Efficient Block Frequency Domain Filtered-x applied to Phantom Sound Source Generation

Ronald M. Aarts[1], Alexander W.M. Mathijssen[1], Piet C.W. Sommen[2], John Garas[2]

[1] Philips Research Laboratories
WY8.25
5656 AA Eindhoven
The Netherlands
E-mail: *rmaarts@natlab.research.philips.com*

[2] Eindhoven University of Technology
Bldg Eh 6.33, P.O.Box 513
5600 MB Eindhoven
The Netherlands.
E-mail: *p.c.w.sommen@ele.tue.nl*

A phantom sound source is a virtual sound image which can be utilized in many applications such as stereo base widening, multimedia, and virtual reality engines. Generation of such a virtual sound image using adaptive filters that employ the filtered-x NLMS algorithm is discussed. Such filters are long and complex for the virtual source to cover the whole audio frequency range. Due to this complexity, real-time implementation on a sample-by-sample basis is not practical and block processing techniques, such as block frequency domain adaptive filters, have to be used. Several versions of the filtered-x algorithm in the frequency domain are derived and the different implementations are compared in terms of convergence, behavior and acoustical performance.

0 Introduction

When audio signals are played by a normal stereo setup, the sound perceived by a listener contains both directional and distance information from which the listener can locate the positions of the sound sources. In some applications like TV-sets and monitors, the distance between the loudspeakers has to be limited in order to obtain a compact arrangement. To give the loudspeaker base a virtual broadening, we made use of the phantom sound source concept. A phantom sound source is a virtual source which is perceived by the listener as a source at a different position than that of the real sources. The basic idea

behind phantom source generation is to alter the loudspeakers' input signals in such a way that the sound pressure at the listener's eardrums generated by the real loudspeakers is the same as that would be generated by the phantom source [1, 2].
Fig. 1 shows a situation where a phantom sound source LS_p at angle β is generated by two real loudspeakers at angle α. $\underline{\mathbf{w}}_L$ and $\underline{\mathbf{w}}_R$ are filters which are used to alter the input signals to LS_L and LS_R. The design of the filters $\underline{\mathbf{w}}_L$ and $\underline{\mathbf{w}}_R$ is the main task in generating a phantom source. These filters represent, in general, long and complex impulse responses which are also position dependent. Therefore it is difficult to obtain an analytical solution. For this reason, long adaptive filters are used which are implemented very efficiently in the frequency domain. This adaptive system is outlined as follows (see Fig. 1). Initially three loudspeakers are used, one (LS_p) un-filtered at the desired position of the phantom source, the other two (LS_L and LS_R) filtered by adaptive filters. White noise is played through loudspeaker LS_p while $\underline{\mathbf{w}}_L$ and $\underline{\mathbf{w}}_R$ are adapted to minimize the SPL measured at the listener's eardrums. When the system has converged the coefficients of $\underline{\mathbf{w}}_L$ and $\underline{\mathbf{w}}_R$ are frozen and the third loudspeaker (LS_p) is disconnected. When the two other loudspeakers are then driven through $-\underline{\mathbf{w}}_L$ and $-\underline{\mathbf{w}}_R$, there is the illusion that only the (not connected) third loudspeaker (LS_p) is playing.
In the following sections a Block Frequency Domain Adaptive Filter will be discussed, of which the complexity will be reduced in consequential steps.

1 Filtered-x algorithm

Fig. 2 shows a simplified (single point) version of the system shown in Fig. 1 together with an adaptive filter with one primary ($\underline{\mathbf{h}}_p$) and one secondary ($\underline{\mathbf{h}}_s$) acoustic path. The coefficients of the adaptive filter $\underline{\mathbf{w}}$ are updated by an algorithm based on the Normalized Least Mean Square (NLMS) algorithm. This algorithm updates the adaptive weight vector $\underline{\mathbf{w}}$ in the negative direction of the gradient vector $\underline{\nabla}$. It is shown in [3] that an estimate of this gradient vector is given by $\underline{\mathbf{x}} \cdot r$, in which $\underline{\mathbf{x}}$ is the input signal vector and r the residual signal and the update equation becomes:

$$\underline{\mathbf{w}}_{\text{new}} = \underline{\mathbf{w}}_{\text{old}} - \frac{2\alpha}{\sigma_x^2} \underline{\mathbf{x}} \cdot r, \tag{1}$$

where α is the adaptation coefficient which is normalized by the variance of the input signal $\sigma_x^2 = E\{x^2[k]\}$.
As can be seen in Fig. 2, the output signal $\hat{e}$ of the adaptive filter is first filtered by the secondary acoustic path $\underline{\mathbf{h}}_s$ before it is added in the microphone. For this reason, we need a slightly different algorithm than that given by (1); the so-called filtered-x algorithm [3]. The update part of this algorithm uses a filtered version x_f instead of the input signal x

itself. This is achieved by filtering x by an estimate of the secondary acoustic path $\hat{\underline{\mathbf{h}}}_s$ as shown in Fig. 2.

1.1 Block Frequency Domain Adaptive Filtering

Because of the system complexity, real-time realization of a phantom source that covers the whole audio frequency range is not practical with the current technology, if carried out on a sample-by-sample basis. Therefore, block processing techniques have to be used. In this work, the Block Frequency Domain Adaptive Filter (BFDAF) [4] has been utilized. In this section, the BFDAF will be described to introduce the notation and in subsequent sections, the filtered-x part will be discussed in more detail.
Fig. 3 shows the BFDAF of the single point model shown in Fig. 2. This figure is almost equivalent to the original BFDAF implementation. The only difference is that the input signal is filtered in the box 'filtered-x', before it enters the update part of the algorithm. Working on a block basis is reflected in the fact that the whole implementation is performed on blocks of data rather than on samples. Each block contains L new samples, with $L \geq 1$. The serial input signal samples are converted to parallel blocks of samples (box 'S/P') while the output samples are converted from parallel blocks to serial samples (box 'P/S'). A direct consequence of this block processing approach is a processing delay — it takes L samples before the first block of $\hat{e}$ can be produced (the calculation time is not taken into account). This processing delay has to be compensated, for synchronization reasons, in the primary source path (box 'd_L').
For a gradient update algorithm such as the NLMS considered here, two main operations have to be computed [3]:

- a (linear) *convolution* to perform the filtering of the input signal x with the M adaptive weights
- a (linear) *correlation* between the input signal x and the residual signal r.

The first calculates the filter output, and the second calculates an estimate of the gradient $\underline{\nabla}$ that is needed for the update of the adaptive weights.
For large filter lengths M, these operations can be carried out very efficiently by using a block processing approach such as the overlap-save method [5], implemented in the frequency domain. Fast Fourier Transforms (FFT's) are used to perform the signal transformations back and forth between the time and frequency domain. For the necessary convolution and correlation the overlap-save method is used (see Fig. 3).
The *convolution* (performed by the right-hand part of Fig. 3) of the 'infinite' input sequence (the input signal samples $x[k]$) with the 'finite' length sequence (the M adaptive weights) is broken down into a sequence of convolutions between two 'finite' sequences of length B_1. The 'infinite' input sequence is first divided into overlapping blocks. The

overlap of $M-1$ samples is needed because the M adaptive weights have first to be shifted in the sequence of B_1 samples. The convolution in the frequency domain produces a circular convolution result. Therefore the overlap is used to obtain a linear convolution result from this circular one. Together with the fact that every block contains L new samples, the chosen block length equals $B_1 = M+L-1$. As depicted in Fig. 3, the blocks with B_1 input signal samples are transformed to the frequency domain with an FFT of length B_1 ('$\mathbf{F}_{B_1}$') resulting in a block that, for notational reasons, is represented by the transformed input signal vector $\underline{\mathbf{X}}[kL]$. The M adaptive weights are first augmented by $B_1 - M$ zeroes to obtain blocks of length B_1, then transformed to the frequency domain resulting in $\underline{\mathbf{W}}[kL]$. The circular convolution is now performed by element-wise multiplying the two vectors $\underline{\mathbf{X}}[kL]$ and $\underline{\mathbf{W}}[kL]$

$$\hat{\underline{\mathbf{E}}}[kL] = \underline{\mathbf{X}}[kL] \otimes \underline{\mathbf{W}}[kL], \tag{2}$$

where the symbol $\otimes$ is used to denote element-wise multiplication. Only the last L samples (in time) from this circular convolution represent the desired linear convolution result. Therefore, the result $\hat{\underline{\mathbf{E}}}[kL]$ is transformed back to the time domain by an inverse FFT ($\mathbf{F}_{B_1}^{-1}$) and the L correct samples are sent to the secondary loudspeaker after going through the parallel to serial operation.

The *correlation* operation between the 'infinite' sequence of input signal samples x and the 'finite' sequence of residual signal samples r is performed by the left-hand part of Fig. 3. The only difference between the correlation and convolution operations is the mirroring in the time domain. This mirror operation can be performed in the frequency domain by simply applying a complex conjugate ('*') operator resulting in the vector $\underline{\mathbf{X}}_f^*[(k-1)L]$ (note that the input signal to the correlation operation, here the filtered-x signal $\underline{\mathbf{X}}_f[kL]$, needs to be delayed by d_L for synchronization reasons and the residual signal $\underline{\mathbf{R}}[(k-1)L]$ contains the same delay d_L). At the same time, L consecutive samples of the residual signal are collected, extended with zeros and transformed to the frequency domain resulting in the vector $\underline{\mathbf{R}}[(k-1)L]$. The scaled gradient is now calculated by

$$\underline{\nabla}[(k-1)L] = \frac{2\alpha}{L\sigma_x^2} \cdot \left(\underline{\mathbf{X}}_f^*[(k-1)L] \otimes \underline{\mathbf{R}}[(k-1)L]\right),$$

in which the scaling is done by the factor $\frac{2\alpha}{L\sigma_x^2}$. Since the correlation is performed in the frequency domain, the result is also circular and the gradient vector $\underline{\nabla}[(k-1)L]$ has to be transformed back to the time domain to select the M correct gradient coefficients (gradient constraint), as done in the convolution operation. These gradient coefficients are then augmented by zeros and the resulting vector of length B_1 is transformed to the frequency domain and used to update the adaptive weights.

1.2 Filtered-x in frequency domain

The 'filtered-x' operation, that is needed for the update algorithm in Fig. 3, is depicted in more detail in Fig. 4. The function of this 'filtered-x' block is to filter the input signal $x[k]$ with, an estimate of, the A weights of the secondary path $\underline{\hat{\mathbf{h}}}_s$. This convolution is performed in the frequency domain using the overlap-save method as explained before. Each block contains L new input signal samples while the segments have an overlap of $A-1$ samples. The resulting segments contain $B_2 = A+L-1$ input signal samples. These are transformed to the frequency domain ($\underline{\mathbf{X}}[kL]$) and the convolution is performed by the element-wise vector multiplication with $\underline{\hat{\mathbf{H}}}_s$ which is obtained by extending the impulse response $\underline{\hat{\mathbf{h}}}_s$ with B_2-A zeros. The correct L samples (of the circular convolution) have to be appended in the time domain to $M-1$ previously calculated samples to form segments of length B_1. These are transformed to the frequency domain and result in $\underline{\mathbf{X}}_f[kL]$ which is used in the correlation operation described in the previous section.
From Fig. 4 it follows that the 'filtered-x' operation itself needs four FFT's. In the following section we will show that these four FFT's can be subsequently reduced down to one only.

2 Efficient filtered-x implementations

2.1 Exact filtered-x implementations

The reduction from four FFT's to one is done in three steps [6]. The first is a reduction to three FFT's obtained by combining more carefully the FFT's in Fig. 3 and Fig. 4. The next reduction to two FFT's stems from the fact that the convolution operation for calculating the filtered-x input signal and the correlation operation needed for the gradient estimate, can be interchanged. The last reduction to one FFT is obtained by combining again FFT operations.

Step 1: Combining two FFT's

The first optimization step is to combine the two separate Fourier transforms that are used to calculate the transformed input signal vector $\underline{\mathbf{X}}[kL]$ of length B_1 in Fig. 3 and length B_2 in Fig. 4. The FFT $\mathbf{F}_{B_2}$ in Fig. 4 can then be left out. The result is depicted in Fig. 5. Here we have assumed that the length A of the secondary path is smaller than the length M of the adaptive weight vector $\underline{\mathbf{W}}$ ($A < M$). By augmenting the impulse response vector $\underline{\hat{\mathbf{h}}}_s$ of the secondary path with more zeros than before, a length B_1 vector is obtained that is transformed to the frequency domain resulting in $\underline{\hat{\mathbf{H}}}_s$. This vector is multiplied element-wise by the transformed input signal vector $\underline{\mathbf{X}}[kL]$ that was available in Fig. 3. The result is one FFT less in comparison to the type 1 implementation as shown in Fig. 4.

Step 2: Interchanging convolution/correlation

The second optimization step is to interchange the order of the convolution operation of the ‘filtered-x’ and the correlation operations. In Fig. 3, the overlap-save method has to be used in the filtered-x block since the input signal is infinitely long. Because of the circular nature of this convolution, the correct samples have to be selected in the time domain costing one inverse FFT and one forward FFT as shown in Fig. 4 and Fig. 5. Noting that both the impulse response (A) and the residual signal block (L) are of ‘finite’ length, it is more efficient to first calculate the convolution of these two ’finite’ sequences. This will result in a ’finite’ sequence of length $A + L - 1$. Implementing this operation in the frequency domain, with Fourier transforms of appropriate length, results in a frequency domain vector that contains no circular (incorrect) results. This frequency domain vector can now be used straight on in the correlation with the ’infinite’ input signal as before. The result of this approach is depicted in Fig. 6. This realization can also be referred to as ’filtered-r’ because the ’filtered-x’ operation is in fact calculated by filtering first the residual signal (‘filtered-r’). No extra inverse Fourier transform is needed, as in Fig. 5. We only have to take care of a correct Fourier transform length $B_3 = M + L + A - 2$. We note that only two FFT’s are needed in comparison with Fig. 4 and Fig. 5.

Step3: Combining two FFT’s

Finally an equivalent step as described in step 2 can be used here. The Fourier transform $\mathbf{F}_{B_3}$ that is needed in Fig. 6 to calculate the transformed input signal vector $\underline{\mathbf{X}}[kL]$ can be combined with the Fourier transform $\mathbf{F}_{B_1}$ that is needed at the right hand side of the same figure for the convolution with the adaptive weight vector. The result is depicted in Fig. 7. From this figure it follows that for each block, the output of the adaptive filter produces $A + L - 1$ correct results from which only L samples are used and $A - 1$ are not. However the total result is that we have introduced an efficient implementation of the ‘filtered-x’ (or ‘filtered-r’) algorithm in the frequency domain that needs an extra FFT compared to the original BFDAF.

2.2 Approximate filtered-x implementations

In situations where large filter lengths are desired (say > 2048 coefficients) in combination with high-sampling frequencies (up to 20 kHz), the capabilities of the system (such as calculation speed and available memory) can limit the performance of the filter. Reduction of the complexity, as discussed in the previous section, is a solution to this problem. If the reduction is insufficient, a solution can be the implementation of filtered-x approximations as will be discussed in the next sections.

Unconstrained filtering

The number of FFT's can be reduced by two when the gradient constraint (marked in Fig. 7 as a chain dotted box) is left out. This unconstrained filtered-x algorithm has to update B_3 coefficients. As discussed in [7], this leads to a reduction of the stable step size range (in Fig. 7, the reduction is a factor $\frac{B_3}{M}$). Also, the unconstrained filtering increases the filter length but not the order of the filter due to the increase in the error of the output signal when the impulse response of the system is larger than the filter can estimate. In order to keep this so-called *wrap-around* error to a minimum, the length of the adaptive filter must be chosen such that the system's impulse response is smaller than M.

Reduced FFT length

The type 4 implementation has six FFT's of length $B_3 = M+L+A-2$. A decrease of B_3 reduces the complexity [6], however a number of weight coefficients are then updated with values calculated out of circular terms. The error encountered in the weight coefficients is less than one would expect due to the following reasons:

1. the error caused by the circular terms is averaged during the calculation of the convolution or correlation

2. the amplitude of the secondary impulse response decreases rapidly.

The first point is an effect of the circular convolution. When the input signal is convolved with the secondary impulse response, B_3-A+1 linear convolution samples are calculated. The next filtered-x sample that is calculated, has one circular term in the summation and is therefore averaged with the other terms. The last sample to calculate has only one linear term in the summation, so the error in one block of filtered-x samples increases towards the last sample. The same effect appears after the correlation of the filtered-x samples with the residual signal, resulting in M correct weight coefficients and an increase in error towards the last coefficient. If B_3 is reduced, a number of coefficients with a small error will be used as filter coefficients.
The second point is characteristic to an acoustic impulse response. If a proper estimation is used for the secondary path, a relative large proportion of the impulse response has a small amplitude compared to the front of the impulse response. This means that the error caused by the circular terms is relatively small.
The reduction we used for this implementation is $A-1$, so B_3 becomes equal to B_1 and will be referred to as type 5. Despite the fact that the FFT length is reduced by $A-1$, this filter is a good approximation of the exact filtered-x implementation.
Instead of decreasing the complexity of the filter, the efficiency of the implementations can be improved by using an unconstrained filter, as is discussed below.

2.3 Efficient use of system resources

If an unconstrained filter is used as described before, a wrap-around error will be generated when the impulse response of the system is too long to fit in the M weight coefficients. When no constrained filter can be used (due to limitations of the system), two solutions can be applied if the filter coefficients are to be fixed after cancellation:

1. apply the gradient constraint on the weight coefficients after adaptation,
2. if the filter coefficients will be frozen after adaptation, it is possible to use all weight coefficients of an unconstrained filter by doubling the FFT length B after adaptation so that no wrap-around error is generated.

The first solution removes the wrap-around error but generates an error due to the removal of a part of the weight coefficients. This error appeared to be even larger than the wrap-around error when both are averaged. However, the wrap-around error is present for a small part of the output samples which causes a rattling sound (at the frequency of the block processing) that is not acceptable in sound processing or noise cancellation applications. With the second solution however, it is possible to use all weight coefficients while maintaining the same complexity and memory use [6]. This can be done by doubling the filter length after adaptation and thereby removing the wrap-around error. This means that the wrap-around error is present during the estimation of the filter coefficients but will be removed after cancellation. What remains is the error caused by the weight coefficients that are calculated out of circular terms. The resulting filter provides again a good approximation of the filtered-x algorithm.

3 Complexity comparison

In the previous section, the number of FFT's used in the exact filtered-x BFDAF implementations is reduced from nine to six. However, the FFT lengths are not equal and depend on the parameters A, L and M. Therefore it is difficult to compare the complexity of the different types just by looking at the number of FFT's. Fig. 8 shows the complexity of the four exact implementations as described above. The complexity is approximated by the total number of multiplications needed for the computation of one output sample [6]. This stems from the used FFT's of length B, each having a complexity of $(B \cdot \log_2(\frac{B}{8}) + 4)$ (for real input (FFT) and real output (IFFT)), the element-wise multiplications $(2B - 2)$ and multiplications by a real scalar (B). From this comparison it follows that, for the chosen parameter set, type 2 has the lowest complexity when $M < 276$ (curve 2) and type 4 when $M > 276$ (curve 4). Thus for applications where A is relatively small compared to the adaptive filter length M, type 4 is preferred. However when A is larger than say half the filter length, type 2 is preferred.

Fig. 9 shows, for the same parameter set, the computational complexity of the approximated filtered-x implementations. Curves 1 and 2 show the complexity of the unconstrained filters type 2 and 4 respectively. Curve 1 also represents the constrained filter type 5 and curve 3 the unconstrained filter type 5 (type 4 with a reduced filter length of B_1). Curve 4 shows the complexity of the unconstrained filter type 5 with a doubling of the FFT length after cancellation.

4 Experimental results

Experiments have been performed [6] on a one point noise cancellation system at a sampling frequency of 32 kHz and white noise as input signal during the cancellation process. The first set-up for this system consists of a primary loudspeaker, a secondary loudspeaker and a microphone, all placed in an anechoic room. The angles are $\alpha = 0°$ and $\beta \approx 35°$. The estimated impulse response $\underline{\hat{\mathbf{h}}}_s$ is fixed during the noise cancellation.

Fig. 10 shows the measured power spectra of the residual signal. Curve 1 is the spectrum of the background noise. Curve 2 is the spectrum of the primary loudspeaker. Curve 3 and 4 are the spectra of the residual signal after cancellation with $L = \frac{B}{2}$ and B is 512 and 2048 respectively. The average attenuation over a frequency range from 1 to 15 kHz is 29 dB for $B = 512$ and 36 dB for $B = 2048$.

With a second set-up, measurements are conducted to compare the different filtered-x implementations. This set-up consists of a pair of headphones (as secondary sound sources), a primary loudspeaker, and a dummy torso, all placed in a listening room. The primary loudspeaker is located at $\beta = 45°$ and a distance of about 1.4 m with respect to the dummy and the headphones are placed on the dummy-head. Fig. 11 shows the difference in attenuation of the residual signal for the constrained (upper curve) and unconstrained (lower curve) filtered-x implementations. The residual signal is best attenuated by the unconstrained filter because the wrap-around error is less than the error caused by the gradient constraint (the impulse response of the system is too large to fit in the M weight coefficients). The wrap-around error however, causes a rattling sound at the frequency of the block processing. Fig. 12 shows the difference in attenuation of the residual signal between a filtered-x implementation of Section 2.1 (lower curve) and the implementation with reduced FFT length (upper curve), both constrained. The reduction of the complexity leads to a decrease in adaptation speed and less attenuation of the residual signal. Fig. 13 shows the attenuation of the residual signal of the unconstrained exact filtered-x implementation (upper curve). The lower curve shows the remaining residual signal after the filter length has been doubled. This filter has the reduced complexity of an unconstrained filter but has no wrap-around error. However, the filter coefficients have to be fixed after cancellation.

5 Conclusions

Generation of a virtual sound image using adaptive filters that employ the filtered-x LMS algorithm has been discussed. Such complex filters are best implemented in real-time using block processing techniques in a unitary transformation domain, such as the Fourier domain. Several implementations of the filtered-x Block Frequency Domain Adaptive Filter (BFDAF) are discussed and their complexities are compared. Simulations and experiments have shown that of the exact implementations, type 2 is the most efficient one when the length of the secondary source path estimate is longer than half the length of the adaptive filter. When the adaptive filter length is much longer than the secondary path estimate, type 4 has shown to be the most efficient one.
When system limits are reached, one could choose to implement a filtered-x approximation by reducing the FFT length or using an unconstrained filter. If the filter coefficients are frozen after adaptation, it is possible to use all weight coefficients of an unconstrained filter by doubling the FFT length after adaptation so that no wrap-around error is generated.
A trade-off between the various FFT lengths is possible which enhances the flexibility considerably.

References

[1] R.M. Aarts, *On the design and psychophysical assessment of loudspeaker systems*, Ph.D. thesis Delft University of Technology, The Netherlands, September 1995.

[2] R.M. Aarts, H. He, P.C.W. Sommen, *Phantom sources applied to stereo-base widening using long frequency domain adaptive filters*, Proceedings of the ProRISC/IEEE workshop 1996, pp. 49-54.

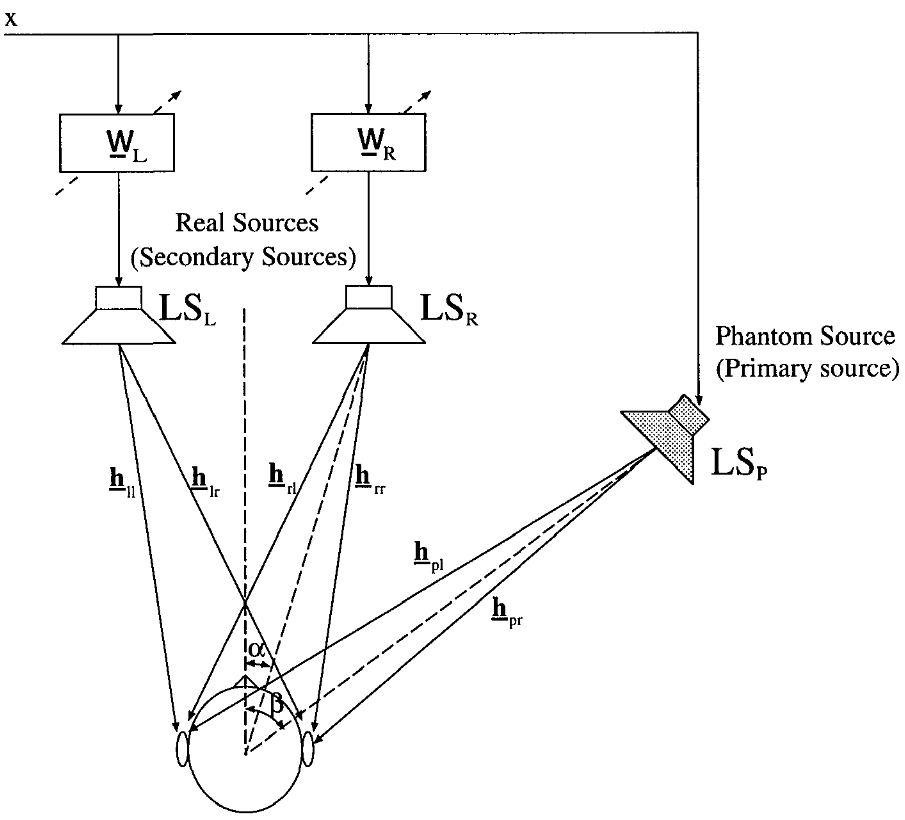
[3] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

[4] P.C.W. Sommen, *Adaptive Filtering Methods*, Ph.D. thesis Eindhoven University of Technology, The Netherlands, June 1992, ISBN 90-9005143-0.

[5] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1975.

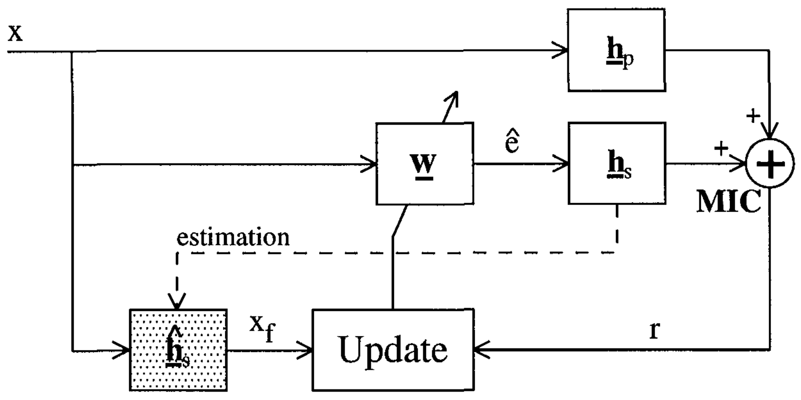
[6] A.W.M. Mathijssen, *Generation of phantom sound sources with Block Frequency Domain Adaptive Filtering*, Master thesis Eindhoven University of Technology, September 1997.

[7] X. Li, W.K. Jenkins, "The Comparison of the Constrained and Unconstrained Frequency-Domain Block-LMS Adaptive Algorithms," *IEEE Transactions on Signal Processing*, vol. 44, no. 7, July 1996, pp. 1813-1816.



x
$\underline{W}_L$
$\underline{W}_R$
Real Sources
(Secondary Sources)
LS_L
LS_R
Phantom Source
(Primary source)
LS_P
$\underline{\mathbf{h}}_{ll}$
$\underline{\mathbf{h}}_{lr}$
$\underline{\mathbf{h}}_{rl}$
$\underline{\mathbf{h}}_{rr}$
$\underline{\mathbf{h}}_{pl}$
$\underline{\mathbf{h}}_{pr}$
α
β

Figure 1: *Set-up for phantom sound source generation.*



x
$\underline{\mathbf{h}}_p$
$\underline{\mathbf{w}}$
ê
$\underline{\mathbf{h}}_s$
+
+
MIC
estimation
$\hat{\underline{\mathbf{h}}}_s$
x_f
Update
r

Figure 2: *Single point system model.*

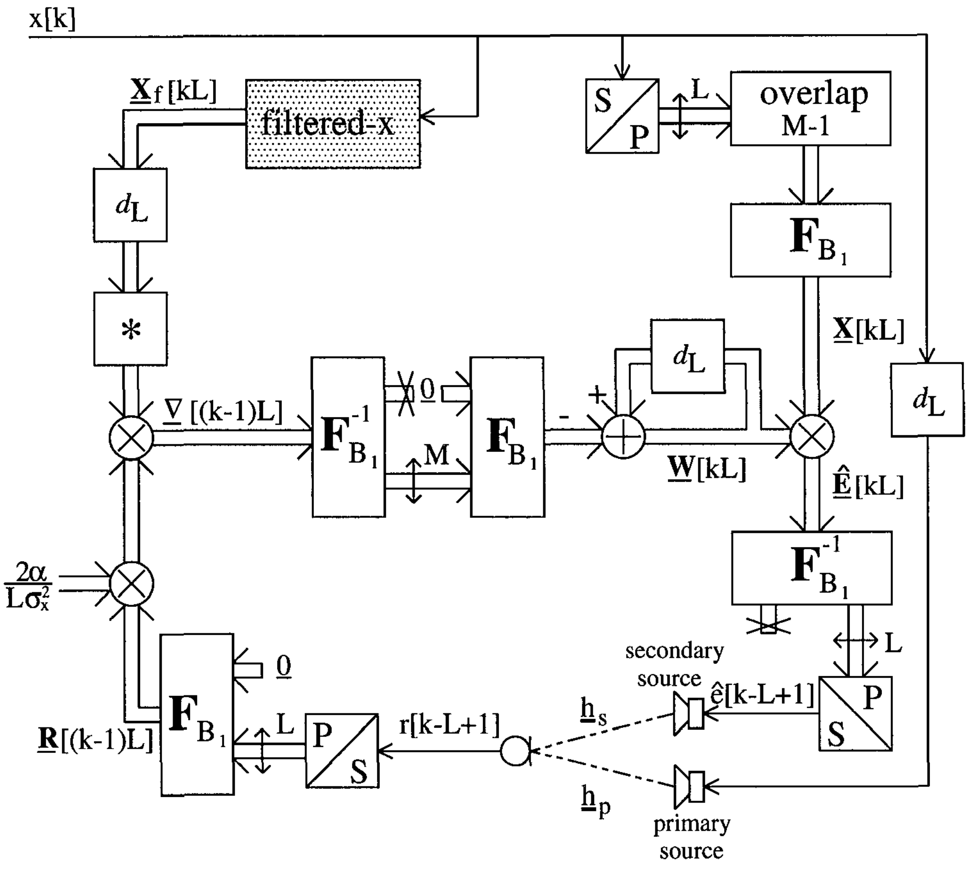


Figure 3: *Block Frequency Domain Adaptive Filter.*

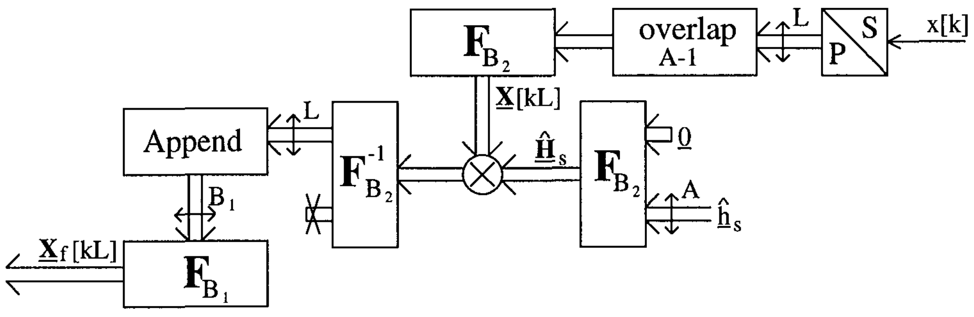


Figure 4: *Filtered-x implementation type 1.*

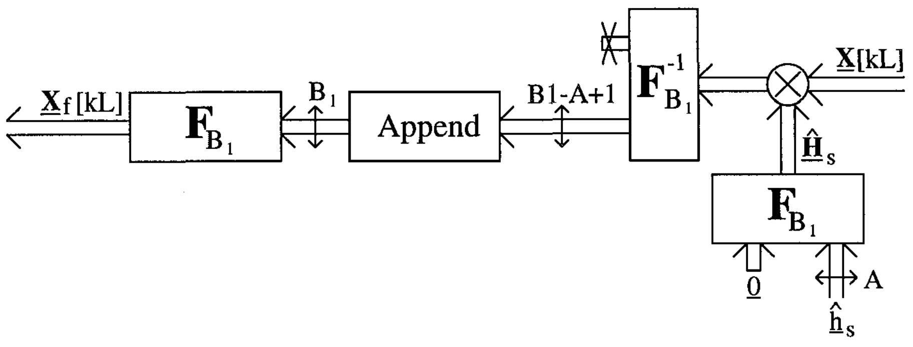


Figure 5: *Filtered-x implementation type 2.*

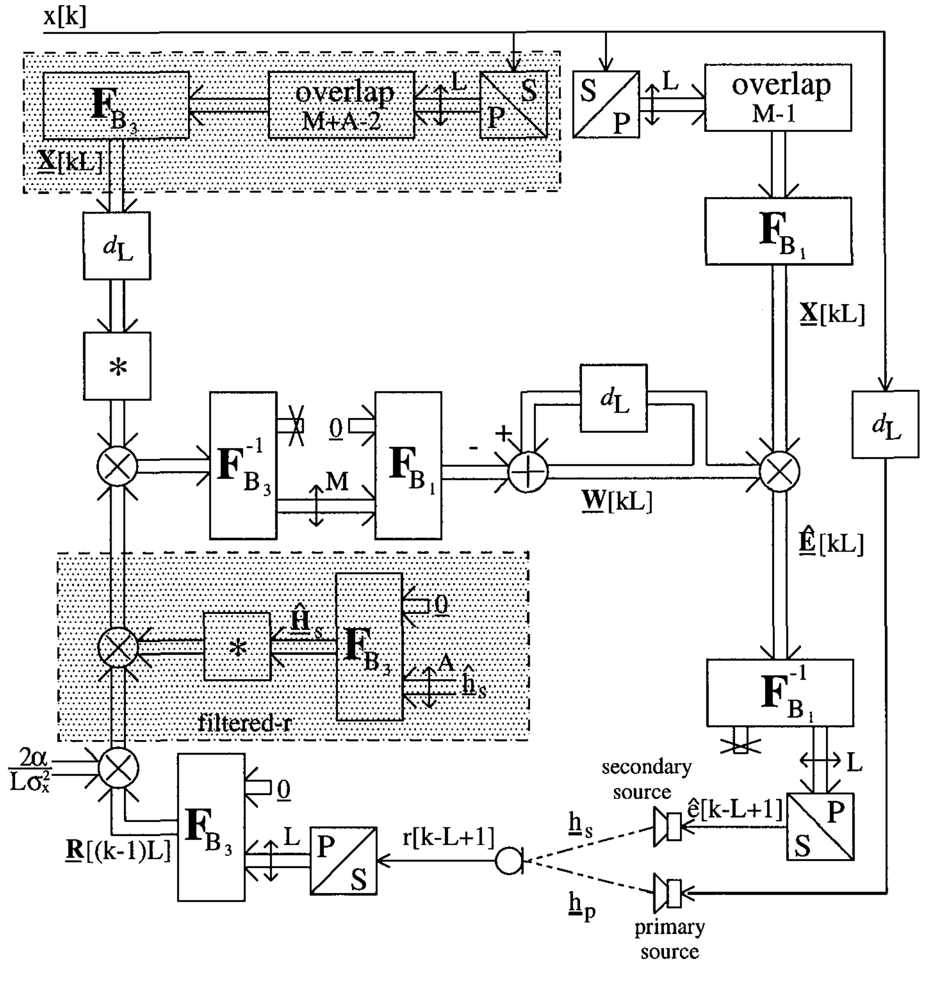


Figure 6: *Filtered-x implementation type 3.*

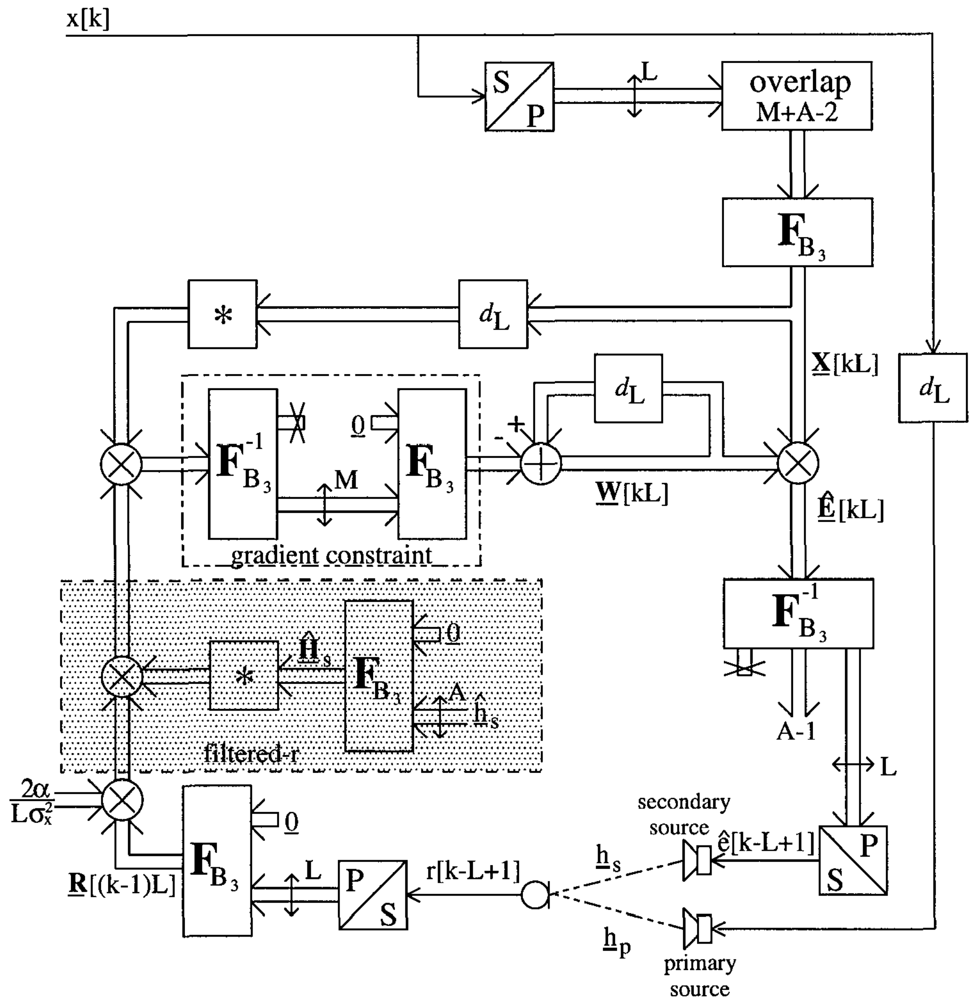


Figure 7: *Filtered-x implementation type 4.*

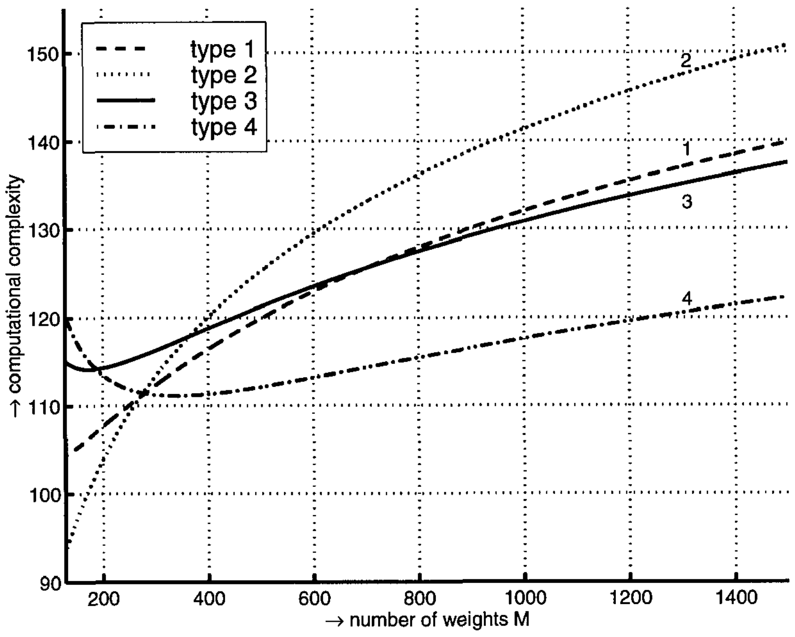


Figure 8: *Computational complexity of the exact filtered-x implementations, from Section 2.1 (A =128, L = M − 1).*

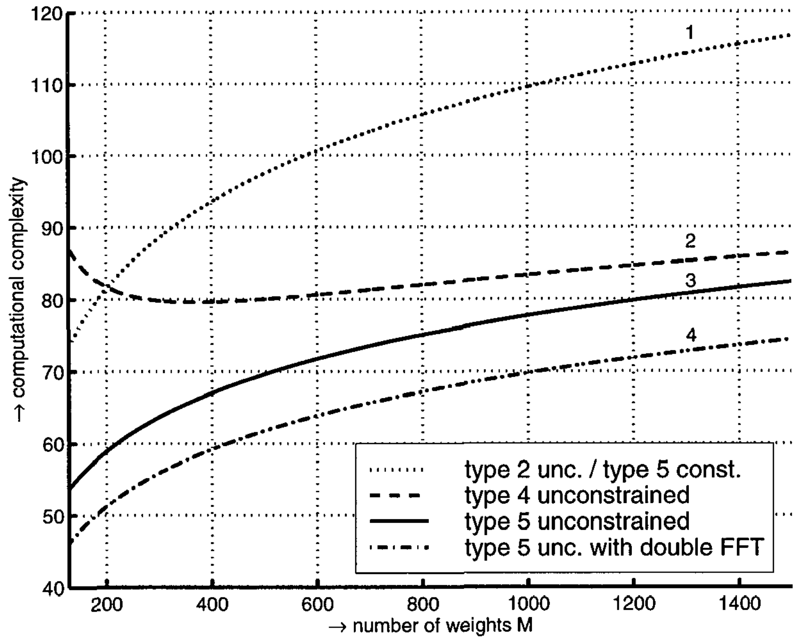


Figure 9: *Computational complexity of the approximated filtered-x implementations (A =128, L = M − 1).*

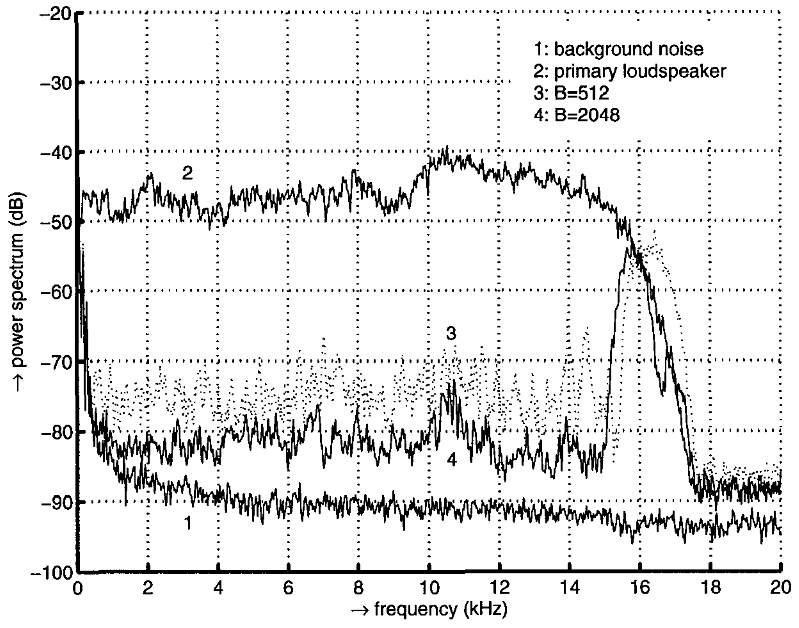


Figure 10: *Power spectrum of the background noise (curve 1), the residual signal before cancellation (curve 2) and after cancellation (curve 3, B=512 and curve 4, B=2048).*

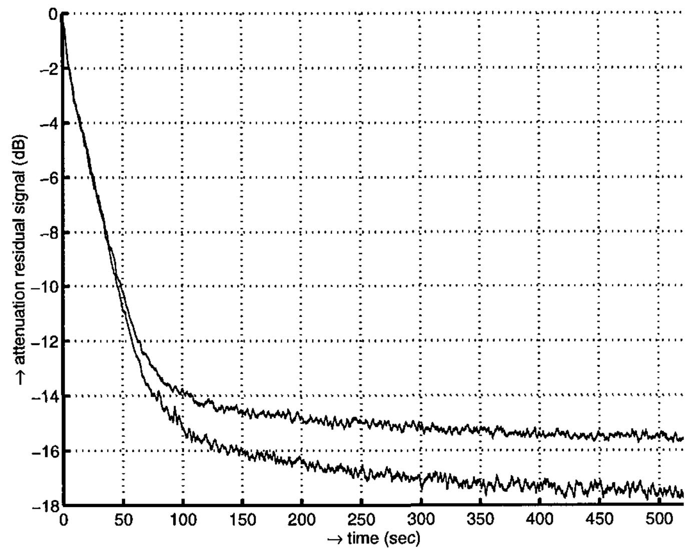


Figure 11: *Attenuation of the residual signal during cancellation for the constrained (upper curve) and the unconstrained (lower curve) exact filtered-x implementations from Section 2.1 measured in the experimental room (*B_1*=2048, L=1024, M=1025).*

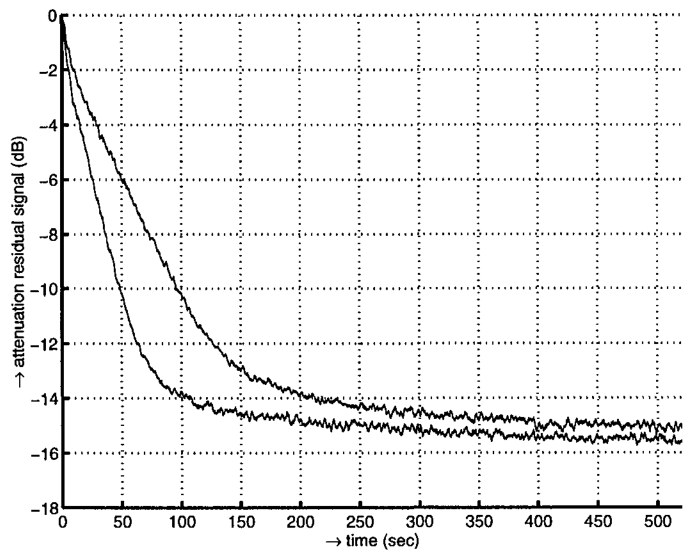


Figure 12: *Attenuation of the residual signal during cancellation for the constrained filtered-x implementation (lower curve) and the constrained filtered-x implementation with reduced FFT length (type 5)(upper curve) in the experimental room.*

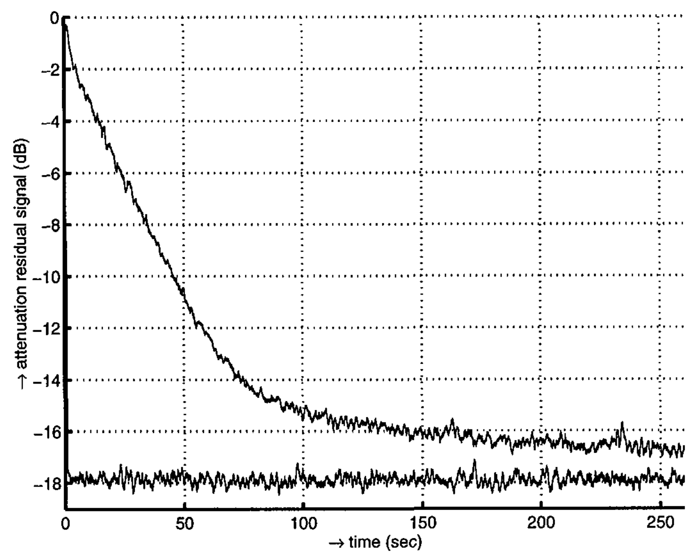


Figure 13: *Attenuation of the residual signal for the exact unconstrained filtered-x implementation during adaptation (upper curve). The lower curve represents the final remaining residual signal after the filter was doubled in length and has been converged.*