Fort Lauderdale, Florida, USA
ACTIVE 99
1999 December 02–04

THE KALMAN FILTER IN ACTIVE NOISE CONTROL

Paulo A. C. Lopes
Moisés S. Piedade

IST/INESC
Rua Alves Redol n.9
1000 Lisboa, Portugal
paclopes@eniac.inesc.pt

INTRODUCTION

Most Active Noise Control (ANC) systems use some form of the LMS [5][7] algorithm due to its reduced computational complexity. However, the problems associated with it are well-known, namely slow convergence and high sensitivity to the eigenvalue spread [3][7]. To overcome this problems the RLS algorithm is often used, but it is now widely known, that the RLS loses many of its good properties for a forgetting factor lower than one. Namely, it has been shown that in some application the LMS algorithm is actually better in tracking non-stationary signals than the RLS algorithm [2][3]. One approach, which is works well with non-stationary signals, is to use some specialized form of the Kalman filter, which can be interpreted as a generalization of the RLS algorithm [1][3][4]. The Kalman filter has a high computational complexity, similar to that of the RLS algorithm, which can make it costly for some applications. Nevertheless for narrow-band ANC, the number of taps is not very large [7] and the application of the Kalman filter in ANC may be easily handled by today DSP's. In fact, in multi-channel applications the Kalman gain can be computed only once for all the error signals, so the complexity hardly increases with a moderated number of those. Even in the cases where it can not be easily implemented in real time, it can still be used as a benchmark. In fact the Kalman filter is optimal for a given system model [1]. In this paper, specialized version of the Kalman filter fitted to ANC is developed both for primary and secondary path modeling. It is shown throw computer experiments that a large reduction in the residual noise can be achieved in non-stationary environments, compared with the LMS and RLS based algorithms, specially with on-line secondary path modeling.

LMS, RLS AND KALMAN ALGORITHMS

The best known algorithms in adaptive filtering are the LMS and RLS algorithms. They both try to solve the problem of estimating the desired signal $d(n)$ filtering the reference signal $u(n)$ through an unknown FIR filter $\hat{\mathbf{w}}(n)$, resulting in the estimate $y(n)$. The Kalman filter has its roots in control theory and represents the solution to a very general estimation problem in state-space formulation [1].

LMS algorithm. The LMS is the most used algorithm in adaptive filtering. It is a gradient descent algorithm; it adjusts the adaptive filter taps modifying them by an amount proportional to the instantaneous estimate of the gradient of the error surface. It is represented in equation (1).

$$\begin{aligned} y(n) &= \hat{\mathbf{w}}^H(n).\mathbf{u}(n) \\ e(n) &= d(n) - y(n) \\ \hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n) + \mu \cdot \mathbf{u}(n) \cdot e(n) \end{aligned} \quad . \tag{1}$$

The step size μ is a constant parameter of the algorithm.

RLS algorithm. The RLS algorithm performs at each instant an exact minimization of the sum of the squares of the desired signal estimation errors. These are its equations:

$$\begin{aligned} y(n) &= \hat{\mathbf{w}}^H(n).\mathbf{u}(n) \\ \alpha(n) &= d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n) \\ \pi(n) &= \mathbf{u}^H(n).\mathbf{P}(n-1) \\ k(n) &= \lambda + \pi(n).\mathbf{u}(n) \\ \mathbf{k}(n) &= \frac{\mathbf{P}(n-1).\mathbf{u}(n)}{k(n)} \\ \hat{\mathbf{w}}(n) &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n).\alpha^*(n) \\ \mathbf{P}'(n-1) &= \mathbf{k}(n).\pi(n) \\ \mathbf{P}(n) &= \frac{1}{\lambda}\left(\mathbf{P}(n-1) - \mathbf{P}'(n-1)\right) \end{aligned} \quad . \tag{2}$$

To initialize the algorithm $\mathbf{P}(n)$ should be made equal to $\delta^{-1}.\mathbf{I}$ where δ is a small positive constant.

Kalman algorithm. The Kalman filter gives the solution to the following problem. Given the state space model in equation (3) or figure 1 where $\mathbf{Q}_1(n)$, $\mathbf{Q}_2(n)$, $\mathbf{C}(n)$, $\mathbf{\Phi}(n+1,n)$ and $\mathbf{y}(n)$ are know quantities, find the best estimate $\hat{\mathbf{x}}(n|\mathbf{y}_n)$ to the state vector $\mathbf{x}(n)$, in the expected least errors squares sense.

$$\begin{aligned} \mathbf{x}(n+1) &= \mathbf{\Phi}(n+1,n).\mathbf{x}(n) + \mathbf{v}_1(n) \\ \mathbf{y}(n) &= \mathbf{C}(n).\mathbf{x}(n) + \mathbf{v}_2(n) \end{aligned} \tag{3}$$

$$\mathbf{E}\left[\mathbf{v}_1(n).\mathbf{v}_1^H(n)\right] = \mathbf{Q}_1(n)\,,\ \mathbf{E}\left[\mathbf{v}_2(n).\mathbf{v}_2^H(n)\right] = \mathbf{Q}_2(n) \tag{4}$$

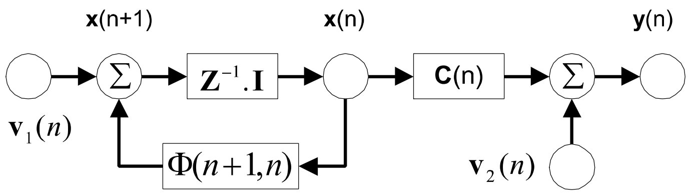

Figure 1 – Space state model used for the system in the Kalman filter.

The Kalman filter equations are:

$$\begin{aligned}
\mathbf{G}(n) &= \mathbf{\Phi}(n+1,n).\mathbf{K}(n,n-1).\mathbf{C}^H(n)[\mathbf{C}(n).\mathbf{K}(n,n-1).\mathbf{C}^H(n)+\mathbf{Q}_2(n)]^{-1} \\
\alpha(n) &= y(n)-\mathbf{C}(n).\hat{\mathbf{x}}(n|\mathbf{y}_{n-1}) \\
\hat{\mathbf{x}}(n+1|\mathbf{y}_n) &= \mathbf{\Phi}(n+1,n).\hat{\mathbf{x}}(n|\mathbf{y}_{n-1})+\mathbf{G}(n).\alpha(n) \\
\hat{\mathbf{x}}(n|\mathbf{y}_n) &= \mathbf{\Phi}(n,n+1).\hat{\mathbf{x}}(n+1|\mathbf{y}_n) \\
\mathbf{K}(n) &= \mathbf{K}(n,n-1)-\mathbf{\Phi}(n,n+1).\mathbf{G}(n).\mathbf{C}(n).\mathbf{K}(n,n-1) \\
\mathbf{K}(n+1,n) &= \mathbf{\Phi}(n+1,n).\mathbf{K}(n).\mathbf{\Phi}^H(n+1,n)+\mathbf{Q}_1(n)
\end{aligned} \tag{5}$$

To initialize the algorithm, $\hat{\mathbf{x}}(0|\mathbf{y}_0)$ should be made equal to $E[\mathbf{x}(0)]$ and $\mathbf{K}(0)$ should be made equal to $E[\mathbf{x}(0).\mathbf{x}^H(0)]$.

Tracking vs. convergence

There has been much interest in fast convergence algorithms, but is fast convergence really needed in ANC? Fast convergence means that the time the algorithm takes from its initialization to the point it reaches an optimal value is short. These algorithms are of great interest in telecommunications system where the goal is to reduce the size of the training sequence and the corresponding overhead. In ANC, however, the main interest is not reducing the time the system takes to reach an optimal noise reduction since it is powered on! What is important is improving the tracking ability of the algorithms to sudden and to continuing path changes in the physical system. That is, to reduce the time taken by the algorithms to respond to a sudden change in the physical system or to improve the continuing tracking of slow changes. Although tracking and convergence are related, they are not really the same. One difference comes from the following fact. Fast converging algorithms achieve some of their fast response, not by an improvement in tracking, but by a transitory widening of their filters in the firsts iterations. With an analogy with the LMS algorithm, it can be said that they increase the step size in the first iterations and then progressively decrease it. With this in mind, this paper will focus in the tracking properties of the algorithms and not their initial convergence.

Comparing the tracking performance of the LMS, RLS and Kalman

In order to illustrate the previous discussion and to show why the Kalman filter achieves a better tracking performance, the tracking abilities of the LMS, RLS and Kalman algorithms will be studied. A number of simplifications are used since the only interest of this analysis is improving the understanding of the working of the Kalman filter. First, only one-tap systems are analyzed. Second, the results refer only to steady state, namely the values for $K(n,n-1)$ and $P(n)$ in the Kalman and RLS algorithms used were for infinite *n*. Finally independence theory was used [3], which in short means that $u(n)$ was assumed uncorrelated from $\hat{w}(n)$. With these simplifications, it is possible to reach equation (6), which represents the evolution of the filter-taps-estimates in all the algorithms.

$$\hat{w}(n+1)=(1-\alpha).\hat{w}(n)+\alpha.w'(n) \tag{6}$$

$w'(n)=d(n)/u(n)$ is an instantaneous estimate of the filter tap and α is given by the following expressions:

$$\alpha_{LMS}=\mu.R \tag{7}$$

$$\alpha_{RLS} = (1-\lambda) \tag{8}$$

$$\begin{cases} \alpha_{kalman} \approx \dfrac{1}{1+\dfrac{Q_2}{R.Q_1}} \Leftarrow \dfrac{Q_2}{R.Q_1} << 1 \\ \alpha_{kalman} \approx \dfrac{1}{1+\sqrt{\dfrac{Q_2}{R.Q_1}}} \Leftarrow \dfrac{Q_2}{R.Q_1} >> 1 \end{cases} \tag{9}$$

This means that all the algorithms can be seen as producing an instantaneous noisy estimate of filter taps, which is then processed by a low-pass first order IIR filter. This filter bandwidth should be chosen taking in to account the compromise between eliminating the noise from the instantaneous estimate and preserving the original signal. Only the Kalman filter gives the ideal solution. Indeed, in the Kalman algorithm the filter bandwidth is adjusted taking in to account the relation between the measurement noise, the reference signal power value and the state noise value. This means, for example, that if the reference signal values is lower the filter should be tighter, since the measurement noise will greatly affect the instantaneous estimate. This is exactly what the RLS algorithm can not do, because its filter bandwidth is fixed. The LMS algorithm has this feature but the quantitative values are not adequate.

Noise reduction vs. system identification

The previous discussion points out that the LMS and RLS algorithms have poor tracking performance for a colored reference signal when compared with the Kalman filter. However, the discussion only focussed on the estimation of the filter taps. The differences between the performance of the algorithms are lower if the goal is to achieve noise reduction instead of system identification.

The convergence of the algorithms can be interpreted as the convergence of several modes corresponding to the eigenvectors of the autocorrelation matrix of the reference signal. If the path to be identified has a somewhat flat frequency response then the modes with lower energy (eigenvalue) in the reference signal will also have low energy in the desired signal. In this case, the step-size and forgetting factor of the LMS and RLS algorithm can be adjusted so that the filters corresponding to the higher energy modes have similar bandwidth to the ones generated by the Kalman filter. This way, a reasonably good tracking is achieved for the modes that matter. However, this changes when the path to be identified has resonances or the desired noise reduction is high.

On the other hand, if the real interest is system identification and not noise reduction (as in the case of on-line secondary path modeling) all the modes are equally important. Therefore, in this case the Kalman filter clearly outperforms the LMS and RLS algorithms.

KALMAN FILTER IN ACTIVE NOISE CONTROL

Figure 2 represents the basic feedforward ANC system where $P_z(n)$ represents the primary path, $S_z(n)$ represents the secondary path and $F_z(n)$ the feedback path.

In the remainder of this paper, the feedback path will not be considered. It will be assumed that the effect of the feedback path was been eliminated by some other means, namely throw adaptive cancellation or by the use of some non-acoustic sensor.

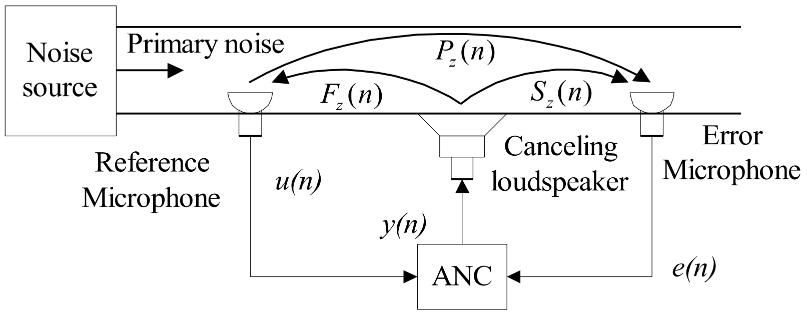

Figure 2. A Basic Active Noise Control System.

Kalman in primary path modeling

A state space model for the ANC system.

Writing down the equations for the system in figure 2 we get:

$$\begin{aligned} e(n) &= \sum_{i=0}^{N} p_i(n).u(n-i) + \sum_{i=0}^{N} s_i(n).y(n-i) + r(n) \\ &= \sum_{i,j=0}^{N} s_i(n).w_j(n-i).u(n-i-j) + \sum_{i=0}^{N} s_i(n).y(n-i) + r'(n) \end{aligned} \tag{10}$$

In order to apply the Kalman filter, this equation has to be fitted to the state space model in equation (3). The goal is to estimate $w(n)$ so this should be the state vector. Nevertheless, equation (10) states that *e(n)* depends not only of $w(n)$ but of $w(n-i)$ with *i* going from zero to *M*. This would increase the state vector dramatically, making it impracticable to implement the algorithm. Heaven if slow changes are assumed and $w_o(n-i)$ is taken equal to $w_o(n)$, $\mathbf{C}_j(n)$ is still given by $\sum_{i=1}^{N} s_i(n).x(n-i-j)$ which is relatively costly to implement.

At this point one may think that it is a good idea to use a filtered-X configuration as in the FXLMS algorithm [7]. However, it is known that this configuration doesn't work well in fast converging algorithms [8]. For this, the configuration shown in figure 3 is more helpful.

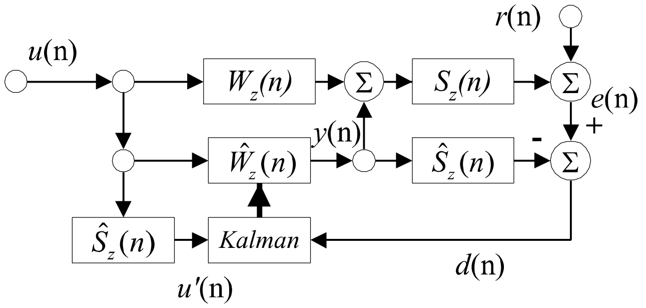

Figure 3. Dealing with the secondary path in the Kalman algorithm.

If slow changes are assumed, so that *S(z)* and *W(z)* can be exchanged, and if $S(z)$ is taken to be equal to $\hat{S}(z)$ then the effects of the secondary path are removed. Now, the traditional version of the Kalman algorithm used in adaptive filtering can be applied. This is done using the following identities:

Kalman Variables	$\mathbf{Q}_1(n)$	$\mathbf{Q}_2(n)$	$\mathbf{y}(n)$	$\mathbf{C}(n)$	$\hat{\mathbf{x}}(n)$	$\mathbf{\Phi}(n+1,n)$
ANC	$\mathbf{Q}=\gamma_w.\mathbf{I}$	$J_{\min}$	$d(n)$	$\mathbf{u}'^T(n)$	$\hat{\mathbf{w}}^*(n)$	$\lambda.\mathbf{I}$

Kalman Variables	$\mathbf{x}(n)$	$\mathbf{v}_1(n)$	$\mathbf{v}_2(n)$
ANC	$\mathbf{w}^*(n)$	$\mathbf{v}(n)$	$r(n)$

It should be noticed that in this model, the variations in the primary path are account for in the state noise $\mathbf{v}(n)$; this is what gives to the algorithm its finite memory. These identities are close to a random walk state-space model [3] for the variations in the state, $\mathbf{w}(n)$. However, some modifications were made to make the model more realistic and increase the stability of the algorithm. The model is represented in equation (11). $\mathbf{v}_j(n)$ is a set of independent white noise processes. This reduces to the random walk state-space model for $\lambda = 1$. However, this model predicts that the variation of the state increases without limit. This is not a reasonable prediction for ANC. Instead γ_w and λ should be chosen together so that the predicted variance of $\mathbf{w}_j(n)$ (state) matches what is expected. Namely, γ_w should be set to $\sigma_w.\sqrt{(2-\lambda)/\lambda}$ where σ_w^2 is the estimated value for the variance of the filters taps.

$$\mathbf{w}_j(n+1) = \lambda.\mathbf{w}_j(n) + (1-\lambda).\mathbf{v}_j(n) \tag{11}$$

The power of $r(n)$, $E[r(n)^2]$, is $J_{\min}$. The values of λ and $J_{\min}$ should be chosen according with the particular application. If these parameters are correct, the Kalman filter achieves a near optimal solution. There are also extended version of Kalman filter where these parameters are made adaptive [1].

Simulation results

Now the results of several computer simulations will be shown. They compare the LMS, RLS and Kalman algorithms using the configuration shown in figure 3, where the Kalman block is replaced by LMS or RLS blocks when needed.

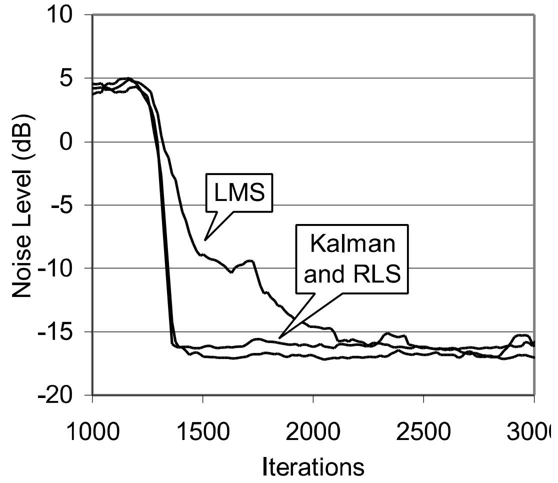

Figure 4 – Initial convergence of the algorithms.

The secondary path was modeled as a 32-taps low pass FIR filter $\mathbf{s}(n)$ and the primary path by the convolution of the secondary path with another 32-taps low pass FIR filter $\mathbf{w}(n)$. The controller filter $\hat{\mathbf{w}}(n)$ was also a 32-taps FIR filter. In this way, perfect identification was possible. However, to make the system more realistic a noise signal $r(n)$ was added to the system, as shown in figure 3. The amplitude of this signal was set to 0.1.
In all the experiments, the reference signal $u(n)$ was generated by passing white noise through a filter with two flat sections 10-dB apart. This was meant to clearly show two different modes of convergence in the LMS algorithm. The parameters of the algorithms were set so that all of then reached the same steady state residual noise. The charts were obtained from ensemble average of many computer simulations, where for each simulation the square error signal was processed by a moving average filter.

Figure 4 compares the learning of the three algorithms. As expected the RLS and Kalman algorithms converge very quickly, while the LMS is much slower.

Figure 5 compares the responses of the three algorithms to sudden changes in the secondary path. The curves are similar to the initial learning curves. They are rather round because of the moving average filter used for smoothing. Again, the Kalman and RLS algorithms perform much better than the LMS.

Figure 6 compares the tracking performance of the algorithms. $\mathbf{w}_j(n)$ was varied according with equation (3) with $\lambda = 0.99999$ and γ_w chosen so that σ_w^2 was one. In this way, the model used in the derivation of the Kalman filter is the same as the one used in the simulation, so the Kalman filter gives almost optimal results. As before $J_{\min}$ was 0.1. It can be seen that Kalman filter achieves more 3-dB of noise reduction than the RLS and LMS algorithms when tracking continuing changes.

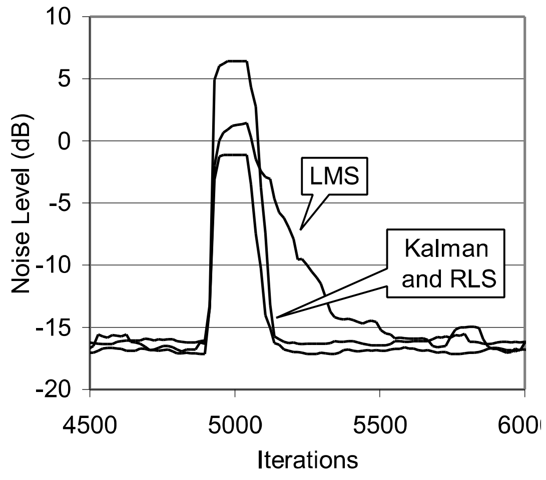

Figure 5 – Response of the algorithms to sudden path changes.

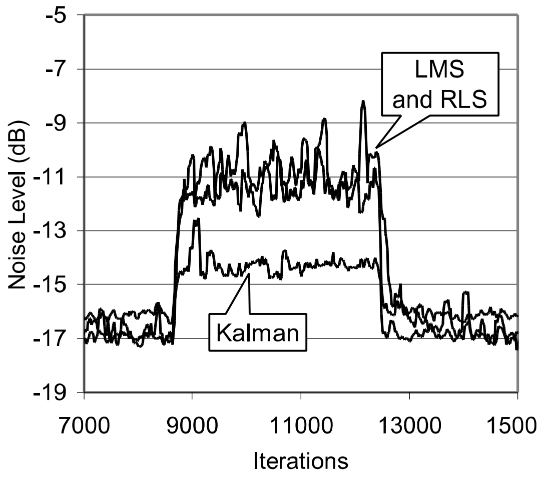

Figure 6 – Response of the algorithms to tracking continues path changes.

Kalman filter for on-line secondary path modeling

There are two major algorithms used in on-line secondary path modeling (OSPM). The additive random noise technique and the overall modeling algorithm. They both try to solve the problems with the basic approach, a classical system identification setup using as reference the anti-noise signal and as output the error signal [7]. This approach gives biased results due to the correlation between the anti-noise signal and the reference signal.

In the first algorithm an uncorrelated low-level white noise signal is added to the anti-noise and is used for the path estimate. In the second algorithm, the bias is removed by modeling the overall system: primary and secondary path; this technique will be used in this paper. This is because the injected noise of the additive random noise technique is obviously an undesired feature and it is difficult to keep it at very low values and still maintain a good tracking behavior.

The fact is, however, that without an additive noise signal, and in a stationary situation, there is not enough information available for the ANC system to uniquely model the overall system, or for that matter, the secondary path. Although this might seam to make the overall modeling algorithm impractical that's not true. The reason the system can not be uniquely identified is simply because there are several possible values for the secondary and primary path that minimize the system estimation error ($\varepsilon(n)$ in figure 7). Namely, working in the Z-transform domain, our knowledge only allows us to write one equation (equation (12)) but there are two unknowns $\hat{S}(z)$ and $\hat{P}(z)$.

$$\begin{aligned}\varepsilon(z) &= W(z).S(z).x(z) + S(z).y(z) + r(z) - \hat{P}(z).x(z) - \hat{S}(z).y(z) \Leftrightarrow \\ \varepsilon(z) &= \left[W(z).S(z) + S(z).\hat{W}(z) + r(z) - \hat{P}(z) - \hat{S}(z).\hat{W}(z)\right].x(z)\end{aligned} \tag{12}$$

However, for any chosen solution the predicted error signal $\hat{e}(n)$ will be close to the actual error signal (since $\varepsilon(n)$ is minimal), so minimizing the first corresponds to minimize the last, which is our goal. Looking it in another way, you can say that there are modes of the joined reference signal in overall modeling algorithm (X(n) in figure 7) that do not have energy due to the correlation between both signals. So, components of the system excited by these modes can not be estimated, but are also not relevant.

The state-space model for on-line secondary path modeling

Once again, writing down the equations for the system in figure 2 it is obtained:

$$e(n) = \sum_{i=0}^{N} p_i(n).x(n-i) + \sum_{i=0}^{M} s_i(n).y(n-i) + r(n). \tag{13}$$

So using the definitions,

$$\begin{aligned}X(n) &= [x(n), x(n-1), \cdots, x(n-N+1), y(n), y(n-1), \cdots, y(n-M+1)]^T \\ T(n) &= [p_0, p_1, \cdots, p_{N-1}, s_0, s_1, \cdots, s_{M-1},]^T\end{aligned} \tag{14}$$

It is possible to write,

$$e(n) = T(n)^T.X(n) + r(n) \tag{15}$$

The goal is to estimate the secondary path and primary path, which means estimating $T(n)$ so this should be the state vector. That results in the following correspondence:

Kalman Variables	$\mathbf{Q}_1(n)$	$\mathbf{Q}_2(n)$	$\mathbf{y}(n)$	$\mathbf{C}(n)$	$\hat{\mathbf{x}}(n)$	$\mathbf{\Phi}(n+1,n)$
OSPM	$\mathbf{Q} = \gamma_T.\mathbf{I}$	$J_{\min}$	$e(n)$	$\mathbf{X'}^T(n)$	$\hat{\mathbf{T}}(n)$	$\lambda.\mathbf{I}$

Kalman Variables	$\mathbf{x}(n)$	$\mathbf{v}_1(n)$	$\mathbf{v}_2(n)$
OSPM	$\mathbf{T}(n)$	$\mathbf{v}(n)$	$r(n)$

As for the case of the Kalman filter for primary path modeling, a variation of the random walk state-space model was used (equation (16)). All the remarks in that section also apply here. Once more, the random-walk state model predicts that the variation of the state increases without limit, so the transition matrix was set to $\lambda\mathbf{I}$ instead of the traditional $\mathbf{I}$. In on-line secondary path modeling, this is even more important because the lack of excitation in some modes may allow the state vector to diverge. The state noise $\mathbf{v}_j(n)$ is a set of independent white noise processes with variance γ_T. This should be set to $\sigma_T.\sqrt{(2-\lambda)/\lambda}$ where σ_T^2 is the estimated value for the variance of the filters taps.

$$\mathbf{T}_j(n+1) = \lambda.\mathbf{T}_j(n) + (1-\lambda).\mathbf{v}_j(n) \qquad (16)$$

The values of λ and $J_{\min}$ should be chosen according with the particular application. The resulting algorithm is represented in figure 7. A white noise signal $n(n)$ has added to the system. This is discussed in the next section, but it can now be stated that is serves to improve the stability of the algorithm. It is especially required when the Kalman gain is replaced by its RLS counterpart for the comparisons.

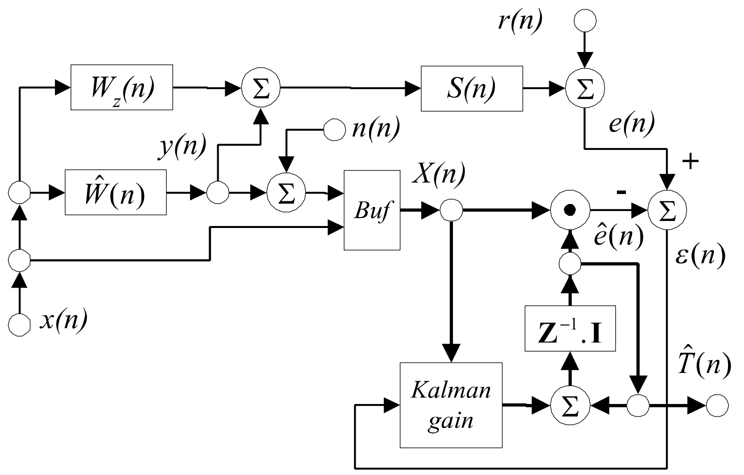

Figure 7. On line secondary path modeling with Kalman algorithm.

Adding white noise

As was previously referred, some modes of the joined reference signal do not have energy. This can be troublesome, because the lack of information regarding a specific part of the system may lead the algorithms to attribute arbitrary values for its estimate. This is especially true for the RLS algorithm. The LMS doesn't have this problem and, in the Kalman filter, it can be controlled by an appropriate selection of the space state model. That is, by the use of the model proposed in with the transition matrix is set to $\lambda\mathbf{I}$ instead of the traditional $\mathbf{I}$ as in the random walk state-space model.

To solve this problem for the case of the RLS algorithm one approach is to add a white noise signal $n(n)$ to the anti-noise input of the algorithm, as shown in figure 7. In this way the input signals are no longer perfectly correlated and all the modes are excited. Of course, this introduces a bias into the solution, but this can be kept small if the white noise signal is small. This was the technique used in this paper.

Simulation results

To analyze the algorithms it is important to have some quantitative measurement of the accuracy of the secondary path model (and primary). To do this, a normalized form of the misalignment (σ_s^2) will be used, as defined in equation (17).

$$\sigma_T^2(n) = \sum_{i=0}^{N-1}\left[\hat{T}_i(n) - T_i(n)\right]^2 \Big/ \sum_{i=0}^{N-1} T_i^2(n). \tag{17}$$

The misalignment, however, was to be used with caution. As stated before, in stationary situation, there are components of $T(n)$ that are not relevant, since $X(n)$ doesn't have energy in these modes. Therefore, high errors in these components doses not necessarily imply low levels of noise reduction. The misalignment fails to take this in to account. On the other hand, even in irrelevant components, high errors can give rise to numerical problems. Alternatively, they can make big damages when they became relevant throw some change in the system. Therefore, the misalignment is still a useful indicator.

The simulation results presented in this section compare the LMS, RLS and Kalman algorithms used for both secondary and primary path modeling. The configuration in figure 3 was used for primary path modeling and the configuration on figure 7 was used for secondary path modeling. The Kalman blocks were replaced by the RLS and LMS counterparts when required. The setup was very similar to the one in the section for primary path modeling. The secondary path was modeled as a 32-taps low pass FIR filter $\mathbf{s}(n)$, and the primary path by the convolution of the secondary path with another 32-taps low-pass FIR filter $\mathbf{w}(n)$. The controller filter $\hat{\mathbf{w}}(n)$ was also a 32-taps FIR filter. The measurement noise signal $r(n)$ amplitude was set to 0.1. A white noise signal $n(n)$ also with 0.1 of amplitude was added in the inputs of the algorithm as shown in figure 7. Once more, the reference signal $u(n)$ was generated by passing white noise through a filter with a frequency response with two flat sections 10-dB apart. The charts were obtained from ensemble average of 128 computer simulations, where for each simulation the square error signal was processed by a moving average filter.

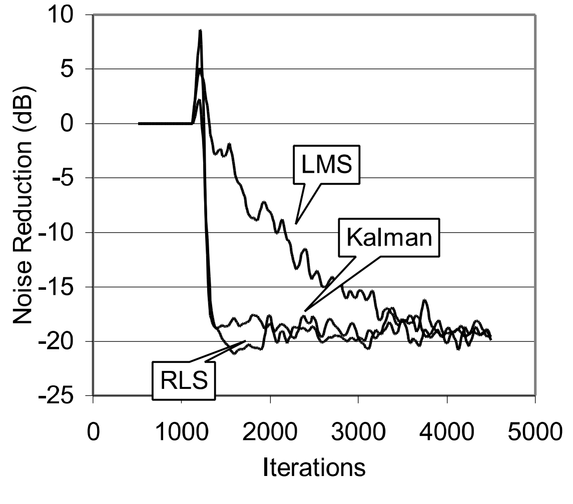

Figure 8 – Initial convergence of the algorithms
(with initial secondary path estimate obtained off-line).

In figure 8 the learning curves of the three algorithms are shown. It can be seen that the RLS and Kalman algorithms are much faster than the LMS. As before, the parameters of the algorithms were adjusted so that steady state the residual errors were the same. The noise reduction is bounded at -20dB by the added measurement noise signal $r(n)$.

Figures 9 and 10 compare the response of the algorithm to sudden path changes. The primary and secondary paths were delayed abruptly by three samples. Figure 10 plots the normalized misalignment and figure 9 plots the noise reduction. It can be seen that the Kalman filter is the quickest. It is also the one with lower misalignment. The LMS is the slowest although not very far from the RLS. It is interesting to note that the RLS achieves a high level of misalignment even with the added white noise. However, as can be seen in the figures, this high misalignment does not really degrade the noise reduction.

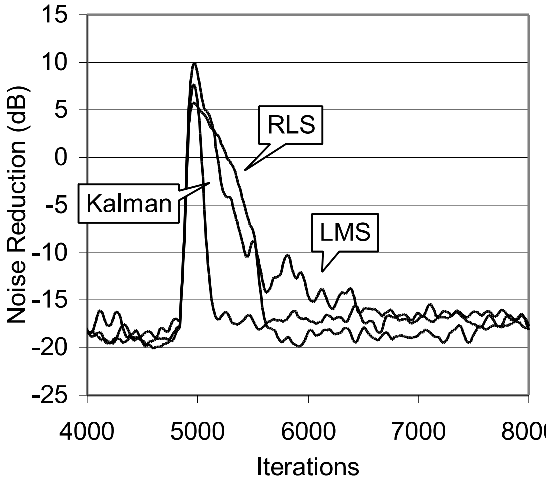

Figure 9 – Response of the algorithms to sudden path changes.

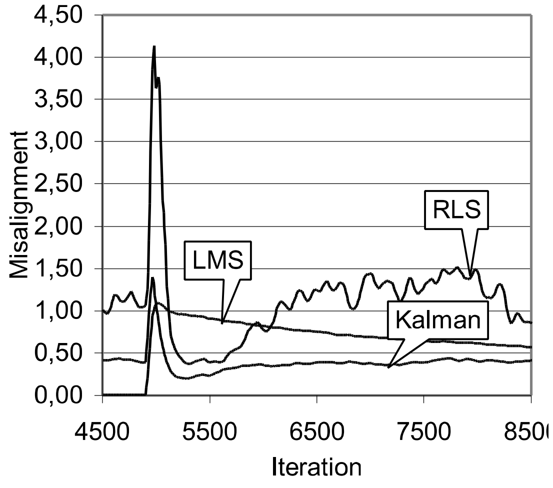

Figure 10 – Response of the algorithms to sudden path changes, with added white noise.

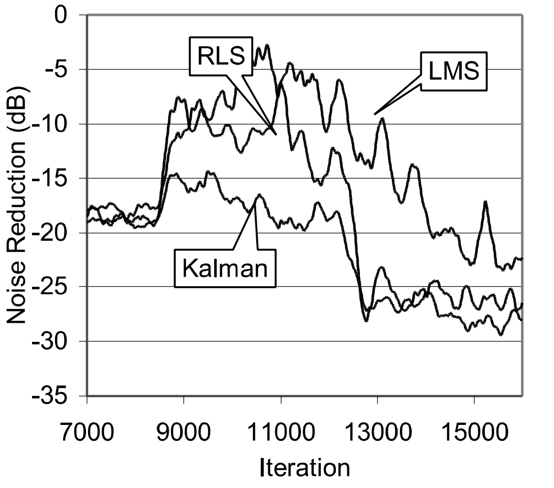

Figure 11 – Response of the algorithms to tracking continues path changes.

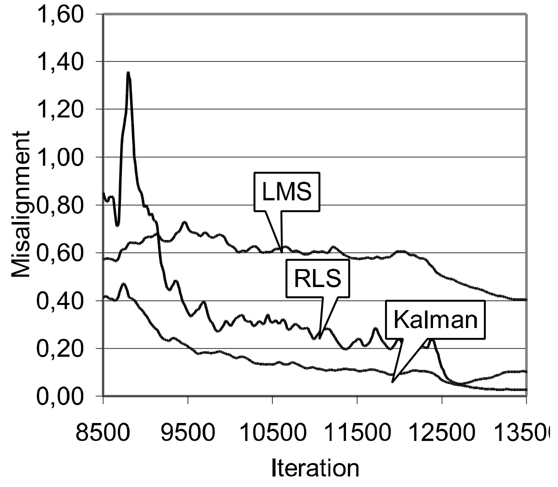

Figure 12 – Response of the algorithms to tracking continues path changes, with added white noise.

Figures 11 and 12 compare the tracking abilities of the algorithms. The primary and secondary paths were varied according to equation (15) with γ_T set so that the variances of the filters taps were one. Once more, the model used for the derivation of the Kalman filters is the same as the one used in the simulation, so the Kalman filter achieves a nearly optimal solution. λ was set to 0.99999. The Kalman filter gains as much as 12-dB of noise reduction to the LMS algorithm and about 7-dB to the RLS algorithm.

SUMMARY

In this paper a specialized version of the Kalman filter fitted to ANC was developed. A simple theoretical comparison with the RLS and LMS algorithms is made, which shows why the Kalman filter is able to outperform the formers. For primary path modeling, a setup that eliminates the effects of the secondary path was used to implement the traditional adaptive filters version of the Kalman filter with good results. For on-line secondary path modeling a Kalman filter version of the overall modeling algorithm was used. For this case, the lack of persistent excitation of all the modes of the signal can be a problem. To solve this a slightly different version of the random walk state space model was used in conjunction with the addition of a white noise signal in the inputs of the algorithm. The results were very good especially when the Kalman approach was used with on-line secondary path modeling. With off-line secondary path modeling, the Kalman was much quicker to adapt to sudden path changes than the LMS algorithm and it achieved more 3-dB of noise reduction than the LMS and RLS algorithms in tracking continuing changes. With on-line secondary path modeling, the Kalman filter was much quicker to adapt to sudden path changes than the RLS and LMS algorithms. When tracking continues changes, the Kalman achieved more 12-dB of noise reduction than the LMS algorithm and more 7-dB of noise reduction than the RLS algorithm.

REFERENCES

[1] Anderson, B. D. O.; Moore B. J.; "Optimal Filtering", Prentice-Hall, Inc., 1979.

[2] Eleftheriou, E.; Falconer, D. D.; "Tracking the Properties and Steady-State Performance of RLS Adaptive Filter Algorithms", IEEE Trans. Acoust. Speech Signal Process., vol. ASSP-34, pp. 1097-1110

[3] Haykin S., "Adaptive Filter Theory", Prentice-Hall, Inc., 1991.

[4] Haykin, S.; Sayed, A.H.; Zeidler, J.R.; Yee, P.; Wei, P.C., "Adaptive tracking of linear time-variant systems by extended RLS algorithms", Signal Processing, IEEE Transactions on, Volume: 45 5 , Page(s): 1118 -1128, May 1997.

[5] P. Lopes, B. Santos, M. Bento, M. Piedade, "Active Noise Control System", Recpad98 Proceedings, IST, Lisbon, Portugal, March 1998.

[6] Sayed, A.H.; Kailath, Thomas; "A State-Space Approach to Adaptive RLS Filtering", IEEE Signal Processing Magazine, pg. 18-60, July 1994.

[7] Sen M. Kuo and Dennis R. Morgan, "Active Noise Control Systems, Algorithms and DSP implementations", John Wiley & Sons, Inc., 1996.

[8] Stuart J. Flockton; "Fast Adaption Algorithms in Active Noise Control", Second Conference on Recent Advances in Active Noise Control of Sound and Vibration, pg. 802 -810, Virginia, 28 April 1993.