$ |
| $\hat{r}_{n_w+1,i}^{(1)} = \begin{bmatrix} \hat{r}_i & \hat{r}_{n_w,i-1}^{(1)^T} \end{bmatrix}^T$ | $\hat{r}_{n_w+1,i}^{(2)} = \begin{bmatrix} \hat{r}_i & \hat{r}_{n_w,i-1}^{(2)^T} \end{bmatrix}^T$ |
| Perform a J-unitary rotation that makes the 1-2 block of the post-array equal to 0: | idem |
| $J = (I_2 \oplus -1),\ \Theta_{i-1}^{(1)} J \Theta_{i-1}^{(1)^T} = J$ | $J = (I_2 \oplus -1),\ \Theta_{i-1}^{(2)} J \Theta_{i-1}^{(2)^T} = J$ |
| $\begin{bmatrix} R_{i-1}^{(1)^{1/2}} & \hat{r}_{n_w+1,i}^{(1)^T} L_{i-1}^{(1)} \\ \begin{bmatrix} 0 \\ \bar{K}_{i-1}^{(1)} \end{bmatrix} & L_{i-1}^{(1)} \end{bmatrix} \Theta_{i-1}^{(1)} = \begin{bmatrix} R_i^{(1)^{1/2}} & 0_{1\times 2} \\ \begin{bmatrix} \bar{K}_i^{(1)} \\ 0 \end{bmatrix} & L_i^{(1)} \end{bmatrix}$ | $\begin{bmatrix} R_{i-1}^{(2)^{1/2}} & \hat{r}_{n_w+1,i}^{(2)^T} L_{i-1}^{(2)} \\ \begin{bmatrix} 0 \\ \bar{K}_{i-1}^{(2)} \end{bmatrix} & L_{i-1}^{(2)} \end{bmatrix} \Theta_{i-1}^{(2)} = \begin{bmatrix} R_i^{(2)^{1/2}} & 0_{1\times 2} \\ \begin{bmatrix} \bar{K}_i^{(2)} \\ 0 \end{bmatrix} & L_i^{(2)} \end{bmatrix}$ |
| $\hat{w}_{i+1}^{(1)} = \hat{w}_i^{(1)} + \bar{K}_i^{(1)} R_i^{(1)^{-1/2}} \epsilon_i^{(1)}$ | $\hat{w}_{i+1}^{(2)} = \hat{w}_i^{(2)} + \bar{K}_i^{(2)} R_i^{(2)^{-1/2}} \epsilon_i^{(2)}$ |

### 2.5. *Calculation complexity*

The calculation complexity of the proposed algorithm was compared with that of the stabilized sliding window RLS algorithm with a third parallel filter. Detailed information about this algorithm can be found in [10]. The results of this analysis are displayed in Table II for the different filter steps. All the terms that are not dependent on $n_w$ or $n_s$ are not shown. For the update of the filter variables with the rotation matrices, which is the most costly operation, a mixed-downdating technique [8] is chosen for the calculation of the computational complexity. It can be seen that for both algorithms,



DOI: 10.1002/acs

Table II. Computational complexity of proposed algorithm and the sliding window recursive least squares filter with reset.

| | Mixed windowed RLS | | Sliding window RLS (with reset) | |
|---|---|---|---|---|
| Part | Multiplications | Additions | Multiplications | Additions |
| Filtering reference: | $n_s^2 + 2n_s$ | $n_s^2 + n_s$ | $n_s^2 + 2n_s$ | $n_s^2 + n_s$ |
| Control signal: | $n_s^2 + 2n_s + 2n_w$ | $n_s^2 + n_s + n_w$ | $n_s^2 + 2n_s$ | $n_s^2 + n_s$ |
| Output adaptive filter: | $2n_w$ | $2n_w$ | $2n_w$ | $2n_w$ |
| Innovation: | - | $2n_w$ | - | $2n_w$ |
| Updated filter variables: | $16n_w$ | $16n_w$ | $24n_w$ | $24n_w$ |
| Update filter coefficients: | $2n_w$ | $2n_w$ | $2n_w$ | $2n_w$ |
| Total | $22n_w + 2n_s^2 + 4n_s$ | $23n_w + 2n_s^2 + 2n_s$ | $28n_w + 2n_s^2 + 4n_s$ | $30n_w + 2n_s^2 + 2n_s$ |

the calculation complexity is growing linearly with $n_w$, but the proposed algorithm uses less floating point operations, which makes the proposed algorithm an interesting alternative to the fast-array sliding window RLS filter.

It is also clear that the initial filtering of the reference signal through the estimated secondary path is achieved with a similar amount of floating point operations for both cases and that its calculation complexity is growing quadratically with $n_s$. It is possible to reduce this by using an output normal form parameterization of the state space model, which makes it possible to use a more efficient algorithm for the update of the estimated secondary path states. More information about this procedure can be found in [17]. The computational complexity of the related exponential fast-array RLS filter and the fast-array Kalman filter are not shown in this paper, but can be found in [7].

## 3. RESULTS AND DISCUSSION

The performance of the new approach was tested both numerically and experimentally. First, the numerical performance was tested and compared with that of a sliding window RLS filter. The numerical experiments were carried out with both measurement data from a duct and synthetic data. For all the experiments, a sampling frequency of $f_s = 2000\ Hz$ was used.

### *3.1. Comparison with growing memory recursive least squares*

For the first simulation, the convergence and tracking of the present method were compared with that of a fast-array RLS filter, with a forgetting factor of $\lambda = 1$. Synthetic primary and secondary paths were used to calculate the reference signal, the error signal, and the resulting control signal. These paths result from a 1D acoustic model of a duct, with a white noise source. The filter contains $n_w = 250$ coefficients.

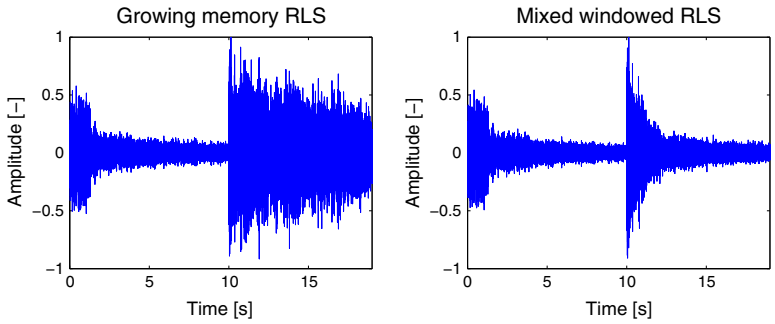


Figure 3. Convergence and tracking performance of the growing memory recursive least squares (RLS) filter (left) and the mixed windowed RLS filter of data length $W = 6000$ (right). The filter is activated after 1 s and the tracking performance is checked by shifting the reference signal after 10 s.



DOI: 10.1002/acs

The tracking behavior was tested by changing the simulation position of the reference sensor after 5 s. For the mixed windowed RLS filter, a data window of length $W = 6000$ was chosen. The results are shown in Figure 3. It is clear that the new algorithm outperforms the growing memory RLS filter when it is used for tracking purposes. However, it is more interesting to compare the results of the present approach with that of a fast-array sliding window RLS algorithm, such as described by Park *et al.* [9].

### *3.2. Comparison with fast-array sliding window recursive least squares*

For the comparison of the mixed windowed RLS with the fast-array sliding window RLS, two cases were be considered: a comparison of both the convergence and tracking properties and the long-term numerical behavior. The rotation matrix of the sliding window RLS filter has been calculated with the orthogonal diagonal method [11] because of its superior numerical behavior, when compared with that of the hyperbolic Givens rotations.

In Figure 4, the fast-array sliding window RLS and the new filter are compared. For these simulations, the data window of the new filter was set to two times the length of the fast-array sliding window RLS filter. Just as with the comparison of the new filter with the growing memory, the tracking performance was tested by changing the reference signal after 10 s, and the results were averaged over 200 simulations for different window lengths. To obtain a good comparison, the filter coefficients of the new filter were set to zero at every reset point. This was carried out to emulate the behavior of the downdate step of the sliding window algorithm.

It can be seen that the mixed window RLS filter approximates the sliding window RLS filter, but that of the MSE is not as smooth. Closer inspection shows that this variation in the MSE is related to the window length, so this is an artifact of the mixing scheme. Further tests show that the amplitude of the fluctuation depends on the magnitude of the regularization coefficient $\delta$. A high value of $\delta$ causes an overshoot when the filter converges, and because at every reset, one of the parallel has to converge again, this can cause a higher MSE. A possible solution to overcome this overshoot would be to incorporate the uncertainty in the secondary path estimates, as described by Fraanje *et al.* [7].

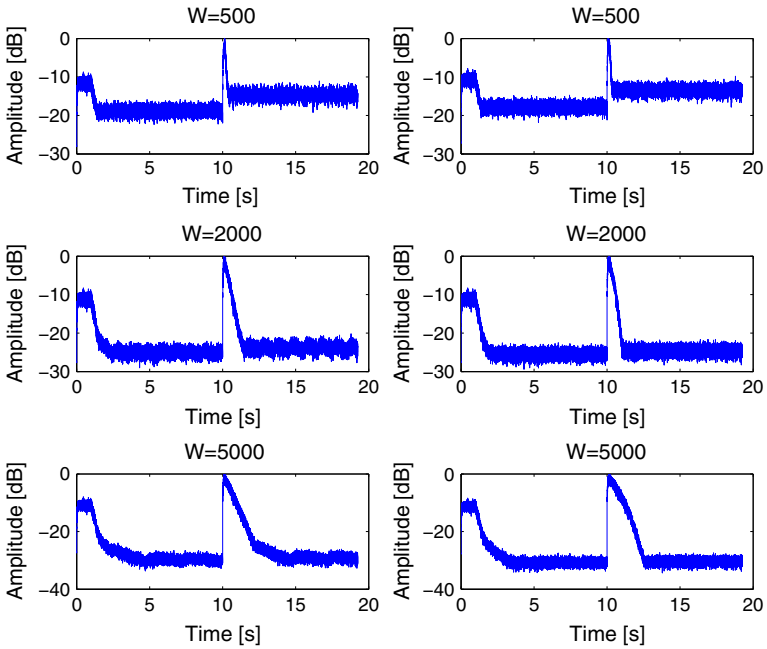


Figure 4. The convergence and tracking performance of the mixed windowed recursive least squares (RLS) filter (left) and the fast-array sliding window RLS (right) for different window lengths averaged over 200 simulations. The filter coefficients are reset to zero.



DOI: 10.1002/acs

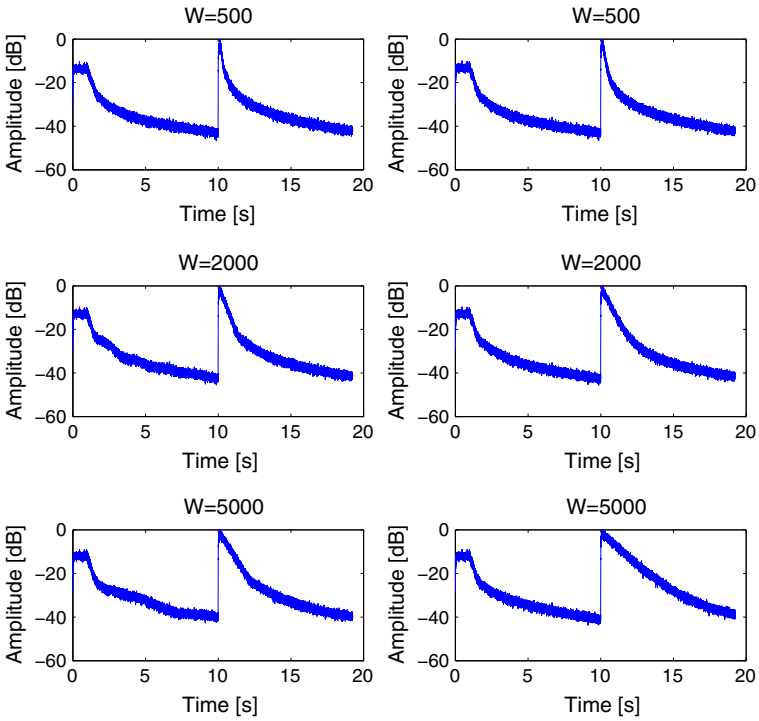


Figure 5. The convergence and tracking performance of the mixed windowed recursive least squares (RLS) filter (left) and the fast-array sliding window RLS (right) for different window lengths averaged over 200 simulations. The filter coefficients are initialized at their old values before the reset.

Because it is also possible to reset the parallel filter with the initial values set to the old filter coefficients, as indicated in Table I, the performance of the filter was compared for this situation as well. The update equations of the sliding window filter were also changed, so that only an update was performed at every time instance. The reasoning behind this change can be found in [10].

Another possibility is to set the filter coefficients to their last estimated value when a reset is applied. Simulations indicate that this can lead to a much smaller MSE, while the tracking behavior does not suffer. The results are comparable with that of a sliding window RLS filter, which does not apply a downdate step to the filter coefficients but only an update step. The results can be seen in Figure 5. It is clear that especially for large window lengths, the mixed windowed RLS only crudely approximates the sliding window filter, but it can also be seen that both filters converge to the same MSE.

### *3.3. Numerical behavior*

Even when the stable orthogonal diagonal method is used to perform the hyperbolic rotations, the fast-array sliding window RLS filter exhibits a linear error growth, which means that the performance of the algorithm will deteriorate when it runs through a large number of recursions. To keep this numerical inaccuracy within bounds, a reset of the algorithm is needed. This means that a third filter has to run in parallel with the sliding window filter when the reset is applied. The mixed windowed RLS filter does not need this extra filter.

This is shown in Figure 6 where the results of simulations with both double and single floating point precision are shown for both filters. This simulation uses time-invariant data, so it is expected that the average of the filter parameters should converge to a certain solution and must not deviate. It can be seen that the value $R_i^{(2)}$ of the sliding window fast-array RLS filter starts to deviate after about $5e5$ iterations. The weighted average value of $R_{i,new} = \alpha_i R_i^{(1)} + \beta_i R_i^{(2)}$ stays constant



DOI: 10.1002/acs

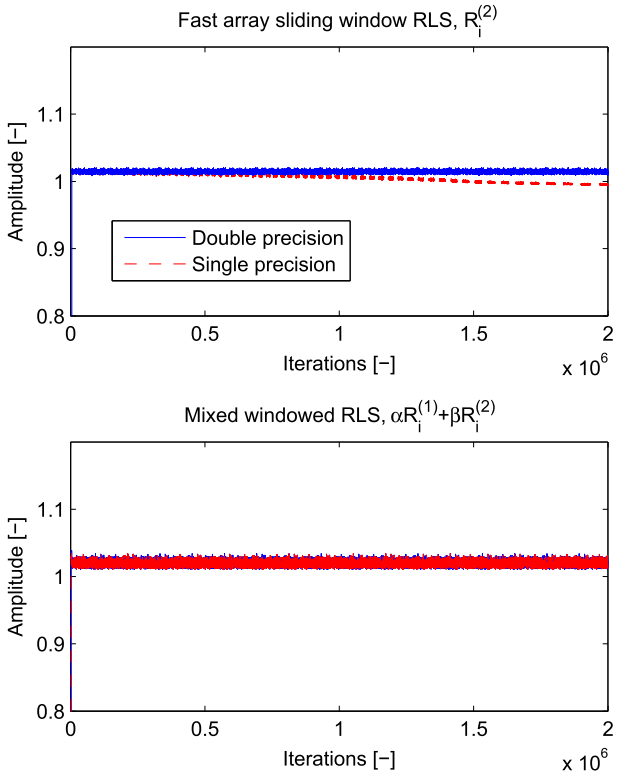


Figure 6. The numerical performance is plotted for the sliding window fast-array recursive least squares (RLS) (top) and the mixed windowed RLS filter (bottom) in single and double point floating precision.

in both single and double floating point precision. Similar comparisons of the numerical accuracy of the fast-array sliding window RLS algorithm with and without a third parallel filter have been carried out by Van Ophem and Berkhoff [10].

From the numerical simulations, it can be concluded that both the new filter and the fast-array sliding window RLS filter have their benefits, but it is has to be noted that the numerical problems of the sliding window fast-array RLS cannot be overcome without adding a third parallel filter, while the fluctuations in the MSE of the new filter are controllable by tuning the filter parameters, such as the regularization term $\delta$.

### *3.4. Experimental results*

The algorithm was also tested in a real-time environment. For the experiment, a duct was used, which was closed on the left hand side and open on the other side. A sound source, emitting white noise, was placed in the pipe on the left hand side, and the goal of the experiment was to minimize the sound pressure at the open end of the pipe by using feed forward control. This was accomplished by sending a control signal to the secondary loudspeaker, placed in duct at about 45 cm from the end of the duct. An error microphone was placed at the open end.

The signal processing was executed on a platform, running Xenomai Linux. The details of this platform are specified in [18]. The secondary path identification was carried out offline by using a sub-space identification algorithm in the SLICOT libraries. This led to estimates with a variance accounted for value of about 99.8%. A digital reference signal was used so that no feedback from the actuator to the reference signal would occur.

Both the rate of convergence and tracking performance were validated for different window lengths, and the noise reduction in dB was calculated. The tracking performance was checked by suddenly shifting the reference by three samples.



DOI: 10.1002/acs

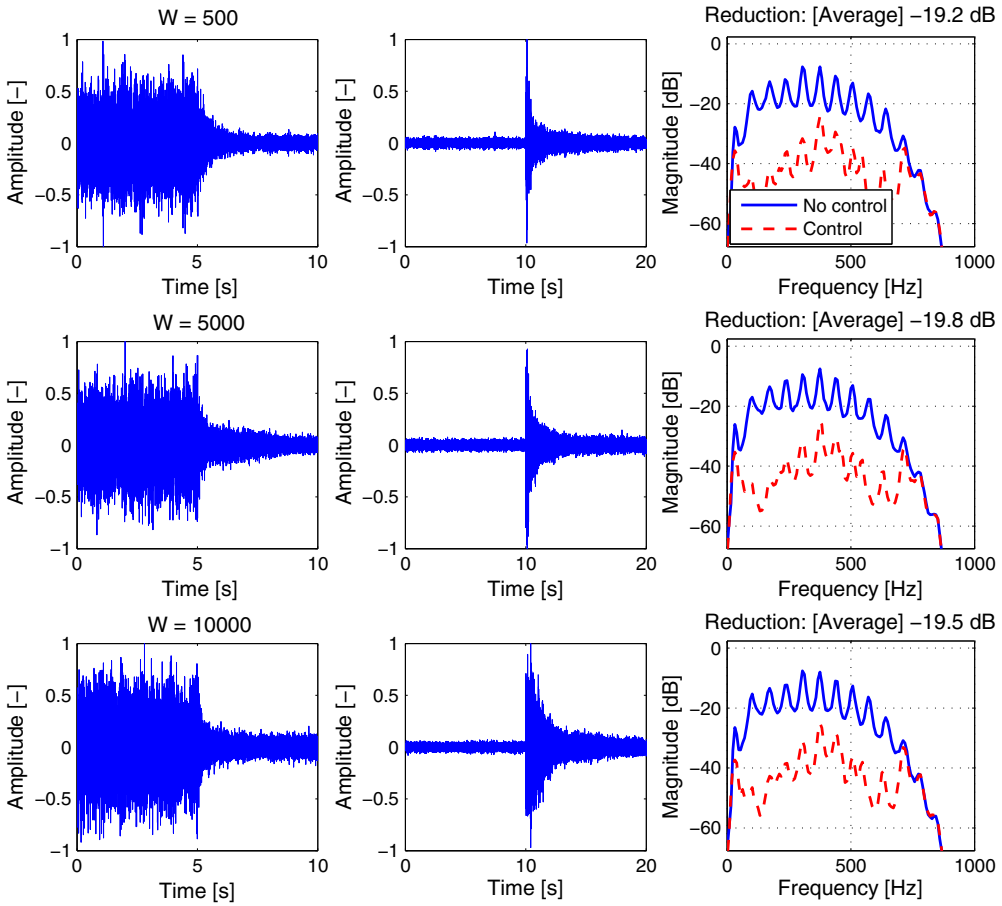


Figure 7. Experimental results for different window lengths. In the left column, the convergence of the algorithm is tested. The measured error signal is shown and the filter is turned on after 5 s. In the middle column, the tracking performance is shown. The reference signal is shifted after 10 s, and again, the measured error signal is shown. In the right column, the power spectrum of the signal with and without control is shown.

The results of the experiments are shown in Figure 7. For all the experiments, a filter that uses the old filter coefficients as initial values at every reset are used because the numerical simulations indicated that the performance of that variant is better. In the first column, the convergence of the algorithm is shown; in the second column, the tracking performance is shown; and in the third column, the power spectra of the error signal with and without control are shown and the associated reduction. These results closely resemble the simulation results.

It can be seen that the filter converges faster for longer window lengths, but has a slower tracking performance than with a short window length. The overall reduction after convergence is approximately the same for different window lengths. Similar experiments for the fast-array Kalman filter have been carried out by the authors, and the results can be found in [6].

## 4. CONCLUSIONS

A new algorithm has been derived, named the mixed windowed RLS, that approximates the solution given by the sliding window RLS filter. This algorithm has a linear calculation complexity and works by using a convex mixing scheme of the filter coefficients, resulting from two parallel finite length, growing memory fast-array RLS filters. This mixing scheme is chosen in such a way that the amount of data, used for the calculation of the control signal, remains constant.



DOI: 10.1002/acs

Although the approximation of the fast-array sliding window RLS is not perfect, especially for longer window lengths, the advantage of this algorithm is the elimination of long-term round-off error propagation at a lower calculation complexity than the fast-array sliding window RLS filter with reset. The performance of the filter has been validated in both numerical and experimental tests.

Further research will be devoted to the implementation of the secondary path uncertainty in the state estimation problem and to investigate a proper way to automatically tune the filter parameters so that the MSE will be more smooth. When this has been derived, an extension of the algorithm to multiple input and multiple output systems should be possible.

## APPENDIX

In this appendix, a derivation is given of Equation (19). Consider a certain time instance $i$ and introduce the matrices:

$$P_i = \left(\Pi + H_i^T H_i\right)^{-1}, \tag{A.1}$$

$$P_{V:i} = \left(\Pi + H_{V:i}^T H_{V:i}\right)^{-1}, \tag{A.2}$$

The inverse can be rewritten to

$$P_i^{-1} = \Pi + H_{i-1}^T H_{i-1} + \hat{r}_{n_w,i}\hat{r}_{n_w,i}^T, \tag{A.3}$$

$$= P_{i-1}^{-1} + \hat{r}_{n_w,i}\hat{r}_{n_w,i}^T, \tag{A.4}$$

$$P_{V:i}^{-1} = \Pi + H_{V:i-1}^T H_{V:i-1} + \hat{r}_{n_w,i}\hat{r}_{n_w,i}^T, \tag{A.5}$$

$$= P_{V:i-1}^{-1} + \hat{r}_{n_w,i}\hat{r}_{n_w,i}^T, \tag{A.6}$$

We use the matrix inversion lemma to rewrite these two expressions in a recursive form. The matrix inversion lemma is given by:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}. \tag{A.7}$$

Do the following identifications:

$$A \leftarrow P_{i-1},\ B \leftarrow \hat{r}_{n_w,i},\ C \leftarrow 1,\ D \leftarrow \hat{r}_{n_w,i}^T, \tag{A.8}$$

This gives

$$P_i = P_{i-1} - \frac{P_{i-1}\hat{r}_{n_w,i}\hat{r}_{n_w,i}^T P_{i-1}}{1 + \hat{r}_{n_w,i}^T P_{i-1}\hat{r}_{n_w,i}}, \tag{A.9}$$

Repeating these arguments for Equation (A.6) gives

$$P_{V:i} = P_{V:i-1} - \frac{P_{V:i-1}\hat{r}_{n_w,i}\hat{r}_{n_w,i}^T P_{V:i-1}}{1 + \hat{r}_{n_w,i}^T P_{V:i-1}\hat{r}_{n_w,i}}, \tag{A.10}$$

We consider the following solution:

$$\begin{aligned}
\hat{w}_{mix,i} &= \alpha_i \left(\Pi + H_i^T H_i\right)^{-1} H_i^T y_i + \beta_i \left(\Pi + H_{V:i}^T H_{V:i}\right)^{-1} H_{V:i}^T y_{V:i}, \\
&= \alpha_i P_i H_i^T y_i + \beta_i P_{V:i} H_{V:i}^T y_{V:i}, \\
&= \alpha_i P_i \left(H_{i-1}^T y_{i-1} + \hat{r}_{n_w,i}\hat{d}_i\right) + \beta_i P_{V:i} \left(H_{V:i}^T y_{V:i-1} + \hat{r}_{n_w,i}\hat{d}_i\right), \\
&= \alpha_i \left(P_{i-1} - \frac{P_{i-1}\hat{r}_{n_w,i}\hat{r}_{n_w,i}^T P_{i-1}}{1 + \hat{r}_{n_w,i}^T P_{i-1}\hat{r}_{n_w,i}}\right)\left(H_{i-1}^T y_{i-1} + \hat{r}_{n_w,i}\hat{d}_i\right) \\
&\quad + \beta_i \left(P_{V:i-1} - \frac{P_{V:i-1}\hat{r}_{n_w,i}\hat{r}_{n_w,i}^T P_{V:i-1}}{1 + \hat{r}_{n_w,i}^T P_{V:i-1}\hat{r}_{n_w,i}}\right)\left(H_{V:i}^T y_{V:i-1} + \hat{r}_{n_w,i}\hat{d}_i\right),
\end{aligned} \tag{A.11}$$



DOI: 10.1002/acs

Now consider the following definitions:

$$\hat{w}_i = P_i H_i^T y_{i-1}, \tag{A.12}$$

$$\hat{w}_{V:i} = P_{V:i} H_{V:i}^T y_{V:i}. \tag{A.13}$$

Working out Equation (A.11) and substitution of Equations (A.12 and A.13) gives

$$\begin{aligned}\hat{w}_{mix,i} = \alpha_i &\left( \hat{w}_{i-1} + \frac{P_{i-1}\hat{r}_{n_w,i}}{1 + \hat{r}_{n_w,i}^T P_{i-1}\hat{r}_{n_w,i}} \left( \hat{d}_i - \hat{r}_{n_w,i}^T w_{i-1} \right) \right) \\ + \beta_i &\left( \hat{w}_{V:i-1} + \beta_i \frac{P_{V:i-1}\hat{r}_{n_w,i}}{1 + \hat{r}_{n_w,i}^T P_{V:i-1}\hat{r}_{n_w,i}} \left( \hat{d}_i - \hat{r}_{n_w,i}^T w_{V:i-1} \right) \right).\end{aligned} \tag{A.14}$$

which we can rewrite more compactly by using the following definitions:

$$K_i = P_{i-1}\hat{r}_{n_w,i}, \tag{A.15}$$

$$R_i = 1 + \hat{r}_{n_w,i}^T P_{i-1}\hat{r}_{n_w,i}, \tag{A.16}$$

$$\epsilon_i = \hat{d}_i - \hat{r}_{n_w,i}^T w_{i-1}, \tag{A.17}$$

$$K_{V:i} = P_{V:i-1}\hat{r}_{n_w,i}, \tag{A.18}$$

$$R_{V:i} = 1 + \hat{r}_{n_w,i}^T P_{V:i-1}\hat{r}_{n_w,i}, \tag{A.19}$$

$$\epsilon_{V:i} = \hat{d}_i - \hat{r}_{n_w,i}^T w_{V:i-1}. \tag{A.20}$$

This gives the result, stated in Equation (19):

$$\hat{w}_{mix,i} = \alpha_i \left( \hat{w}_{i-1} + \frac{K_i}{R_i}\epsilon_i \right) + \beta_i \left( \hat{w}_{V:i-1} + \frac{K_{V:i}}{R_{V:i}}\epsilon_{V:i} \right). \tag{A.21}$$

## REFERENCES


1. Elliott SJ. *Signal Processing for Active Control*. Academic Press: London, 2001; 124.
2. Bjarnason E. Active noise cancellation using a modified form of the filtered-x LMS algorithm. *Proceedings of Eusipco 92, 6th European Signal Processing Conference*, Brussels, Belgium, 1992; 1053–1056.
3. Sayyarrodsari B, How JP, Hassabi B, Carrier A. Estimation-based synthesis of $h_\infty$-optimal adaptive FIR filters for filtered-LMS problems. *IEEE Transactions on Signal Processing* 2001; **49**(1):164–178.
4. Omoto A, Fujiwara K. Behavior of adaptive algorithms in active noise control systems with moving noise sources. *Acoustical Science and Technology* 2002; **23**(2):84–89.
5. Berkhoff AP. Control strategies for active noise barriers using near-field error sensing. *The Journal of the Acoustical Society of America* 2005; **118**(3):1469.
6. Van Ophem S, Berkhoff AP. Multi-channel Kalman filters for active noise control. *The Journal of the Acoustical Society of America* 2013; **133**(4):2105–2115.
7. Fraanje R, Sayed AH, Verhaegen M, Doelman NJ. A fast-array Kalman filter solution to active noise control. *International Journal of Adaptive Control and Signal Processing* 2005; **19**(2-3):125–152.
8. Sayed AH. *Fundamentals of Adaptive Filtering*. Wiley: NY, 2003; 732–873.
9. Park P, Cho YM, Kailath T. Chandrasekhar recursion for structured time-varying systems and its application to recursive least squares problems. *Second IEEE Conference on Control Applications* 1993; **2**:797–803.
10. Van Ophem S, Berkhoff AP. Active control of time-varying broadband noise and vibrations using a sliding-window Kalman filter. *Proceedings of the International Conference on Noise and Vibration Engineering, ISMA*, Leuven, Belgium, 2014; 107–118.
11. Chandrasekaran S, Sayed AH. Stabilizing the generalized Schur algorithm. *SIAM Journal on Matrix Analysis and Applications* 1996; **17**(4):950–983.
12. Stewart M, Van Dooren P. Stability issues in the factorization of structured matrices. *SIAM Journal on Matrix Analysis and Applications* 1997; **18**(1):104–118.




DOI: 10.1002/acs

13. Arenas-Garcia J, Figueiras-Vidal A, Sayed A. Mean-square performance of a convex combination of two adaptive filters. *IEEE Transactions on Signal Processing* 2006; **54**(3):1078–1090.
14. Silva MTM, Nascimento VH. Improving the tracking capability of adaptive filters via convex combination. *IEEE Transactions on Signal Processing* 2008; **56**(7):3137–3149.
15. Bershad N, Bermudez J, Tourneret JY. An affine combination of two LMS adaptive filters - transient mean-square analysis. *IEEE Transactions on Signal Processing* 2008; **56**(5):1853–1864.
16. Ferrer M, Gonzalez A, De Diego M, Pinero G. Convex combination filtered-x algorithms for active noise control systems. *IEEE Transactions on Audio, Speech, and Language Processing* 2013; **21**(1):156–167.
17. Verhaegen M, Verdult V. *Filtering and System Identification: A Least Squares Approach*. Cambridge University Press: Cambridge, 2007; 219–227.
18. Wesselink JM, Berkhoff AP. Fast affine projections and the regularized modified filtered-error algorithm in multichannel active noise control. *The Journal of the Acoustical Society of America* 2008; **124**(2):949–960.



DOI: 10.1002/acs