FREQUENCY DOMAIN EXPLOITS FOR SYMMETRIC ADAPTIVE DECORRELATION

A thesis presented in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in

Engineering

at Massey University, Albany, New Zealand.

Jonathan Harris

2014

ABSTRACT

Symmetric adaptive decorrelation (SAD) is a semi-blind method of separating convolutely mixed signals. While it has restrictions on the physical layout of the demixing equipment, restrictions not present for many other *blind source separation* (BSS) techniques, it is more ideally suited for some applications (for example, live sound mixing) due to the fact that no post-processing is required to ascertain which *output* corresponds with which *source*. Since the SAD algorithm is based on second-order statistics (SOS), it also tends to have a lower computational load when compared with those based on *higher order statistics*. In order to increase the efficiency of the SAD algorithm, a multibranched recursive structure is investigated. While a nominal gain in efficiency *is* attained, we move away from this approach in pursuit of more substantial gains. A *frequency-domain symmetric adaptive decorrelation* (FD-SAD) algorithm is proposed, with savings increasing not only with larger filter sizes, but also with increasing the number of sources. The convergence and stability of this new algorithm is proven. A trade-off of the FD-SAD algorithm is that it introduces a delay in the output, which renders the algorithm unsuitable for real-time applications. Therefore a hybrid approach is also proposed that does not suffer from the lag of the frequency domain approach. While the proposed algorithm *is* slightly less computationally efficient than the pure frequency domain algorithm, it is significantly more efficient than the time-domain approach. A comparison of the frequency domain and hybrid algorithms shows that both achieve separation equivalent to the time-domain algorithm in a real-world environment. The proposed adaptations could also be used to extend other BSS approaches (such as *Triple-N ICA for Convolutive mixtures* (TRINICON) [1], which can also be based on SOS), and a comparison of the proposed methods with TRINICON is explored.

ACKNOWLEDGEMENTS

I would firstly like to thank my supervisors, Fakhrul Alam and Tom Moir for their help in guiding me through the problems that inevitably arose.

I would also like to thank the committee who reviewed my work and gave insights into areas I had overlooked, along with valuable recommendations on what needed to be done to overcome these problems.

And finally, I would like to thank my wife, whose support and encouragement gave me the strength to persevere through both the tough and the tedious.

CONTENTS

1. INTRODUCTION

The *blind source separation* (BSS) problem occurs when a desired signal (e.g. speech) is corrupted by some undesired signal, and there is no information regarding the source positions, probability density functions or frequency spectra.

When the undesired signal is known a priori, there exists a number of suitable *supervised* algorithms that can remove the noise from the mixture. For example, both the *least mean squares* (LMS) [2] and *recursive least squares* (RLS) [3] algorithms operate under the assumption that the undesired signal is known. These supervised algorithms use two microphones (in the two source case): one for the mixture, and the other to pick up a noise estimate. They then proceed to remove any signal from the mixture that is correlated to the noise estimate. However, in real environments, these approaches tend to degenerate due to the presence of crosstalk from the desired signal in the noise estimate. In order for the noise estimate to be signal free, the noise receiver has to be located a reasonable distance from the desired source, yet the farther apart the two receivers are, the more the noise estimate deviates from the noise component of the mixed signal.

BSS is the act of separating signals with the sole assumption that the sources are independent. This assumption widens the scope of potential applications when compared to previous techniques because information on the geometry or spectral density is no longer needed. This thesis covers a myriad of different techniques and explains the problem, the use of second-order and higher-order statistics, demixing system structures, measures of independence between signal *probability density function*s (PDFs), various optimization techniques, and comparisons between the use of time and frequency domains in separation.

There are an extensive number potential applications for an algorithm that can successfully separate multiple mixed sources from their mixture. While the main focus of this thesis is towards mixtures of audio signals (specifically speech) [4, 5], such an algorithm would also have potential applications in telecommunications [6, 7], underwater sonar [8, 9], the medical industry [10, 11], etc.

A number of aspects of the problem complicate it further, especially for audio applications. In a real-room environment, the mixture is not just a simple mixture of two signals. The reverberations from the walls in the room will add to the signals, meaning that the source components at each microphones do not identically match across the microphones, but are filtered versions of each other. Furthermore, these filters may be *non-minimum phase*, so any inverse of the filters will be unstable [12]. The presence of additive noise may further corrupt the mixtures. These are all problems that need to be overcome in order to implement a successful algorithm [9].

In order to attain a higher quality signal, there is a need for an algorithm that can separate a mixture of signals into its original components. For audio applications, such a method of separation needs to work well in highly reverberant environments. If it needs to be implemented as a real-time process, another highly attractive feature is if the algorithm is computationally efficient.

The work described in this thesis focuses on the approach known as *symmetric adaptive* *decorrelation* (SAD) [13], which is a semi-blind approach based on second-order statistics (SOS). Adaptations to the base algorithm are proposed which increase efficiency significantly for larger filter sizes, with little impact on separation performance.

In this work, the following contributions are made

- A detailed derivation of the *multiple-input multiple-output* (MIMO) SAD algorithm
- Introduction of a multibranched recursive adaptation to the SAD algorithm, increasing efficiency.
- Proposal of the *frequency-domain symmetric adaptive decorrelation* (FD-SAD) algorithm with an analysis of its convergence
- Development of a hybrid SAD algorithm that substantially increases efficiency without introducing an input-output delay

The structure of the thesis is as follows. Chapter 2 contains a review of the literature relating to the separation of mixed sources. Chapter 3 shows the derivation and analysis of the *time-domain symmetric adaptive decorrelation* (TD-SAD) algorithm for MIMO situations, and chapter 4 uses a multibranched approach to increase efficiency. While it does increase efficiency, these increases are not overly substantial, so chapter 5 extends the SAD algorithm to the frequency domain. Chapter 6 outlines a hybrid approach that does not introduce an input-output delay as with the FD-SAD, but is significantly more computationally efficient than the TD-SAD.

For the sake of conciseness of this work, only the key steps in derivations are included in its body. Extended derivations can be found in Appendix B.

2. LITERATURE REVIEW

2.1 Background Information

The problem of signals becoming corrupted by noise is inherent to many different methods of measuring some unknown signal. This corruption will often have very adverse effects on the quality of the *observed signal*, and if significant enough, may render it totally useless. There has been much research done on ways of increasing the quality of a signal through the use of either physical methods, or by post-processing. This review is focused on the use of a class of signal-processing techniques to increase the *signal to noise ratio* (SNR) of the mixture, with a specific emphasis on audio signals.

One of the main situations in which the noise is produced occurs when there are multiple sources, of which only one is desired. For example, when a number of people are in a room, all talking simultaneously, the human brain is able to process the speech coming from a specific person. This is known as the *cocktail party problem*, and is yet to be adequately solved using state-of-the-art techniques. However, much research has been done over the past two decades in this area, with gradual progress.

A tool that is used extensively in the literature to combat this problem is the use of multiple receivers. Girolami [14] notes how for humans, binaural hearing has advantages over monaural hearing not only in the removal of unwanted noise, but also in dereverberation of the desired signal. This tends to indicate that the human hearing system incorporates

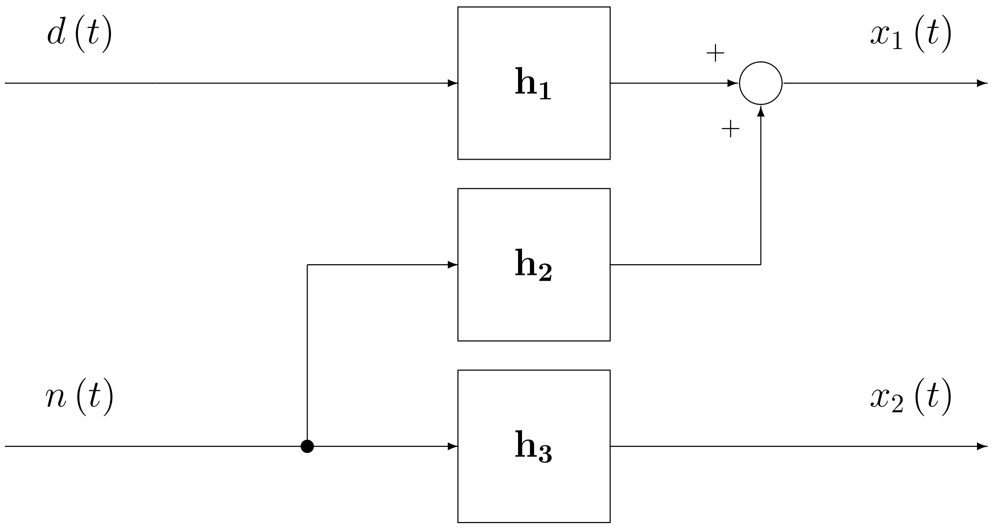


Fig. 2.1: Simple Mixing System

the use of both ears not only when trying to increase the SNR of desired audio sources to background noise, but also to reduce the detrimental effects due to reverberations from the walls in a room. Similarly, many algorithms show that the use of multiple receivers can significantly improve the signal-to-noise ratio.

Considering the simple system shown in figure 2.1,

$$\begin{aligned} x_1(t) &= \sum_{i=0}^{L-1} h_{1,i} d(t-i) + h_{2,i} n(t-i) \\ x_2(t) &= \sum_{i=0}^{L-1} h_{3,i} n(t-i) \end{aligned} \tag{2.1}$$

where d is the desired signal, n is the noise, $\mathbf{h}_1$, $\mathbf{h}_2$, $\mathbf{h}_3$ are the channel filter vectors defined by

$$\mathbf{h}_j = \begin{bmatrix} h_{j,0} & h_{j,1} & \dots & h_{j,L-1} \end{bmatrix}^T,$$

L is the number of taps for each of the channel filters, d is the desired signal, n is the noise signal, x_1 and x_2 are the mixture and noise estimate respectively, and the superscript ^{T}

denotes the transpose operation.

In the Z-*domain*, the filters $\mathbf{h}_j$ are represented by their Z-domain equivalents $H_j(z)$, and the *outputs* $x_1(t)$ and $x_2(t)$ from (2.1) become

$$\begin{aligned} X_1(z) &= H_1(z)\,D(z) + H_2(z)\,N(z) \\ X_2(z) &= H_3(z)\,N(z) \end{aligned} \tag{2.2}$$

Because $X_1(z)$ and $X_2(z)$ are known, if $H_2(z)/H_3(z)$ can be found, then it is possible to remove any signal correlated to the noise from $X_1(z)$, thereby resulting in a filtered version of the desired signal.

$$\begin{aligned} Y(z) &= X_1(z) - \frac{H_2(z)}{H_3(z)} X_2(z) \\ &= X_1(z) - \frac{H_2(z)}{H_3(z)} H_3(z)\,N(z) \\ &= H_1(z)\,D(z) + H_2(z)\,N(z) - H_2(z)\,N(z) \\ &= H_1(z)\,D(z) \end{aligned} \tag{2.3}$$

In [2], Widrow *et al.* recognized this and developed the *least mean squares* (LMS) algorithm. The LMS algorithm uses an adaptive filter to remove any signal from $x_1(t)$ that is correlated to the signal $x_2(t)$. Once the filter has converged, and under the assumption that there are no other noise components, the output will be the filtered version of the desired signal $d(t)$.

2.1.1 The Least Mean Squares Filter

The LMS filter operates by finding a vector $\mathbf{w}(t) = \begin{bmatrix} w_0(t) & w_1(t) & \ldots & w_{L-1}(t) \end{bmatrix}$ that minimizes the power of the error given by

$$E(z) = X_1(z) - W\left(z^{-1}, t\right) X_2(z) \tag{2.4}$$

where $W(z,t)$ is the adaptive filter of length L, defined by

$$W(z,t) = \sum_{i=0}^{L-1} w_i(t)\, z.$$

Because $x_2(t)$ only includes the noise term, and it is assumed that the desired signal is uncorrelated to the noise signal, the error will be reduced such that all components of $x_1(t)$ correlated to the noise signal are removed. The effect of this is that $W(z^{-1}, t)$ converges to $H_2(z)/H_3(z)$, thereby making the error approach y defined in Equation (2.3). The full LMS algorithm for the update of the filter vector is given by the following equations

$$\begin{aligned} \mathbf{w}(t+1) &= \mathbf{w}(t) + \mu e(t)\, \mathbf{x}_2(t) \\ e(t) &= x_1(t) - \mathbf{w}^T(t)\, \mathbf{x}_2(t) \end{aligned} \tag{2.5}$$

where μ is the step size, and $\mathbf{x}_2(t)$ is the vector defined by

$$\mathbf{x}_2(t) = \begin{bmatrix} x_2(t) & x_2(t-1) & \ldots & x_2(t-L+1) \end{bmatrix}.$$

One problem with the LMS filter is that if the step size is too small, then a low-powered (low variance) $x_2(t)$ signal will increase the convergence time, but if the step-size is too big, a high-powered $x_2(t)$ could potentially cause the algorithm to become unstable. Due to these facts, the step-size is usually chosen to be a overly small value for stability reasons, making the convergence time unnecessarily large.

In order to remedy this problem, Mathews and Xie [15] describe what is known as the *normalized least mean squares* (NLMS) algorithm. The NLMS algorithm modifies the ordinary LMS algorithm by making the step size adaptive. If the power of $x_2(t)$ increases, then the step size is decreased, but if the power of $x_2(t)$ decreases, then the step size is increased. This not only increases the stability of the algorithm, but also reduces the convergence time. While the normalized algorithm uses the same error equation as ordinary LMS (as described in Equation (2.5)), it substitutes the original update equation with the following

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \frac{\mu e(t)\,\mathbf{x}_2(t)}{\mathbf{x}_2^T(t)\,\mathbf{x}_2(t)}. \tag{2.6}$$

This allows the step size to increase with decreases in *source* power, and decrease with increases in source power, meaning that the algorithm is more stable and faster converging than its non-normalizing counterpart. In [16], Makino *et al.* propose an exponentially weighted approximation of the power that increases convergence speed while having no extra computational load.

There have been other adaptations to the LMS algorithm. In [17], Kalluri and Arce propose a class of nonlinear normalized adaptive filters based on LMS and the *Volterra series*, which can be thought of as the Taylor series with memory. For example, for a second order system, a *Volterra filter* has two filter terms: a length-N array for the linear terms and a $N \times N$ matrix for the quadratic terms. For audio applications, this is unnecessary because acoustic reverberation tends to be a *linear* mixture of signals [5], causing the quadratic term in a Volterra filter to simply converge to all-zero matrices.

Vector-LMS

In [18], Batra and Barry show how the LMS algorithm can be adapted to work on a vector of signals rather than just individual signals. This should not be viewed as independent LMS algorithms working in an array. Rather it is the LMS algorithm working on a mixture of signals; it takes into account the crosstalk between signals. Its structure is very similar to an ordinary LMS algorithm, except that instead of having scalar coefficients in the filter, there are matrix coefficients. Benesty *et al.* show a similar result in [19], but also include the *recursive least squares* (RLS) case, and in [20] Douglas shows that although the fundamental behavior of the *vector least mean squares* (VLMS) algorithm is the same as ordinary LMS, its convergence is slower for the amount of information contained in the sequence.

While ordinary LMS will find the transversal filter weights when given both the input and output of a filter, VLMS will find the mixing system given the inputs and outputs of the mixing system. For example, if we applied LMS to *two-input two-output* (TITO) system shown in Fig. 4.1, the matrix-polynomial of the filter would converge to $\mathbf{H}$. [18] shows the derivation of the VLMS algorithm

$$\mathbf{W}\left(t+1\right)=\mathbf{W}\left(t\right)+\mu\mathbf{S}\left(t\right)\mathbf{e}\left(t\right)^{T}$$

where $\mathbf{W}\left(t\right)$ is the estimate at time t of the mixing polynomial matrix $\mathbf{H}$, μ is the step size, $\mathbf{S}\left(t\right)=\left[\begin{array}{cccc}\mathbf{s}^{T}\left(t\right), & \mathbf{s}^{T}\left(t-1\right), & \ldots, & \mathbf{s}^{T}\left(t-L+1\right)\end{array}\right]^{T}$ is a vector of length $2L$ of the inputs where L is the number of taps in the filter, and $\mathbf{e}\left(t\right)$ is a length-2 vector of the errors between the desired filter output $\mathbf{d}\left(t\right)$ and its actual output $\mathbf{x}\left(t\right)$ where $\mathbf{x}\left(t\right)=\mathbf{W}^{T}\left(t\right)\mathbf{S}\left(t\right)$.

Frequency-Domain LMS

An important adaptation to the LMS algorithm is to use the fast Fourier transform (FFT) to perform both the filtering and the update in the frequency domain. The primary motivation for this is that convolution and correlation are computationally expensive operations, but map to simple element-by-element multiplication in the frequency domain. Due to the efficiency of the FFT algorithm, especially for larger filter lengths, there can be sizeable computational savings. In [21], Dentino *et al.* first outline a frequency domain implementation of the LMS algorithm. However, they neglect to account for the fact that multiplication in the frequency domain is equivalent to *circular* convolution and correlation, not the *linear* convolution and correlation that are needed for the LMS algorithm.

Ferrara, in [22], points this out, and shows how zero-padding the blocks prior to frequency domain transformation can avoid the inaccuracies of this oversight. He then compares the computational complexity between the algorithms for a variety of filter sizes and shows that his proposed algorithm's efficiency surpasses that of the time-domain algorithm when filter sizes equal or exceed 64 taps. In [23], Shynk provides an outline of frequency domain processing for any adaptive algorithm, but with a specific emphasis on LMS. He shows both the advantages and the drawbacks of *frequency domain adaptive filter*s (FDAFs); they are much more computationally efficient but have an end-to-end delay and do not track as well as their time-domain counterparts.

Because FDAFs are inherently block processes, the end-to-end delay is dependent on the block size. In general, the block size is chosen to be equal to the filter size L, as this is the most efficient choice [24]. This means that the end-to-end delay is equal to the length of the impulse response of the adaptive filter. In acoustic environments, a room impulse response may be several hundred milliseconds long, requiring the output of the adaptive filter to delay the input by an equal length of time. This delay may be too substantial for comfortable use in many situations.

The adaptive step size of FDAF algorithms also needs to be scaled down by a factor of L [25]. If the spread of the input *autocorrelation matrix* is large, then the block implementations of the adaptive algorithms will converge more slowly than the sample-by-sample implementations.

One significant problem inherent to algorithms such as LMS which are based on the mixing system described in figure 2.1 is that they require a noise estimate free of any crosstalk from the desired signal. While this may be feasible in some situations (such as a jet cockpit, where a signal of speech-free engine noise is readily acquired), in most other applications this is at best impractical if not impossible. The microphones need to be sufficiently far such that $x_2(t)$ does not contain any of the desired signal, yet they also have to be sufficiently close such that the noise signal in each is closely correlated.

In order to solve this problem, Zinser *et al.* in [26] proposed cross-coupling two LMS algorithms in order to simultaneously remove the crosstalk from each. They subsequently showed that their algorithm performed substantially better than the ordinary LMS algorithm. This concept has also been improved upon in the years following.

2.1.2 Blind Source Separation

The problem of crosstalk brought a new direction in the late 1980's, when the concept of *blind source separation* (BSS) was initially investigated [11]. Rather than making assumptions about how the microphones and sources are positioned, BSS only makes the assumption that the sources are independent of each other. This makes it a very powerful tool because in many situations this assumption is true, and it does not need any other knowledge of the sources or receivers in order to separate the signals.

From the start, BSS promised to be a very valuable tool for acquiring a desired signal from a mixture of signals. Previous work was primarily aimed at beamforming, where the signals coming from all but a specific direction were attenuated. However, this was based on knowledge of the source's position relative to the receiver; if this information was unknown, or if the source was moving, beamforming was no longer a plausible solution. BSS could potentially eradicate the need for this additional knowledge.

Because it is a discipline that has been investigated for nearly three decades, in the past decade a number of authors have introduced standardized methods of measuring separation performance. In [27] and [28], Vincent *et al.* propose a unifying measure of separation that incorporates interference, additive noise, and any artifacts that may arise from the separation procedure. Other papers focus on speech intelligibility as the ultimate goal, and attempt to find metrics which are able to closely match the results of experimental human listening tests [29–31].

This focus on standardization has then given rise to methods of better comparing algorithms against each other. Two notable efforts are the *PASCAL CHiME speech separation and recognition challenge* (as outlined in [32]), and *the Signal Separation and Evaluation Campaign* (SiSEC) [33].

Blind source separation has then lead on to other techniques. For example, *informed* source separation is a method of separating sources with a significantly higher quality of output than can be attained with BSS [34]. However, it requires that information is encoded inaudibly within the mixtures, allowing the separator to better estimate the mixing matrices. This means that the mixing system needs to be known *a priori*, which is possible for applications such as artificially mixed music samples, but does not lend itself to naturally mixed signals.

2.1.3 Potential applications

While the main focus of this work is in the area of separating mixed audio sources, there are potential applications for many other fields. Possibly the most significant is in the field of communications. [6,13,18,35–38] are just a few articles that specifically look into this problem from a communications perspective. On a similar thread is [39] which looks at the problem of separating radar signals. A number of papers also mention that source separation techniques can be used for biomedical applications such as *magnetic resonant imaging*s (MRIs), *electoromyogram*s (EMGs) and *electroencephalogram* (EEG) [8, 9, 40–53]. Other papers list potential uses that include sonar [8,9,40,46], image enhancement [8,9,40,41,44,52,54], reduction of crosstalk between channels in twisted-pair bundesles or between tracks in magnetic recordings [18], or machine vibrational analysis [10, 55]. Some even more unique applications are shown in [56], where BSS is used to analyze the radiated spectrum from exoplanetary systems, or [57], where videos of peoples faces are processed in an attempt to provide a non-contact method of discerning their pulse rate. However, the majority of papers covered in this review pertain specifically to speech processing [5, 14, 26–34, 42, 45, 46, 53, 58–80].

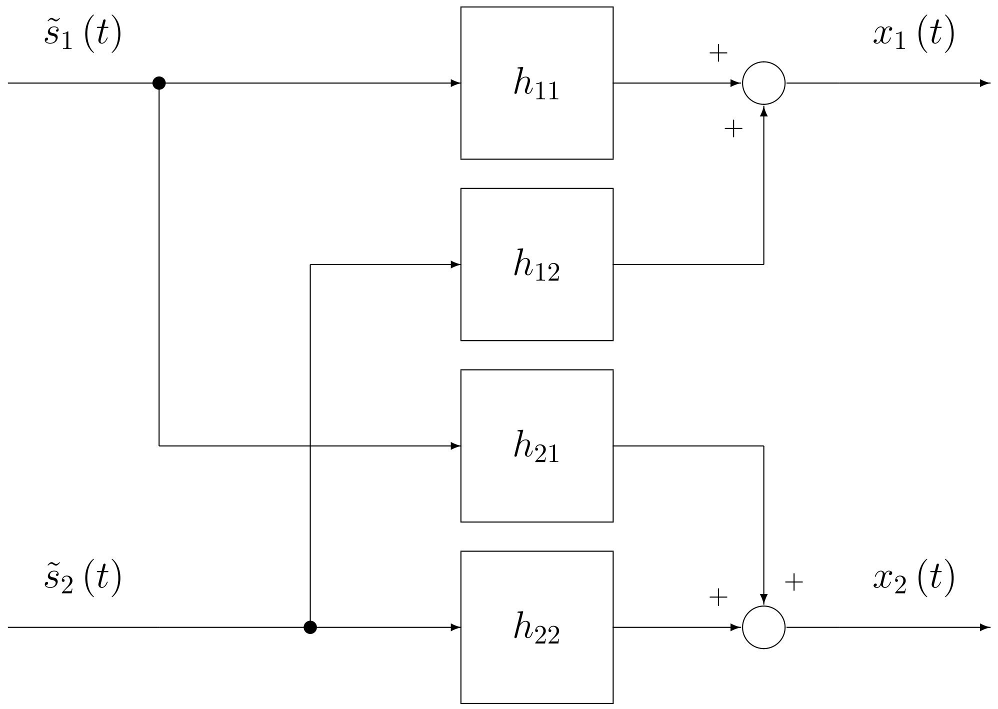


Fig. 2.2: The actual mixing system

2.2 The Problem

As mentioned in the introduction, the mixing system as described by figure 2.1 is based on the assumption that a signal-free noise approximation can be acquired. Because this cannot always be guaranteed, figure 2.2 better describes the true situation (assuming a non-reverberant environment).

2.2.1 The Instantaneous Mixing System

The *instantaneous* mixing system (a mixing system that assumes no delay and single-path propagation) as shown in figure 2.2 can be summarized by the following matrix equation

$$\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{pmatrix} \tilde{s}_1(t) \\ \tilde{s}_2(t) \end{pmatrix} \tag{2.7}$$

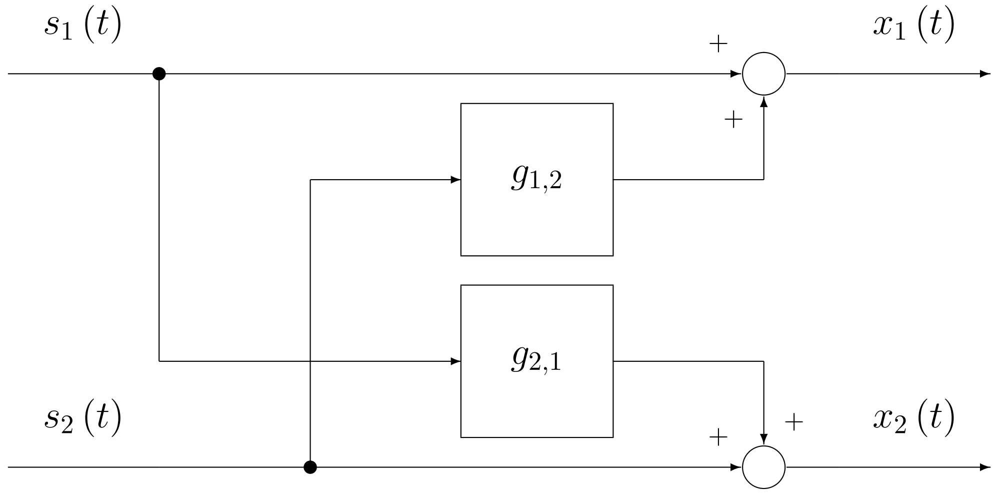


Fig. 2.3: The simplified mixing system

where $\tilde{s}_1$ and $\tilde{s}_2$ are the sources, $x_1(t)$ and $x_2(t)$ are the observed signals, $h_{1,1}$, $h_{1,2}$, $h_{2,1}$ and $h_{2,2}$ are the scalar mixing coefficients, and t is the time index.

Alternatively, this can be seen as each receiver is receiving some component of both sources, with no delays and no filtering. Any developed separating system must converge to some pseudo-inverse of the mixing system, which is generally assumed stationary over short periods of time. How well the demixing system follows any changes that *do* occur in the mixing system is known as its *tracking* ability.

The instantaneous model was used in the earlier investigations into the separation of mixed signals (such as [6]). While it is more simplistic than what actually occurs in real environments, it takes into account crosstalk, whereas adaptive filtering techniques such as LMS do not. This substantially relaxes the requirements on the positioning of the sources and sensors, and hence has provision for situations where it is either impossible or impractical to arrange the sensors in such a way that there is a signal-free noise estimate.

In the majority of situations, separation is achieved by the assumption of statistical independence of the sources. In [43], Cardoso gives a general overview of the statistical principles used with BSS and *independent component analysis* (ICA). Although he looks at the problem specifically from an instantaneous viewpoint, the concepts described within can be extended to *convolutive* mixtures.

When only separation of signals is desired, a model based on figure 2.3 is sufficient. Under these conditions, separation will not necessarily result in the original signals $\tilde{s}_1$ and $\tilde{s}_2$, rather the scaled versions $s_1 = h_{1,1}\tilde{s}_1$ and $s_2 = h_{2,2}\tilde{s}_2$. Because the primary goal is to separate the sources rather than measure their absolute magnitude, separation up to scaling is acceptable.

Another issue that arises is that the outputs may also be a permuted version of the sources. In other words, without additional information about the sources and further processing, there is no way of knowing which source maps to which output.

These two properties can be summarized by the equation

$$\mathbf{y}_{ins}(t) = \Lambda P \mathbf{s}(t) \tag{2.8}$$

where $\mathbf{y}_{ins}$ is the vector of instantaneous separated signals, Λ is a diagonal scaling matrix, P is a permutation matrix, and $\mathbf{s}$ is the vector of sources.

The assumption of a mixing system as in Fig. 2.3 will fall down for the case of an *underdetermined* mixture. This occurs when the number of sources exceeds the number of observations. While the current work will have a focus on the *determined* case, the underdetermined problem is worth noting as many authors have investigated this specific issue. For example, Li *et al.* in [53] look into this problem, and attempt to best estimate the mixing matrix where the number of sources is greater than the number of observations.

Although the convolutive model is generally more accurate than the instantaneous one, there are some situations (such as the radar signal separation in [39]) where assuming a non-convolutive model is not significantly detrimental to the results. Its comparative simplicity may justify its use in these cases.

2.2.2 The Convolutive Mixing System

In recent years, more interest has been taken in convolutive environments due to the presence of reverberation in real environments and non-instantaneous media [69,72,81]. The convolutive model extends on the instantaneous model by exchanging the scalar mixing coefficients with filters.

$$\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{bmatrix} \mathbf{h}_{1,1}^T & \mathbf{h}_{1,2}^T \\ \mathbf{h}_{2,1}^T & \mathbf{h}_{2,2}^T \end{bmatrix} \begin{pmatrix} \mathbf{s}_1(t) \\ \mathbf{s}_2(t) \end{pmatrix} \tag{2.9}$$

where $\mathbf{s}_1$ and $\mathbf{s}_2$ are the length-L source regressor vectors, $\mathbf{h}_{1,1}$, $\mathbf{h}_{1,2}$, $\mathbf{h}_{2,1}$ and $\mathbf{h}_{2,2}$ are the length-L vectors of channel filter taps, x_1 and x_2 are the observed signals, t is the time index, and the superscript ^{T} is the transpose operation.

For audio applications, this reverberation can be a very complex system. To give an idea of just how complex, consider the work done in [75]. Described in this paper is a new and efficient method of modeling the acoustic wave pattern using the parallel processing of GPUs. When modeling a source in a mid-sized room, a one-second long room transfer function takes 18 minutes to calculate. This, however, is significantly faster than the five hours it takes to model the wave pattern on a traditional CPU.

Although more complex than the instantaneous model, separating methods using the convolutive model have more possible real applications because real environments tend to be convolutive, and the medium upon which the signal propagates is never truly instantaneous; there are always delays.

An important issue that arises when dealing with convolutive mixtures is that the system may be *non-minimum phase*. Non-minimum phase channels occur when their zeros exist outside of the unit circle in the Z-domain. This means that, although the channel itself is stable, its inverse is not, rendering any separation algorithm using the channel inverse useless. In the audio environment, a *minimum phase* channel will occur when the room impulse response is as heavily weighted towards early reflections as is possible for the given frequency response of the channel's tranfer function (see appendix A for more details). In [45], Lee *et al.* show how a non-minimum or true phase system can be viewed as the combination of a minimum phase system combined with an all-pass system.

$$h(z) = h_{min}(z)\, h_{AP}(z) \tag{2.10}$$

where $h_{min}(z)$ is the minimum phase system with the same frequency response as $h(z)$, and $h_{AP}(z)$ is an all-pass filter that simply delays any signal that passes through it. This identity allows for the calculation of a pseudo-inverse to the channel, although the output will be a delayed version of the source signal. The amount of delay is defined by $h_{AP}(z)$ in Equation (2.10).

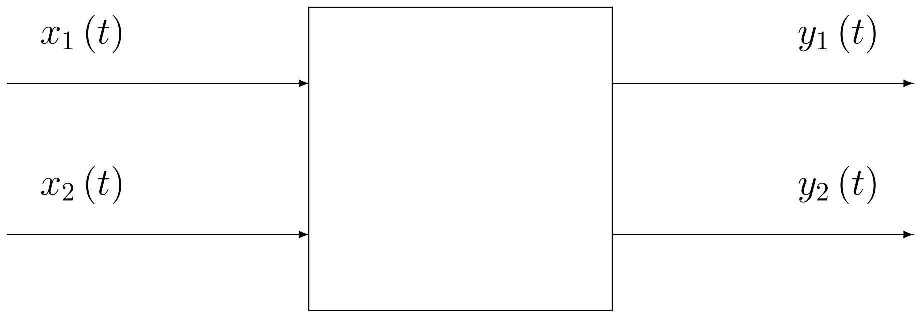


Fig. 2.4: A generic TITO demixing system.

2.3 Potential Solutions

In general, the purpose of any given separating algorithm is to find some demixing system (as shown in figure 2.4) such that the outputs $y_1(t)$ and $y_2(t)$ match the sources $s_1(t)$ and $s_2(t)$ up to some tolerable variation in form. Often all but one of the sources are simply additive noise, meaning that the demixing system only needs one output. However, this is based on a priori information about the sources which may actually be unknown.

One such method is beamforming as described in the overview given by Van Veen and Buckley in [82]. Beamforming is based on the assumption that the *direction of arrival* (DOA) of the desired source is known, therefore enabling signals received from all other directions to be ignored. However, this makes beamforming inadequate for many situations as often the DOA is unknown.

Either one of the two demixing systems as described in figures 2.5 and 2.6 may be used to achieve separation. In general, the devised algorithm will adapt the filter weights according to some specified separating requirements. Torkkola shows in [42] that methods using the feedforward structure in figure 2.5 will result in *white* outputs, whereas those employing the feedback structure in figure 2.6 will result in filtered versions of the sources.

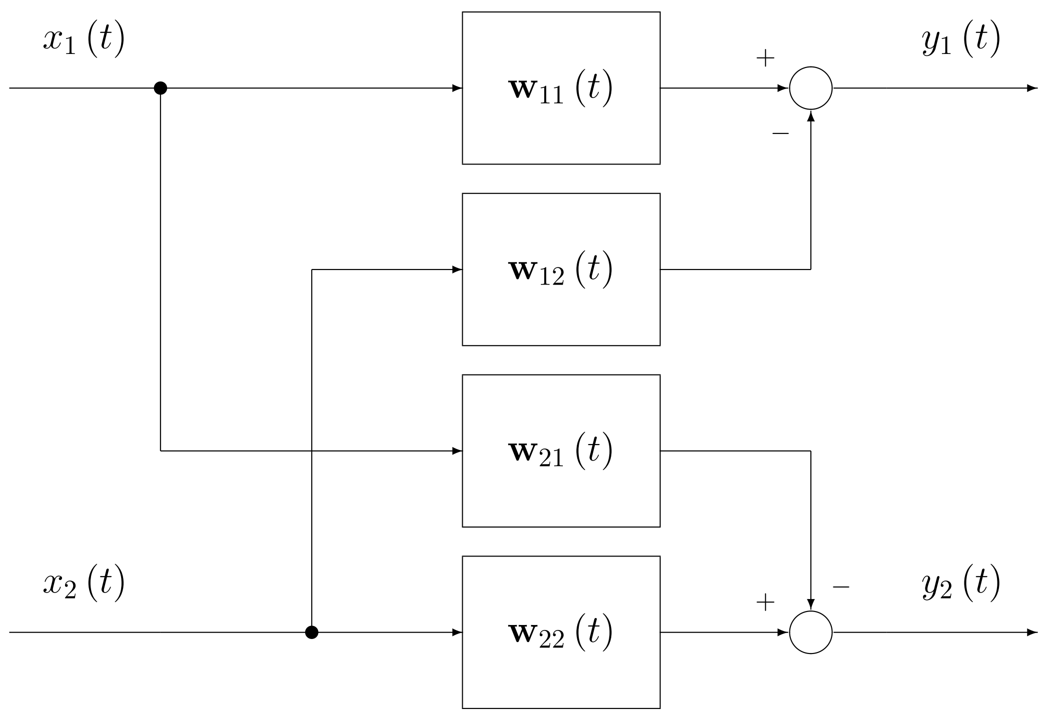


Fig. 2.5: A TITO feedforward demixing system structure.

Taking the feedforward case, both x_1 and x_2 contain a mixture of both sources. Therefore, in maximizing independence, $w_{1,2}$ and $w_{2,1}$ will converge to values that remove not only crosstalk between x_1 and x_2, but also any *temporal* correlation in the sources.

In essence, the feedforward case is solving the demixing problem from the view of

$$\mathbf{y}_{ff}(z)=w_{ff}\left(z^{-1}\right)\mathbf{x}(z) \tag{2.11}$$

whereas the feedback case is solving it from the perspective of

$$\mathbf{x}(z)=w_{fb}\left(z^{-1}\right)\mathbf{y}_{fb}(z) \tag{2.12}$$

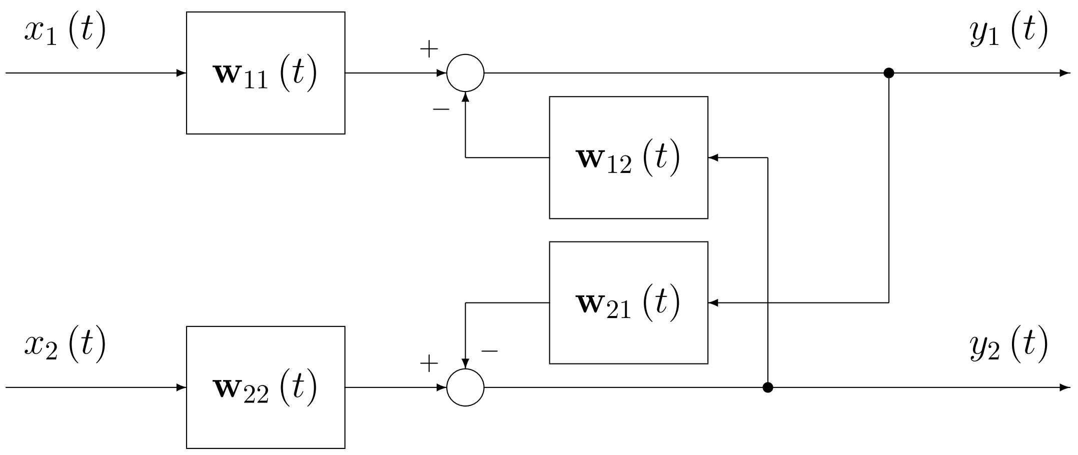


Fig. 2.6: A TITO feedback demixing system structure.

where

$$\mathbf{x}(z)=\mathcal{Z}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$
$$\mathbf{y}(z)=\mathcal{Z}\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}$$

and $w(z^{-1})$ is the desired demixing system. In general, on separation $w_{ff} \neq w_{fb}^{-1}$ and $\mathbf{y}_{ff} \neq \mathbf{y}_{fb}$, where $\mathbf{y}_{ff}$ and $\mathbf{y}_{fb}$ are the output vectors of the feedforward and feedback cases respectively. Rather, the desired $\mathbf{y}_{ff}$ and $\mathbf{y}_{fb}$ are simply vectors with independent entries, and are not unique.

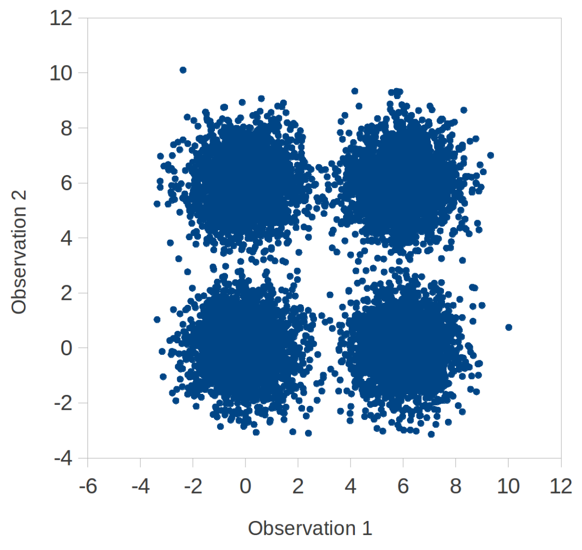
Therefore, the choice of demixing system should be dependent on the application. Speech, for example, is naturally temporally correlated; *whitening* the signal could make it sound unnatural. On the other hand, if trying to separate communications signals, whitening may be desired.

It is important to note that separation algorithms based on the mixing and demixing models described in this work do not work well in the case when the noise cannot be modeled as a point source, known as *diffuse noise*. For example, in [83] Takahashi *et al.* points out that ICA does not do well in separating a point source from diffuse noise. However, it can adequately separate the diffuse noise from the point source. They use these properties to improve on the performance of typical ICA by subtracting the separated noise from the mixture in the power-spectrum, resulting in a better approximation of the point source. This approach would also potentially be feasible to other algorithms that are based one one of the structures in figures 2.5 and 2.6 but are not ICA-based.

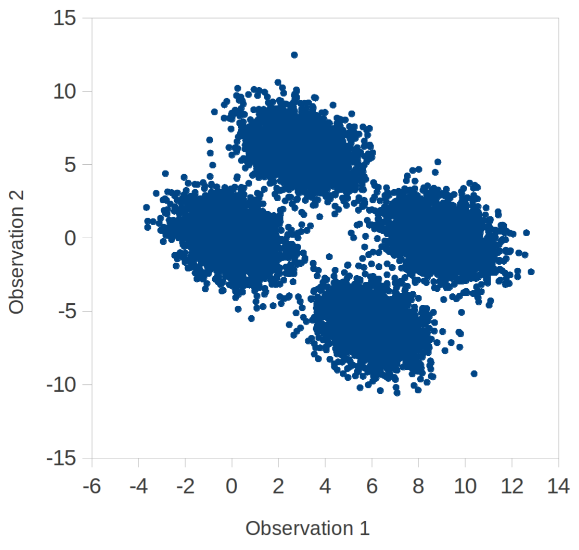
2.3.1 Independent Component Analysis

ICA is based solely on the assumption that the sources are statistically independent. It was first investigated in its present form by Jutten and Herault, who referred to it under the acronym INCA [11]. In [43], Cardoso outlines how basic statistical principles lead to ICA, and requirements for it to work. He then expands on this to shows how mixing signals tends to Gaussianize them (according to the central limit theorem), indicating that a separating procedure will de-Gaussianize the mixture. This fact can also be used to explain why two *Gaussian* signals cannot be separated by this method, but require the existence of *higher order statistics* (such as non-zero kurtosis).

For example, figure 2.7a (adapted from the example given in [43]) shows the *probability density function*s (PDFs) of two independent sub-Gaussian (a process that has a PDF with a kurtosis less than zero) signals, where the horizontal and vertical axes represent the two sources, and each point represents the signal sent by each of the sources at a specific time. As is clear from the figure, variations in one signal has no effect on the PDF of the other signal.



(a)



(b)

Fig. 2.7: fig:LiteratureReview:SubgaussianPlatykurtic Independent and fig:LiteratureReview:SubgaussianPlatykurticMixed mixed *sub-Gaussian* signals.

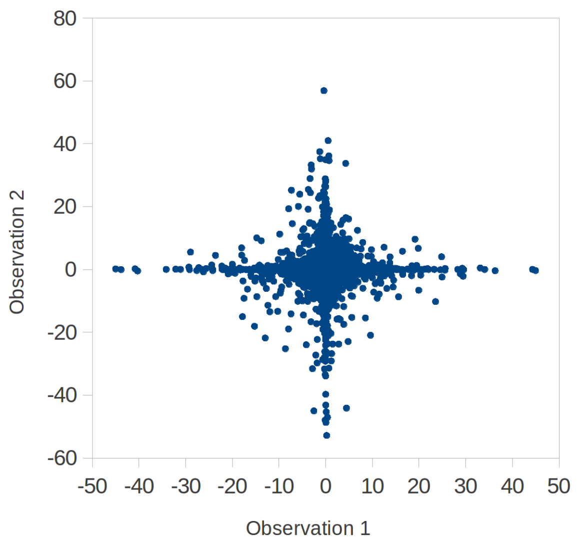
Figure 2.7b shows the PDF of the same sources, but after being passed through the instantaneous mixing matrix

$$H = \begin{bmatrix} 1 & 0.5 \\ -1 & 1 \end{bmatrix} \tag{2.13}$$

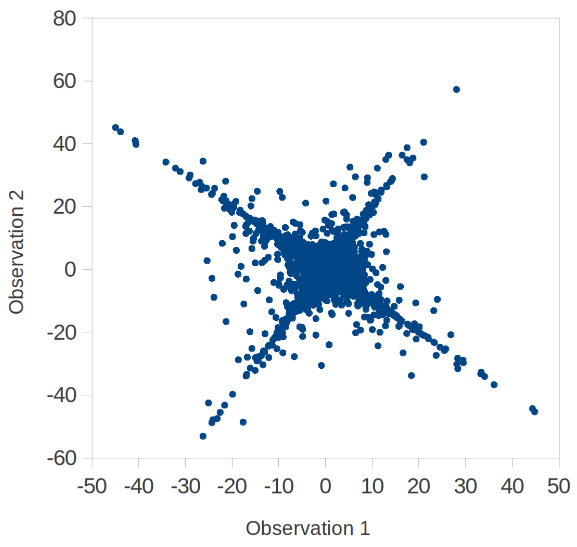
to give the displayed output.

From this figure it is clear that the variations of one signal does affect the PDF of the other signal. Therefore the aim of ICA is to try to find a demixing matrix such that demixed signals are again independent.

Figures 2.8a and 2.8b show a similar situation as in figures 2.7a and 2.7b, but with two



(a)



(b)

Fig. 2.8: fig:LiteratureReview:SupergaussianLeptokurtic Independent and fig:LiteratureReview:SupergaussianLeptokurticMixed mixed *super-Gaussian* signals.

super-Gaussian (a process that has a PDF with a kurtosis greater than zero) signals rather than two sub-Gaussian signals. Again, the dependence between PDFs in figure 2.8b is clearly observed.

However, figures 2.9a and 2.9b show that for two Gaussian signals, the independence cannot be verified. For example, if the two source signals are mixed by a rotation matrix such that it rotates the PDF in figure 2.9a by the 45° rotation matrix

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}, \tag{2.14}$$

the PDF will not have changed, even though the output will be as mixed as possible. From this, it can be seen that the independence assumption is not adequate to separate a mixture

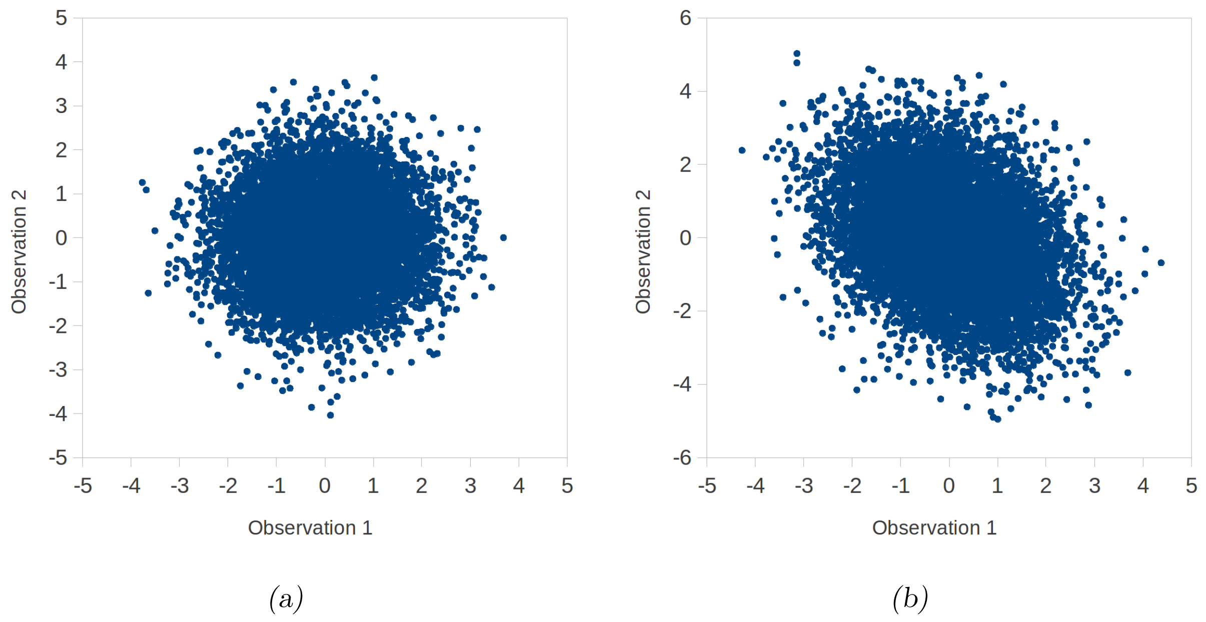


Fig. 2.9: fig:LiteratureReview:GaussianMesokurtic Independent and fig:LiteratureReview:GaussianMesokurticMixed mixed Gaussian signals.

where more than one source is Gaussian.

In order to model these PDFs properly, there has to be some method of estimating the PDFs. In [84], Comon illustrates how these functions can be modelled using the Edgeworth expansion and the third and fourth order cumulants of the signals. Similar to how the Taylor series is used to model a function given certain powers of the independent variable, the Edgeworth expansion is a PDF estimator based on the cumulants of the variable.

One problem that can arise in ICA is what is known as the whitening effect. This occurs not only when dependencies between channels are minimized, but also temporal dependencies within channels are removed, in a similar way as with *mutlichannel blind deconvolution* (MBD). Because speech is naturally temporally dependent, when these dependencies are removed the sound of the audio becomes unnatural. In [63], Liang *et al.* propose additional constraints to the cost function to reduce this temporal whitening. In their paper they also propose a method of increasing convergence rate by using a null beamformer to initialize the separation matrix.

MBD stemmed from the problem of deconvolving signals in reverberant environments. However, rather than the assumption of one source (such as Miyoshi and Kaneda in [5]) the multichannel case attempts to deconvolve a mixture of sources. Its basic premise is very similar to ICA, except that it also assumes that the sources are *independent and identically distributed* (i.i.d.). On the basis that this assumption is correct, MBD outperforms conventional BSS because it can reconstruct the original signals up to permutation and *scaling*, whereas the assumptions of typical BSS only allow reconstruction of the signals up to permutation and *filtering*.

While the assumption of temporal independence may be true of signals such as are present in wireless communications applications, it is not so in human speech, which is highly temporally correlated. Douglas [62] approaches this problem with the assumption that speech can be modeled as an autoregressive (AR) system which is driven by a temporally independent signal. Using this, he is able to reconstruct the original signals up to permutation, scaling, and possible delay.

ICA can be extended in various different ways. For example, in [52], Shi *et al.* exploit *non-Gaussianity*, linear predictability, and nonlinear predictability to increase the computational efficiency of ICA.

Independent Vector Analysis

In more recent years, there has been a focus on the convolutive case. As a result, in [76] (then later in [80]), Kim *et al.* investigate the idea of *independent vector analysis* (IVA). Rather than treating the sources as singular components, IVA transforms them to the frequency domain, then treats them as vectors that span across all frequency bins. Treating them as vectors allows separation, but with consistency across each frequency. This has the advantageous effect of solving the permutation problem that plagues many frequency-domain algorithms.

This idea is also employed in [47] and [48] by Anderson *et al.* who use it to analyze MRI signals, where there is *high* correlation between datasets. While they show that their method works well for the case of MRI signals, they also note that it would *not* be suitable for frequency domain BSS, where there is not a strong correlation between datasets.

Sparse Component Analysis

Another aspect of ICA that has been under investigation in recent years is *sparse component analysis* (SCA). For a truly blind situation, there are many cases where we *cannot* guarantee that the number of sources is not greater than the number of observations. SCA endeavors to solve this problem.

In [53], Li *et al.* attempt to use sparsity to discover the mixing matrix of an underdetermined system. They point out that while sources tend not to be sparse in the time domain (they are consistently active), they may be sparse in the frequency domain. This sparsity can be exploited to better estimate the mixing matrix, even though the number of sources exceed the number of observations. In [85], Jayamaran *et al.* pick up this idea but also try to ensure that any matrix estimation will work in the case that the mixing system is *overdetermined* (i.e. the number of sources *is less than* the number of observations).

Other work in this area includes that done by Mohimani *et al.* in [86], where the l^0 norm is used as a measure. Typically the l^1 norm is used, but this adaptation performs similarly while substantially cutting down on the computational requirements.

Virtanen [74] also uses SCA, although he specifically looks at the single-channel case. The main application in mind for his work is separating a music mixture. These mixtures tend to be instantaneous, so they are somewhat easier than the convolutive case. However, this has high quality requirements, as music is for listening pleasure as compared to something like speech where the main aim is simply information transfer.

2.3.2 Maximum Likelihood

The maximum likelihood principle approaches the BSS problem with the assumption that the approximate PDF of the sources is known *a priori*. It attempts to find a mixing matrix H such that when its inverse is multiplied by the mixed signal vector $\mathbf{x}$, it closely approximates the assumed PDF of the sources. If taking the sub-Gaussian example as described by figures 2.7a and 2.7b, the maximum likelihood method would try to find H such that

$$H\mathbf{x} = \hat{\mathbf{s}} \tag{2.15}$$

where $\hat{\mathbf{s}}$ is a random vector of independent entries with the same statistical distribution as what is assumed of the sources.

One of the most common methods for measuring the deviation from the hypothesized PDF is by using the *Kullback-Liebler divergence*. The Kullback-Liebler divergence is calculated using [43]

$$K[\mathbf{f}|\mathbf{g}] \triangleq \int_{\mathbf{s}} \tilde{f}(\mathbf{s}) \log \left(\frac{\tilde{f}(\mathbf{s})}{\tilde{g}(\mathbf{s})} \right) d\mathbf{s} \tag{2.16}$$

where $\mathbf{f}$ and $\mathbf{g}$ are two random vectors, and $\tilde{f}(\mathbf{s})$ and $\tilde{g}(\mathbf{s})$ are their respective PDFs. This function results in a quantity ≥ 0 with equality only when the PDF of $\mathbf{f}$ matches the PDF of $\mathbf{g}$. Although it is used to find the closeness of two PDFs, the Kullback-Liebler divergence is not *symmetric* (ie. $K[\mathbf{f}|\mathbf{g}] \neq K[\mathbf{g}|\mathbf{f}]$).

Girolami [14] notes that for speech, the PDF is a Laplacian distribution, which makes speech super-Gaussian. He therefore proposes a PDF that closely approximates the Laplace distribution for signals that are known to be super-Gaussian, with an alternative PDF for signals that are known to be sub-Gaussian (such as a computer fan in an office environment).

A number of authors choose to do the separation in the frequency domain, which is advantageous because that signals in the frequency domain tend to have highly super-Gaussian PDFs [60]. This not only makes modeling the sources easier (simply taking super-Gaussian PDFs), it also means that there is potential for the separating system to achieve better performance. However, along with its benefits, the frequency domain separation procedures suffer from other problems, such as the *permutation problem* described in section 2.3.6.

Of important note is the infomax principle. Although it uses the *Shannon entropy* rather than the Kullback-Liebler divergence, it results in the same contrast function as the maximum likelihood approach. One such approach that chooses to maximize the entropy over

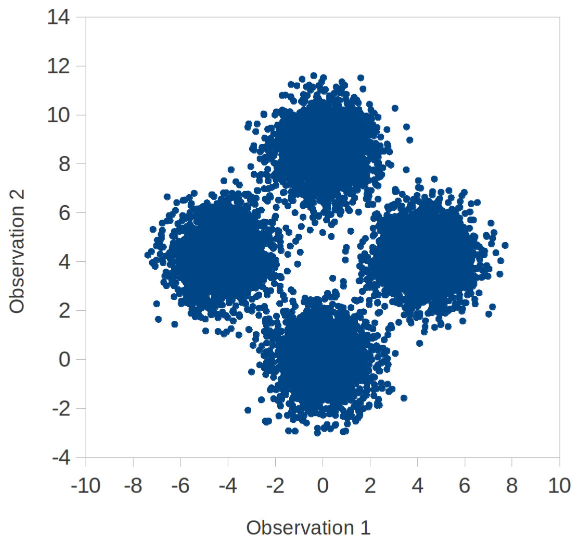


Fig. 2.10: The result of separation based on maximum likelihood when there is a model mismatch.

minimizing the *mutual information* is the approach proposed by Torkkola in [42].

2.3.3 Mutual Information

A significant problem with the maximum likelihood approach is that it is based on the assumption that the approximate PDF of the sources is known. The mutual information approach combats this problem by integrating a measure of deviation of the hypothesized model from the true PDFs into the contrast function.

Take the case of trying to separate signals consisting of a mixture of two sub-Gaussian sources as described by figure 2.7a when using a model based on super-Gaussian sources as in figure 2.8a. The match will be closest when the global mixing matrix C is the 45° rotation matrix in (2.14), where

$$\mathbf{y} = C\mathbf{s} \tag{2.17}$$

This does not result in separation, rather in each of the entries of $\mathbf{y}$ having equal components of each of the sources. The need therefore arises to integrate into the contrast function a factor that measures the deviation between hypothesized and actual model PDF.

The first step in developing such a contrast function is to introduce a random vector $\tilde{\mathbf{y}}$ such that its entries are independent and have the same PDF as the output vector $\mathbf{y}$. In [43], Cardoso points out the important property that

$$K\left[\mathbf{y}|\hat{\mathbf{s}}\right] = K\left[\mathbf{y}|\tilde{\mathbf{y}}\right] + K\left[\tilde{\mathbf{y}}|\hat{\mathbf{s}}\right] \tag{2.18}$$

where $\hat{\mathbf{s}}$ can be any vector with independent entries.

From this identity, we can see that the Kullback-Liebler divergence between the hypothesized PDFs and the demixed signals' PDFs is comprised of the sum two factors: 1) the independence of the outputs and 2) how closely the hypothesized PDF matches the output PDFon.

The described decomposition aids in preventing the contrast of any derived algorithm from developing a global minimum that does not occur at total separation.

2.3.4 Second-Order Statistics

A number of papers have proposed independence criterion based on second-order statistics (SOS) [1]. However, as is shown in [87], SOS alone do not provide sufficient constraints for complete separation. Therefore, further assumptions are needed. If not explicitly exploiting higher-order-statistics in the form of kurtosis or other measures of non-Gaussianity, the cost functions usually employ an assumption of either *nonstationarity* or *nonwhiteness*.

In [88] Tong *et al.* first proposed using singular value decomposition (SVD) on the autocorrelation matrix of the observed signals to separate instantaneously mixed signals. Although such a method is based only on SOS, it overcomes the insufficiency of SOS in separation by exploiting the nonstationarity of the input signals. This procedure has come to be known as *joint diagonalization*.

Joint diagonalization takes advantage of the following identity

$$\mathbf{P}_t\left(z^{-1}\right) = \mathbf{H}\left(z^{-1}\right)\mathbf{D}_t\left(z^{-1}\right)\mathbf{H}\left(z^{-1}\right)^H \tag{2.19}$$

where $\mathbf{P}_t\left(z^{-1}\right)$ is the cross-spectral density of the observed signals at time t, $\mathbf{H}\left(z^{-1}\right)$ is the polynomial mixing matrix, and $\mathbf{D}_t\left(z^{-1}\right)$ is the cross-spectral density of the sources at time t, and the superscript ^{H} denotes the hermitian transpose. Note that due to the assumption that the source signals are uncorrelated, $\mathbf{D}_t\left(z^{-1}\right)$ will be diagonal for all t.

Therefore, if it is possible to estimate the operation as described in (2.19), then both the mixing system and the original sources can be found. In [61], Rahbar and Reilly propose such an algorithm in order to find the matrices that best approximate the diagonalization procedure. They describe how not only is it possible to separate the sources, but that the separated signals are permuted and *scaled* versions of the original sources, rather than *filtered* versions.

Because joint diagonalization is dependent on SOS, further assumptions have to be made in order for separation to take place. The most common choice is either nonstationarity or non-Gaussianity. Like Rahbar and Reilly, Mei *et al.* choose nonstationarity for the additional information [41]. However, while Rahbar and Reilly attempt to find and demix the mixing system, Mei *et al.* propose to directly find a demixing system, increasing computational efficiency.

In [46], Parra and Spence also suggest a procedure that utilizes the SOS at multiple time periods. This allows for complete separation under the assumption that the *cross-correlation* matrices change over time. This explicitly requires that the sources are nonstationary. Also, unlike most other authors, they do not make the assumption that the sensors are noise-free, but incorporate a noise component in their analysis. Under the assumption that the noise signals on the different sensors are independent, they show that this noise does not have a significant effect on the convergence properties of the separating system. In [89], Choi and Cichocki also perform separation based on SOS under the constraint of a noisy environment, showing the robustness of the joint diagonalization procedure to this additional noise.

Similarly, Sahlin and Broman in [65] exploit nonstationarity, but with an emphasis on real-world signals. More recently, Kocinski *et al.* [30] also have an emphasis on a real-world application, specifically speech. As a performance metric, they use speech intelligibility of the separated signals, and compare it against the SNR of the observations, with fairly substantial results.

Buchner *et al.* [72] not only employ the assumption of nonstationarity, they also assume nonwhiteness. Using these additional assumptions, a broadband frequency domain separation procedure is proposed. These additional constraints not only aid in providing constraints for separation, but also in solving the internal permutation problem discussed in section 2.3.6.

Although not strictly only second order statistics, in [1] Buchner *et al.* also propose the *Triple-N ICA for Convolutive mixtures* (TRINICON) framework which exploits the three "N"s; nonstationarity, nonwhiteness, and non-Gaussianity. Although the non-Gaussianity explicitly introduces higher order statistics, the inclusion of all three assumptions gives a more robust algorithm. For example, when the signals are Gaussian, TRINICON can still exploit nonstationarity and nonwhiteness to provide the constraints needed for separation.

Another class of algorithms based on SOS are those based on the *crosstalk resistant adaptive noise canceller* (CTRANC) of Mirchandani *et al.* in [90]. It overcomes the SOS indeterminacy by putting restrictions on the mixing system, meaning that it is no long entirely blind. However, these mixing system constraints also mean that the outputs equal the inputs up to *filtering* rather than *filtering and permutation*, which is common for most BSS algorithms. This makes such algorithms more suitable for applications where permutation is a significant problem, for example with live audio mixing. The CTRANC and other similar algorithms are covered in more detail in section 2.3.7.

2.3.5 Optimization Procedures

When a contrast needs to be minimized, there are a number of methods available. The important qualities of each method is:

- *Convergence time.* The more quickly an optimization procedure can converge to the minimum of the contrast, the better it is at tracking any changes in the system.

- *Convergence to the correct solution.* It is important that the minimum of the contrast is actually reached, and that the minimum actually corresponds to separation.
- *Stability.* Some optimization procedures are prone to instability under certain conditions. In [91], Amari *et al.* perform a stability analysis of some of the more common optimization algorithms; namely the gradient algorithms. They note that instability often occurs when the source signals are a mixture of super-Gaussian and sub-Gaussian signals.

In [92], Pham and Garat develop a relative gradient algorithm based on the maximum likelihood contrast. They show how separation can be achieved by solving the following set of estimating equations

$$\hat{E}\left[\psi_i\left(\mathbf{e}_i^T\hat{H}^{-1}\mathbf{x}\right)\mathbf{e}_i^T\hat{H}^{-1}\mathbf{x}\right]=0,\qquad i\neq j=1,\ldots,N \tag{2.20}$$

where $\hat{E}$ is the sample average operator, $\psi\left(\cdot\right)$ is a nonlinear function based on its argument's PDF, e_i is the i^{th} column of the identity matrix $\mathbf{I}$, $\hat{H}^{-1}$ is the demixing matrix estimate, $\mathbf{x}$ is the vector of observed signals, and N is the number of sources to separate.

This shows an interesting property: because the output $y_i\left(t\right)$ is the only non-zero entry of the vector

$$\mathbf{e}_i^T\hat{H}^{-1}\mathbf{x}$$

the gradient is simply the expected value of a nonlinear function of the output signals $\hat{H}^{-1}\mathbf{x}$. Using this, they develop an off-line algorithm to separate the signals. Although they base their algorithm on a maximum likelihood contrast, they show through simulations that if even if there is a slight model mismatch, the algorithm can still converge to the correct solution. However, if the mismatch is substantial enough, the algorithm will likely converge to a spurious result.

In [6], Cardoso and Laheld also implement a relative gradient approach, but based on serial updating. Serial updating is when the update equations are in the form $W(t+1) = HW(t)$. Their update equation is

$$\begin{aligned} W_{t+1} =& W(t) - \boldsymbol{\lambda}(t) B(\mathbf{y}(t)) W(t) \\ =& [I - \lambda(t) B(\mathbf{y}(t))] W(t) \end{aligned} \tag{2.21}$$

where $\lambda(t)$ is a sequence of adaptation steps, and $B(\mathbf{y}(t))$ maps the length-n estimated signal vector $\mathbf{y}(t)$ to a $n \times n$ matrix.

Serial updating has an interesting and useful property. The trajectory of the *global* mixing system

$$C(t) = W(t) H \tag{2.22}$$

is equivariant to the mixing matrix H for specified initial conditions $W(0)$. This can be seen as follows.

Let $C(t)$ be defined in (2.22) as the global system. Considering Equation (2.21),

$$\begin{aligned} C(t+1) =& [I - \lambda(t) B(\mathbf{y}(t))] W(t) H \\ =& [I - \lambda(t) B(\mathbf{y}(t))] C(t) \end{aligned} \tag{2.23}$$

This shows that if the initial *global* state of two systems is the same, then the path taken by the global system will be the same, regardless of the mixing system H.

Let us now consider two unique mixing systems, one with a mixing matrix H, and the other with the mixing matrix H'. In order for the two systems to follow identical trajectories, we must choose $W(0)$ and $W'(0)$ such that $W(0)\,H = W'(0)\,H'$. This shows that changing the mixing system is equivalent to simply changing the initial conditions.

Perhaps the most implemented optimization algorithm for use in BSS is the natural gradient adaptation proposed by Amari in [93]. A vast number of papers opt to use the approach due to its Newton-like convergence yet with a computational complexity similar to that of stochastic gradient descent. While the ordinary gradient descent algorithm tends to plateau before it properly converges to the equilibrium point, the natural gradient does not slow its convergence as significantly when approaching the desired solution [40].

The reason that the natural gradient can converge to the solution more quickly than the ordinary gradient is that the ordinary gradient assumes a *Euclidean space* for its contrast where the natural gradient more correctly assumes a *Riemannian manifold* [8]. The result of this is that the ordinary gradient descent only works well in contrasts that are *isotropic* about the minimum.

For example, when the earth's surface is mapped to an atlas, the shortest point between two points on that atlas is not a straight line, but a curve. This is due the curvature of the earth's surface having to be reparameterized in terms of a Euclidean plane. If solving a contrast using the ordinary gradient descent algorithm, this is equivalent to taking the straight-line path according to the atlas, which is *not* necessarily the shortest distance [94].

Amari and Douglas [94] show how on a Riemannian manifold, the distance between two points $\mathbf{a}$ and $\mathbf{c} = (\mathbf{a} + \mathbf{b})$ (where $\mathbf{a}$ and $\mathbf{b}$ are vectors) can be given by the following equation

$$\sqrt{\mathbf{b}^T G(\mathbf{b})\, \mathbf{b}} \tag{2.24}$$

where $G(\mathbf{b})$ is a matrix known as the *Riemannian metric*. When $\mathbf{a}$ and $\mathbf{c}$ are in Euclidean space, the Riemannian metric is simply the identity matrix, and Equation (2.24) simplifies to the *Euclidean norm*.

In [93], Amari derives in detail the natural gradient learning equation for BSS as

$$\frac{dW}{dt} = \mu_t \left(I - \boldsymbol{\phi}(\mathbf{y})^T \mathbf{y} \right) W \tag{2.25}$$

where W is the demixing matrix, $\mathbf{y}$ is the vector of demixed signals μ_t is a step-size, and $\boldsymbol{\phi}(\cdot)$ is defined by

$$\boldsymbol{\phi}(\mathbf{y}) = [\phi_1(y_1), \ldots, \phi_n(y_n)] \tag{2.26}$$

$$\phi_l(y_l) = -\frac{d}{dy_l} \log f_l(y_l) \tag{2.27}$$

where $f_l(\cdot)$ is a nonlinear function that is chosen based on the PDFs of the sources. While exact choices of $f_l(\cdot)$ for given PDFs optimize the convergence rate, in general the PDFs are unknown exactly, so

$$f_l(y_l) = \mathrm{sign}(y_l) \tag{2.28}$$

is typically chosen for super-Gaussian sources, and

$$f_l(y_l) = y_l \, |y_l|^2 \tag{2.29}$$

is typically chosen for sub-Gaussian sources.

Of important note is the fact that the natural gradient update equation in (2.25) is a serial updating algorithm. As mentioned earlier, this means that the track taken by the demixing matrix as it converges is independent of the mixing system, but only dependent on the initial *global* mixing system.

The natural gradient can also be implemented on cost functions based on SOS. For example, Aichner *et al.* [64] propose such an algorithm, and implement it in both the time domain and in the frequency domain. While this may seem unnecessary, as contrasts based on SOS are inherently isotropic, Amari and Douglas [94] point out that the demixing filter's *Hessian matrix* (the matrix of all second-order derivatives) is equal to the Riemannian metric. This makes the natural gradient identical to Newton's method, which has much higher convergence rate than that of ordinary gradient descent.

In [81], Douglas *et al.* noted that when the natural gradient adaptation was used to minimize *finite impulse response* (FIR) filters where the filter orders were less than order of the channel,the filters would often converge to solutions that had spikes at either end of the filter. They then propose an adaptation to the natural gradient adaptation to remove these spikes, at the cost of an increase in compuational complexity.

Another optimization method is the use of *Lagrange multipliers*. In [60], Mitianoudis and Davies use Lagrange multipliers to maximize a nonquadratic contrast, basing it on the maximum likelihood estimation. They shows that, while it is more computationally expensive than the natural gradient algorithm, it converges substantially faster.

A somewhat different optimization procedure is a genetic algorithm. These algorithms are based on a similar structure as natural selection works in the biological world. Two "parent" systems produce a number of "offspring" systems with different permutations of the parent's parameters. Whichever offspring systems have the best outputs according to some cost function are then mated to produce more offspring. Additionally, mutations are allowed to prevent the optimization procedure from converging to local solutions. In the approach proposed by Sundaralingam and Sharman [95] a genetic algorithm is used to optimize *infinite impulse response* (IIR) filters. They show how it is fundamentally more stable than gradient-based methods, and is not as susceptible to convergence to local minima. This idea is then extended by Tan and Wang in [96] to the BSS problem.

2.3.6 Frequency Domain Algorithms

The main reason that authors choose to perform the separation procedure in the frequency domain as compared to the time domain is that it tends to be computationally more efficient. One reason for this is that while convolution may be a fairly computationally intensive process in the time domain, in the frequency domain it maps to a simple multiplication operation. This also means that solving a convolutive BSS problem in the time domain becomes solving multiple instantaneous BSS problems in the frequency domain, potentially simplifying the problem. In [36] Joho and Schniter propose a frequency domain equivalent to the MBD algorithm proposed by Amari *et al.* in [7]. They show that the adaptation makes the algorithm three times more efficient, although this comes at a cost of having a threefold increase of the convergence time.

Perhaps the most important problem unique to frequency domain BSS algorithms is known as the *permutation problem*. This is different from (though not unrelated to) the *global* permutation which occurs in BSS where the i^{th} source does not necessarily map to the i^{th} output of the separating system. Rather, the permutation inconsistency occurs when the signals that are separated in different frequency bins are not mapped to the same sources. This means that the separated signals in one frequency bin are not necessarily combined in the correct order with the signals separated in another frequency bin. Without solving this problem, conversion back to the time domain using the inverse Fourier transform will simply mix the signals again.

Ikram and Morgan explore the permutation problem in more detail in [71] (and later in [97]), and speculate that the problem may not be isolated to frequency domain algorithms, but also may exist to some extent in time domain algorithms when the filter size is large. However, as Buchner *et al.* point out in [72], while not disproven, this conjecture has very little practical evidence. That being said, it is still a possibility that should be kept in mind when developing time domain algorithms.

A number of approaches have been developed to combat this problem. In [60], Mitianoudis and Davies propose a measure of signal amplitude across the frequency axis, assuming that one source is louder than another at a certain point in time. Using this, they devise a method of using this loudness property to force the signals to separate to the correct permutation.

Another possible approach to solving the permutation problem is to derive a *broadband* frequency domain cost function as Buchner *et al.* do in [72]. While trying to minimize the cross-correlation between two signals in the same frequency band, the algorithm also minimized the correlation across different frequencies. This means that each signal is separated into the correct permutation, therefore maximizing separation. In [79], Sawada *et al.* also use correlation across frequencies, but additionally incorporate DOA information.

Also proposed is to have a cost function that is well-defined in the time domain while completing the actual calculations in the frequency domain. Joho and Schiter's [36] frequency adaptation of the approach proposed by Amari *et al.* [7] is such a case. Because the cost function is defined in the time domain, and assuming that time domain algorithms *do not* suffer from this permutation problem, the resultant algorithm should not suffer from it either.

A more recent approach is that of IVA, as first proposed by Kim *et al.* in [76] and [80], then later investigated by numerous authors (for example, [47, 48, 77]). This extends the idea of ICA to vectors, which overcomes the permutation problem when performed in the frequency domain. In [98], Itahashi and Matsuoka prove that a permuted solution *never becomes a local minimum*, provided the contrast exhibits certain properties.

2.3.7 Crosstalk Resistant Adaptive Noise Canceling

The CTRANC is a method of essentially cross-coupling two LMS or RLS filters in order to reduce the detrimental effect that crosstalk has on adaptive filters. In [13], Van Gerven and Van Compernolle describe the basic premise behind the TITO case, and claim that it is possible through only decorrelation to separate the signals using the feedback system as described in figure 2.5 and the update equations

$$\begin{aligned}\mathbf{w}_{1,2}(t) &= \mathbf{w}_{1,2}(t-1) + \mu_1 y_1(t)\,\mathbf{y}_2(t)\\ \mathbf{w}_{2,1}(t) &= \mathbf{w}_{2,1}(t-1) + \mu_2 y_2(t)\,\mathbf{y}_1(t)\end{aligned} \tag{2.30}$$

where $\mathbf{w}_{1,2}(t)$ and $\mathbf{w}_{2,1}(t)$ are the decoupling filters at time t, μ_1 and μ_2 are the adaptation step-sizes, $y_1(t)$ and $y_2(t)$ are the system outputs at time t, and $\mathbf{y}_1(t)$ and $\mathbf{y}_2(t)$ are the output regression vectors defined by

$$\mathbf{y}_1(t) = [y_1(t-1), y_1(t-2), \ldots, y_1(t-N_2)]^T$$
$$\mathbf{y}_2(t) = [y_2(t-1), y_2(t-2), \ldots, y_2(t-N_1)]^T$$

where N_1 and N_2 are the orders of the filters $\mathbf{w}_{1,2}(t)$ and $\mathbf{w}_{2,1}(t)$ respectively. The filters $\mathbf{w}_{1,1}$ and $\mathbf{w}_{2,2}$ are set to unity gain, meaning that the outputs will be filtered versions of the sources.

It is interesting to point out that the output signals will only be filtered versions of the sources (not permuted). This lack of permutation is due to the CTRANC being a *semi-blind* algorithm; the following restriction is placed on the mixing system - each microphone is the closest microphone to a unique source. In more geometric terms, for the TITO case, the sources are on either side of the plane that bisects the two microphones. Therefore, the CTRANC is not truly as “blind” as some other approaches.

What might first appear as a flaw in the update equations of (2.30) is the existence of only SOS. It is widely accepted that SOS are not sufficient for the separation procedure [43,72,87]. This seems to indicate that the separating algorithm in (2.30) is unlikely to properly separate the sources since decorrelation relies solely on SOS.

Weinstein *et al.* [87] explore the second-order problems that arise with ordinary LMS when in the presence of crosstalk. They also propose a separation system that utilizes two cross-coupled adaptive filters. In a TITO model, it is proven that if one of the crosstalk filters are known *a priori*, decorrelation is sufficient to decouple the signals. However, if neither of the crosstalk terms are known, they show that decorrelation is not sufficient for separation. To remedy this, they suggest that decorrelation can be supplemented with other criteria such as statistical independence (utilizing higher-order statistics), signal nonstationarity or spectral matching to be sufficient to separate the signals. In their proposed algorithm, they chose to use nonstationarity for the additional information.

In [43], Cardoso explains the insufficiency of the sole use of SOS in separating a mixing system. With a demixing system as described by equations (2.11) and (2.12), there are N^2L unknowns, where the demixing matrix $\mathbf{W}$ is a $N \times N$ matrix of filters of order $(L-1)$. Therefore, to discover what each of these coefficients are requires N^2L equations. However, SOS only supplies $N\left(N+1\right)L/2$ equations, making it inadequate in solving for $\mathbf{W}$.

The reason that crosstalk resistant adaptive noise cancelers can perform separation while only utilizing SOS is based on the assumption that the mixing system is of the form in Fig. 2.3, where $\mathbf{w}_{1,1}$ and $\mathbf{w}_{2,2}$ are both assumed unity gain filters, and $\mathbf{w}_{1,2}$ and $\mathbf{w}_{2,1}$ are strictly *causal* filters.

In [99], Lindgren and Broman show that under certain mixing conditions, SOS *are* adequate to separate a mixture of signals. If the *cross-correlation matrix* between the outputs of the demixing system is diagonal, this does not always correspond with separation. However, if certain constraints are placed on the mixing system, then the off-diagonal entries of the cross-correlation matrix can only be zero at separation.

These constraints are as follows:

1. The sources s_1 and s_2 are mutually i.i.d. with nonzero variance and zero mean.

2. The direct channels from source i to observed signal i are asymptotically stable.
3. The length of the impulse response for every channel is less than or equal to the length of the demixing filters.
4. The mixing system must be minimum phase.
5. The mixing system is strictly convolutive. The crosstalk channels must have at least two nonzero coefficients each.
6. The channels must be *causal*.

The first three of the six constraints are assumed true for a large majority of BSS algorithms, and the fourth is also assumed true for any that attempt to find an inverse of the mixing system. The fifth constraint will be true for a majority of convolutive mixing systems; there are very few that have only one path from source to observed signal, especially for audio applications.

The final constraint, that the channels must be causal, is one that few algorithms assume. However, the CTRANC *is* one of them. This restricts the positioning of the physical system; in an audio envirnoment, it means that each microphone has to be the closest microphone to a unique source. While this may be impractical for a truly blind system, it also has the advantage that the outputs match the sources up to filtering, rather than filtering *and permutation*, as is common for truly blind techniques. Therefore if there is knowledge of which source is the closest source to each observed signal, then no further processing is needed to determine which output corresponds to which source.

When implemented in a feedforward structure (as in Fig. 2.5), (2.30) holds. However, [13] also specifies that they can also be used in the feedback structure in Fig. 2.6, which assumes that $\nabla_{\mathbf{w}_{2,1}} y_2 = \nabla_{\mathbf{w}_{1,2}} y_1 = 0$ where $\nabla_u v$ is the partial derivative of v with respect to u. This assumption greatly decreases the computational complexity of the algorithm, but it results in adding a bias to the update equations, skewing the results.

Lepauloux *et al.* show in [73] that when taking into account the effect that $\mathbf{w}_{1,2}$ has on y_2, the update equations are adjusted to

$$\mathbf{w}_{1,2}(n+1)=\mathbf{w}_{1,2}(n)+\mu_1 y_1(t)\left[\mathbf{y}_2(t)+\mathbf{w}_1^T(t)\nabla_{\mathbf{w}_{1,2}}\mathbf{y}_2\right] \tag{2.31}$$

$$\mathbf{w}_{2,1}(n+1)=\mathbf{w}_{2,1}(n)+\mu_2 y_2(t)\left[\mathbf{y}_1(t)+\mathbf{w}_2^T(t)\nabla_{\mathbf{w}_{2,1}}\mathbf{y}_1\right] \tag{2.32}$$

They then explain how this assumption can lead to biasing where the feedback *symmetric adaptive decorrelation* (SAD) algorithm converges, but not necessarily to the correct solution.

Yousefi Rezaii and Geravanchizadeh [58] also chose to use the SAD implementation in a feedback structure as the basis for their proposal (meaning that it is susceptible to biasing). Rather than basing the algorithm on NLMS, they chose to cross-couple two *least mean mixed norm* (LMMN) filters, which tend to have faster convergence than NLMS filters. They additionally incorporated a voice activity detector (VAD) to increase the computational efficiency of the algorithm. Therefore, the SAD portion of their approach would only have to update when voice was detected. However, this is based on the assumption of a robust VAD, and would be ineffective in situations where the desired signal could not be detected by the VAD.

Considering the restriction that each source must be on either side of the plane that bisects the microphones, Kuo and Peng [59] try to utilize this effect by adding a delay to one of the observed signals. This approach, also based on the SAD algorithm, translates the plane into a cone, making the CTRANC more directional. In their paper, they suggest a delay that is one sample less than the time equivalent of the distance between the microphones. This changes the operation of the CTRANC not dissimilar to that of a beamformer, only picking up signal from a specific direction. However, it varies from a beamformer in that it assumes only one noise point source (for the case of a TITO system).

Mirchandani *et al.* [90] provide an alternative to the feedback SAD algorithm that does not make the assumption that $\nabla_{\mathbf{w}_{2,1}} y_2 = \nabla_{\mathbf{w}_{1,2}} y_1 = 0$. This means that it does not have the biasing problems of the feedback SAD algorithm, but on the other hand is much more complex. In [26], Zinser *et al.* propose a similar approach.

In order to provide an algorithm that performs the correct convergence as described in [26, 90] but without the same computational complexity, Lepauloux *et al.* [73] describe an alternate algorithm that does not assume $\nabla_{\mathbf{w}_{2,1}} y_2 = \nabla_{\mathbf{w}_{1,2}} y_1 = 0$. As opposed to [26, 90], they make the assumption that the filters do not change significantly over time. This allows them to substantially simplify the algorithm, decreasing its computational complexity. They then show that their algorithm matches the performance of [90].

2.4 Summary

A number of approaches to solve the BSS problem are outlined in this review. The main focus of this thesis will be on the procedure known as SAD, as its inherent simplicity and dependence only on SOS potentially lends itself to being a less computationally intensive algorithm than most others mentioned in this review. While it may not achieve the best separation of the mentioned algorithms, its low computational requirements means that it will potentially lend itself better to time-critical applications.

3. MIMO SAD

The *symmetric adaptive decorrelation* (SAD) algorithm (as mentioned in chapter 2) is a method of separating convolutely mixed *sources* based on second-order statistics (SOS). While it has been previously derived for the *two-input two-output* (TITO) case [13], and extended to the *multiple-input multiple-output* (MIMO) case [44], the MIMO algorithm was not derived from first principles but simply an extension of the TITO case. As a result, there has been no real analysis of its convergence properties, nor whether the constraints on the mixing system as described in [13] need to be revised.

Van Gerven and Van Compernolle derive the equations for the TITO SAD algorithm as shown earlier in (2.30). This idea is extended to the MIMO case by Mei and Yin in [44] to get the following update equations.

$$\begin{aligned} w_{i,j}^{m}\left(t+1\right)&=w_{i,j}^{m}\left(t\right)-\frac{\mu}{\sigma_j^2}y_i\left(t\right)y_j\left(t-m\right)\\ w_{j,i}^{m}\left(t+1\right)&=w_{j,i}^{m}\left(t\right)-\frac{\mu}{\sigma_i^2}y_j\left(t\right)y_i\left(t-m\right) \end{aligned} \tag{3.1}$$

where the symbols are represented by the parameters in Fig. 3.6.

However, in order to derive the update equations for the TITO SAD algorithm, there are constraints placed on the mixing system, and these constraints need to be altered in order

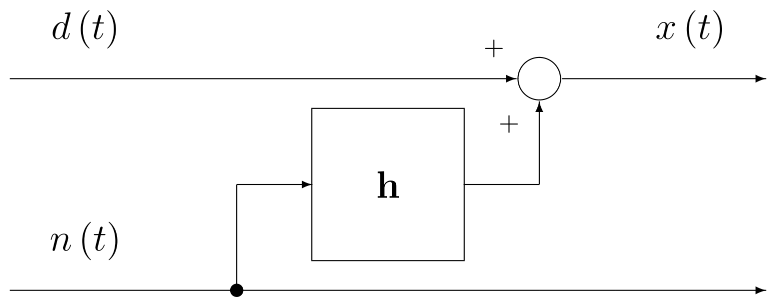


Fig. 3.1: Simple Mixing System

for the extension to a MIMO demixing system to be justifiable. In this chapter we derive the MIMO SAD algorithm and show the new constraints on the mixing system.

3.1 Background Information

The SAD algorithm is based on minimizing the correlation between *outputs* of the mixing system. It is loosely based on the *least mean squares* (LMS) filter, but is less susceptible to problems that arise due to crosstalk from the desired source in the noise estimate. A brief discussion of both the LMS filter and the SAD algorithm follows.

3.1.1 Least Mean Squares Adaptive Filtering

The LMS filter is an adaptive filter that is designed to remove any component of a mixture that is correlated to some known noise reference signal. The structure of the mixing system is assumed to be as shown in Fig. 3.1, where an unpolluted version of the desired signal is not known.

However, because a signal-free noise estimate *is* available, if the cross-filter $\mathbf{h}$ can be found, then it is possible to remove any noise from the mixture resulting in a noise-free though filtered version of the desired signal. Considering the structure in Fig. 3.2, if it is

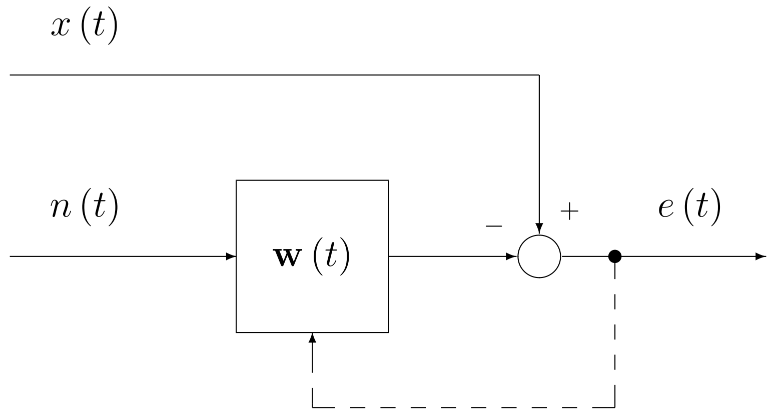


Fig. 3.2: LMS adaptive filter structure.

possible to find an adaptive filter $\mathbf{w}(t)$ that converges such that its output is identical to the *observed signal* at the output of filter $\mathbf{h}$ in Fig. 3.1, then the error $e(t)$ will converge to $d(t)$.

The LMS filter was first proposed by Widrow and Hoff in [100], and finds an estimate of the mixing filter $\mathbf{h}$ by using stochastic gradient descent to find the filter $\mathbf{w}(t)$ that minimizes the power of the error $e(t)$. Because power is a second-order statistic, gradient descent will find the global optimum regardless of the initial conditions. Firstly, the error is defined as

$$e(t){=}x(t) - \mathbf{w}^T(t)\,\mathbf{n}(t)\,. \tag{3.2}$$

To minimize the mean square error (the power of $e(t)$), its gradient must be found with respect to the adaptive filter. Under the assumptions that the adaptive filter, desired signal and noise are all mutually uncorrelated, and that the mean square error can be adequately approximated by its instantaneous estimate, the gradient becomes

$$\frac{\partial E\left[e^2(t)\right]}{\partial \mathbf{w}(t)}{\approx}2e(t)\,\mathbf{x}(t) \tag{3.3}$$

where $E[\cdot]$ denotes the expectation operator, and

$$\mathbf{x}(t) = \begin{bmatrix} x(t) & x(t-1) & \ldots & x(t-L+1) \end{bmatrix}$$

and $L-1$ is the order of the transverse filter $\mathbf{w}(t)$.

Using stochastic gradient descent provides the following update equation for minimizing the power of the output of the adaptive filter

$$\mathbf{w}(t+1) = \mathbf{w}(t) - 2\mu e(t)\mathbf{x}(t) \tag{3.4}$$

where μ is the adaptive step-size, and should be bounded by

$$0 < \mu < \frac{1}{E\left[\mathbf{x}^T(t)\mathbf{x}(t)\right]} \tag{3.5}$$

for stability's sake.

The *normalized least mean squares* (NLMS) [15] algorithm takes advantage of this constraint to find the value of μ which maximizes convergence while maintaining stability. Taking $\mu = \frac{1}{2\mathbf{x}^T(t)\mathbf{x}(t)}$ the new update equations become

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \frac{e(t)\mathbf{x}(t)}{\mathbf{x}^T(t)\mathbf{x}(t)} \tag{3.6}$$

This means that the adaptive step size stays approximately around its most ideal value. Because the LMS algorithm minimizes an *isotropic* cost function, the ideal step size will be halfway between the minimum step size and maximum step size. However, although (3.6) describes an algorithm that maximizes convergence speed, the update term is often scaled down to reduce the chance of instability being introduced by the instantaneous approximation of the scaling coefficient $\frac{1}{E[\mathbf{x}^T(t)\mathbf{x}(t)]}$.

Also, because the LMS algorithm makes the assumption that the power of the filter output $E\left[e^2\left(t\right)\right]$ can be approximated by its instantaneous value $e^2\left(t\right)$, the minimum cost function that (3.4) minimizes constantly shifts about the *actual* values of $\mathbf{w}\left(t\right)$ that minimize the power of $e\left(t\right)$. Because of this, even though a value of μ may satisfy the stability constraint in(3.5), a smaller value of μ will likely converge to a better estimate of the actual filter weights $\mathbf{h}$, though convergence will take longer. This error in the cost function is termed *misadjustment* by Widrow *et al.* in [101] and should be considered along with the stability of the algorithm when choosing a suitable value for μ.

3.1.2 The Crosstalk Resistant Adaptive Noise Canceler

The *crosstalk resistant adaptive noise canceller* (CTRANC) was first developed in [90] to nullify the effects that crosstalk has on the operation of a LMS filter. Rather than having a signal-free estimate, it consists of two cross-coupled LMS filters to adjust for any crosstalk from the desired source. Also, due to the fact that it is symmetric, the terms 'desired' signal and 'noise' signal now have less relevance to the source signals.The structure of the CTRANC is a feedback structure as shown in Fig. 3.3.

The update equations are

$$\begin{aligned}\mathbf{w}_1\left(t+1\right)&=\mathbf{w}_1\left(t\right)-2\mu y_1\left(t\right)\frac{\partial y_1\left(t\right)}{\partial \mathbf{w}_1\left(t\right)}\\ \mathbf{w}_2\left(t+1\right)&=\mathbf{w}_2\left(t\right)-2\mu y_2\left(t\right)\frac{\partial y_2\left(t\right)}{\partial \mathbf{w}_2\left(t\right)}\end{aligned} \tag{3.7}$$

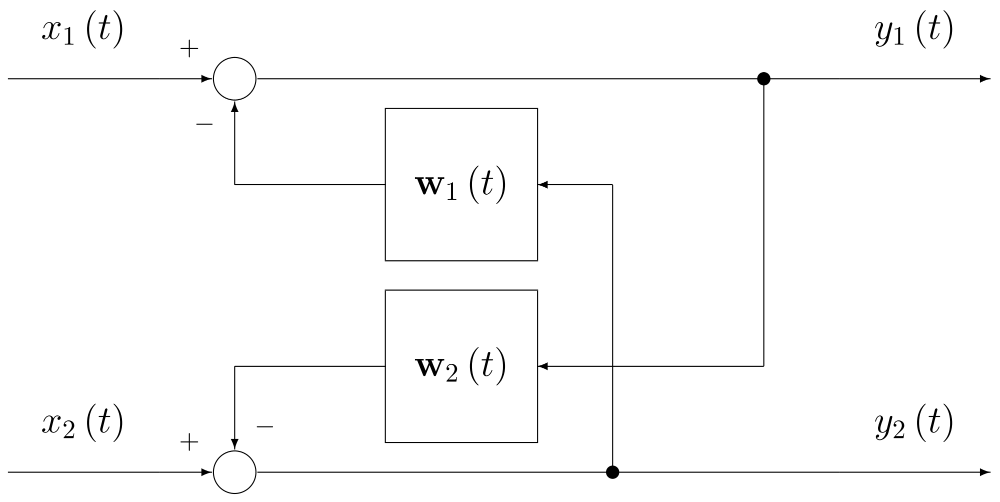


Fig. 3.3: A feedback CTRANC.

which, like LMS, uses stochastic gradient descent to minimize the power of the outputs with respect to the demixing filters. However, the CTRANC now needs two update equations, as it no longer assumes that the noise estimate is signal-free.

The gradient term in (3.7) is shown in [90] to be

$$\frac{\partial y_i(t)}{\partial \mathbf{w}_i(t)} = C_0(t)\left[\sum_{k=1}^{2L} C_{i,k}(t)\,\frac{\partial y_i(t-k)}{\partial w_i^k(t-k)} - \mathbf{y}_j(t)\right] \tag{3.8}$$

where

$$\begin{aligned}
\mathbf{y}_j(t) &= \begin{bmatrix} y_j(t) & y_j(t-1) & \dots & y_j(t-L+1) \end{bmatrix}^T, \\
C_{i,k}(t) &= \begin{cases} \sum\limits_{l=1}^{k} w_i^l(t)\, w_j^{k-l}(t-l) & 1 \le k \le L \\ \sum\limits_{l=k-L}^{L} w_i^l(t)\, w_j^{k-l}(t-l) & L+1 \le k \le 2L \end{cases}, \\
C_0(t) &= \frac{1}{1 - w_1^0(t)\, w_2^0(t)}
\end{aligned} \tag{3.9}$$

and

$$j = \begin{cases} 1 & i = 2 \\ 2 & i = 1 \end{cases}$$

While not specified in [90], the feedback structure of the CTRANC gives rise to a causality problem. In order to find $y_1(t)$, one must know $y_2(t)$, yet in order to find $y_2(t)$, one must know $y_1(t)$. This paradox is easily overcome by making the assumption that either $w_1^0(t) = 0$ or $w_2^0(t) = 0$, but to retain the symmetric nature of the separator, we will assume that $w_1^0(t) = w_2^0(t) = 0$. From (3.9), this makes $C_0(t) = 1$, and simplifies (3.8) to

$$\frac{\partial y_i(t)}{\partial \mathbf{w}_i(t)} = \left[\sum_{k=1}^{2L} C_{i,k}(t) \frac{\partial y_i(t-k)}{\partial w_i^k(t-k)} - \mathbf{y}_j(t) \right] \tag{3.10}$$

Once the size of the filter becomes large, the memory and computational requirements of the CTRANC also increase significantly. In [13], Van Gerven and Van Compernolle address this problem by proposing that the separating system be a feedforward structure. This is discussed further in the following section.

3.1.3 Symmetric Adaptive Decorrelation

While the CTRANC is only proposed using the feedback structure of Fig. 3.3, the SAD proposed by Van Gerven and Van Compernolle in [13] is also implemented using the feedforward structure in Fig. 3.4 for the demixing system.

Although the CTRANC is derived by using stochastic gradient descent to minimize the joint power of the outputs, the SAD algorithm is derived by using a Newton zero-search on the *cross-correlation* of the outputs. For the feedforward structure, a Newton zero-search of

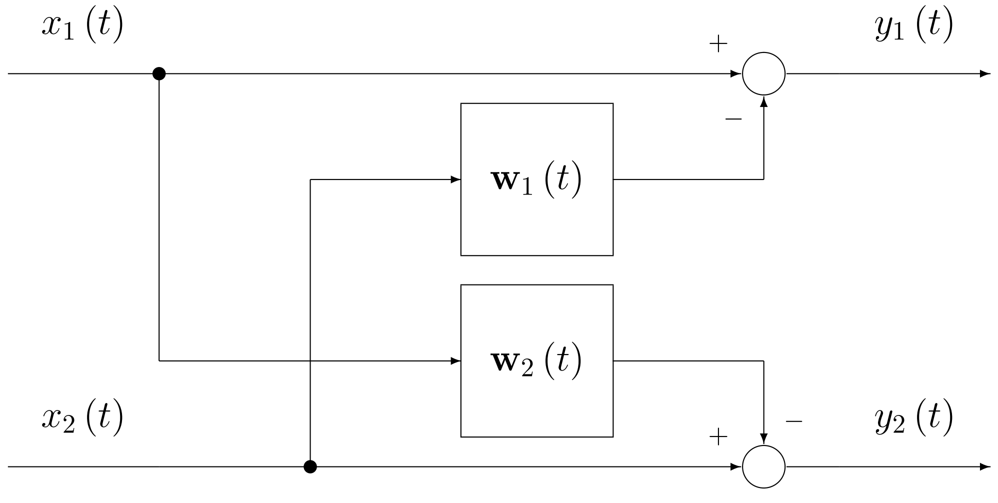


Fig. 3.4: The feedforward demixing system.

the cross-correlation results in identical update equations to stochastic gradient descent of the joint output power.

The fundamental form of the update equations as described in [13] are

$$\begin{aligned}\mathbf{w}_1(t+1) &= \mathbf{w}_1(t+1) + \mu_1 y_1(t)\,\mathbf{y}_2(t)\\ \mathbf{w}_2(t+1) &= \mathbf{w}_2(t+1) + \mu_2 y_2(t)\,\mathbf{y}_1(t)\end{aligned} \tag{3.11}$$

where

$$\begin{aligned}\mathbf{y}_1(t) &= [y_1(t), y_1(t-1), \ldots, y_1(t-L+1)]\\ \mathbf{y}_2(t) &= [y_2(t), y_2(t-1), \ldots, y_2(t-L+1)]\end{aligned} \tag{3.12}$$

and $L-1$ is the filter order.

The main problem with the algorithm described by Equation (3.11) is that it assumes that filter $\mathbf{w}_1$ has no effect on y_2 and that $\mathbf{w}_2$ has no effect on y_1. While this may be true for the feedforward case, when implemented using a feedback structure, the SAD algorithm is prone to biasing. However, the update equations of the CTRANC provided by Mirchandani *et al.* in [90] *do* include a gradient term that accounts for the dependence of y_2 on $\mathbf{w}_1$ and y_1 on $\mathbf{w}_1$. Comparing the CTRANC update equations (3.7) to the SAD update equations (3.11), it can be seen that the SAD algorithm arrives with a gradient term of

$$\frac{\partial y_i\left(t\right)}{\partial \mathbf{w}_i\left(t\right)}=-\mathbf{y}_j\left(t\right) \tag{3.13}$$

which is incorrect when comparing it to the actual gradient term in (3.10). Although this introduces a potential bias in the cost surface for the feedback SAD, it is computationally expensive to calculate the correct gradient term. Lepauloux *et al.* [73] propose a more efficient version of the CTRANC by assuming that the room transfer function does not significantly change over time, roughly halving the number of operations. The resulting algorithm, however, is still significantly more computationally complex than the SAD algorithm.

3.2 Derivation of MIMO SAD

While the TITO SAD algorithm has been derived for both a feedforward and feedback demixing structure, the update term in the adaptive algorithm cannot be properly justified for the feedback case [73]. For this reason, only the feedforward MIMO SAD will be discussed.

For the derivation, only the key steps have been included in the body of this work, for a step-by-step extended derivation, see Appendix B.1. Firstly, we will define the following variables:

$$s_i(t) \triangleq \text{source } i \text{ at time } t$$

$$x_i(t) \triangleq \text{observed signal } i \text{ at time } t$$

$$y_i(t) \triangleq \text{output } i \text{ at time } t$$

$$\mathbf{s}_i(t) \triangleq \begin{bmatrix} s_i(t) & s_i(t-1) & \ldots & s_i(t-L+1) \end{bmatrix}^T$$
$$\mathbf{x}_i(t) \triangleq \begin{bmatrix} x_i(t) & x_i(t-1) & \ldots & x_i(t-L+1) \end{bmatrix}^T$$
$$\mathbf{y}_i(t) \triangleq \begin{bmatrix} y_i(t) & y_i(t-1) & \ldots & y_i(t-L+1) \end{bmatrix}^T$$

$\mathbf{h}_{i,j} \triangleq$ a vector containing the mixing filter taps for the filter from source j to observed signal i

$\mathbf{w}_{i,j}(t) \triangleq$ a vector containing the demixing filter taps for the filter from observed signal j to output i at time t

For N sources and N observed signals, a linear mixture can be described by the following equation

$$x_i(t) = \sum_{j=0}^{N-1} \mathbf{s}_j^T(t)\, \mathbf{h}_{i,j} \tag{3.14}$$

where $x_i(t)$ is the i^{th} observed signal, $\mathbf{h}_{i,j}$ is the vector of mixing filter taps from source j to observed signal i, and

$$\mathbf{s}_j(t) = \begin{bmatrix} s_j(t) & s_j(t-1) & \ldots & s_j(t-L+1) \end{bmatrix}$$

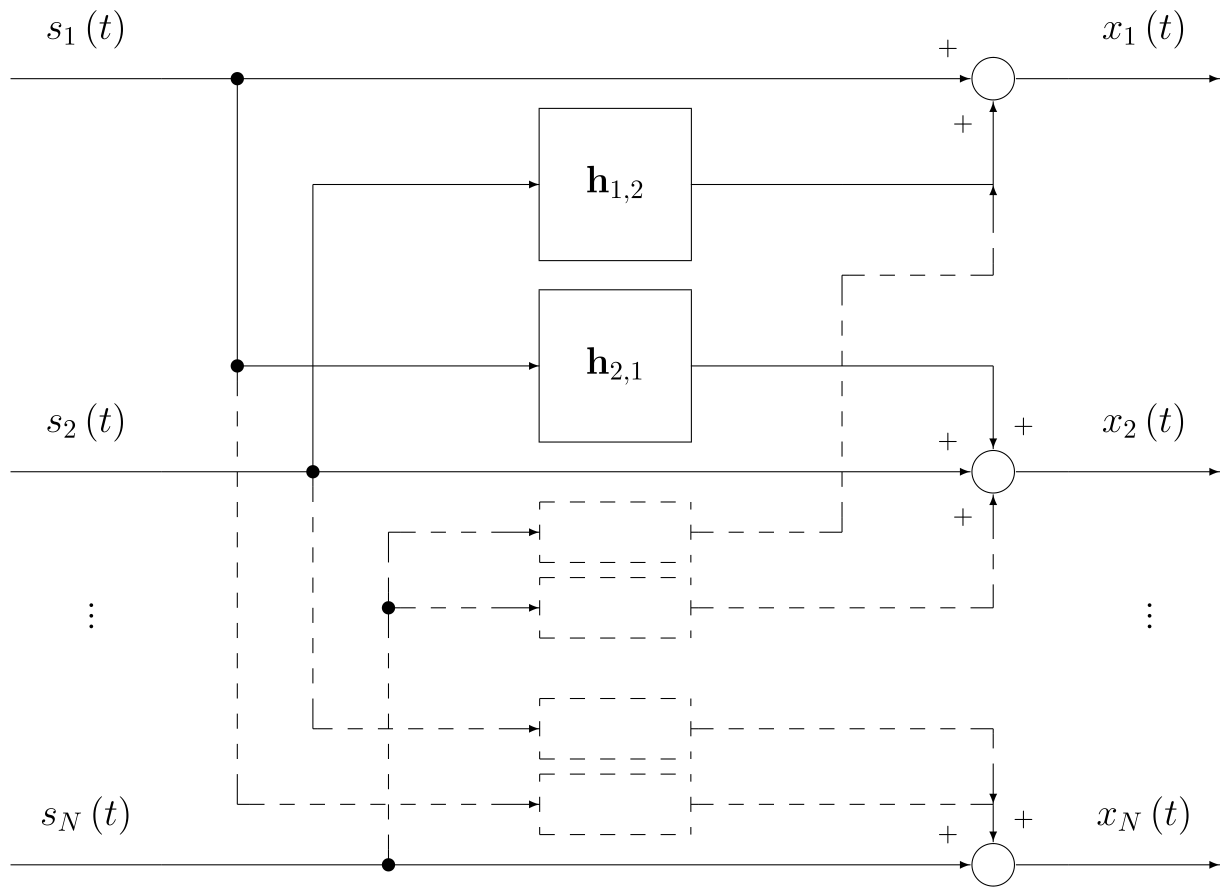


Fig. 3.5: The simplified mixing system.

where $s_j(t)$ is the j^{th} source at time t and $L-1$ is the order of the mixing filter.

This is then simplified to the mixing system shown in Fig. 3.5, where the mixing filters from source i to observed signal i are assumed to be unity-gain non-*convolutive* channels. For this to be possible, the actual channels from source i to observed signal i must be minimum phase, as the existance of the simplified mixing filters $h_{i,j}$ requires the invertibility of the channels from one source to the corresponding microphone. This corresponds to what is outlined in [99], where Lindgren and Broman show that separation based on SOS is possible if the mixing system is minimum phase.

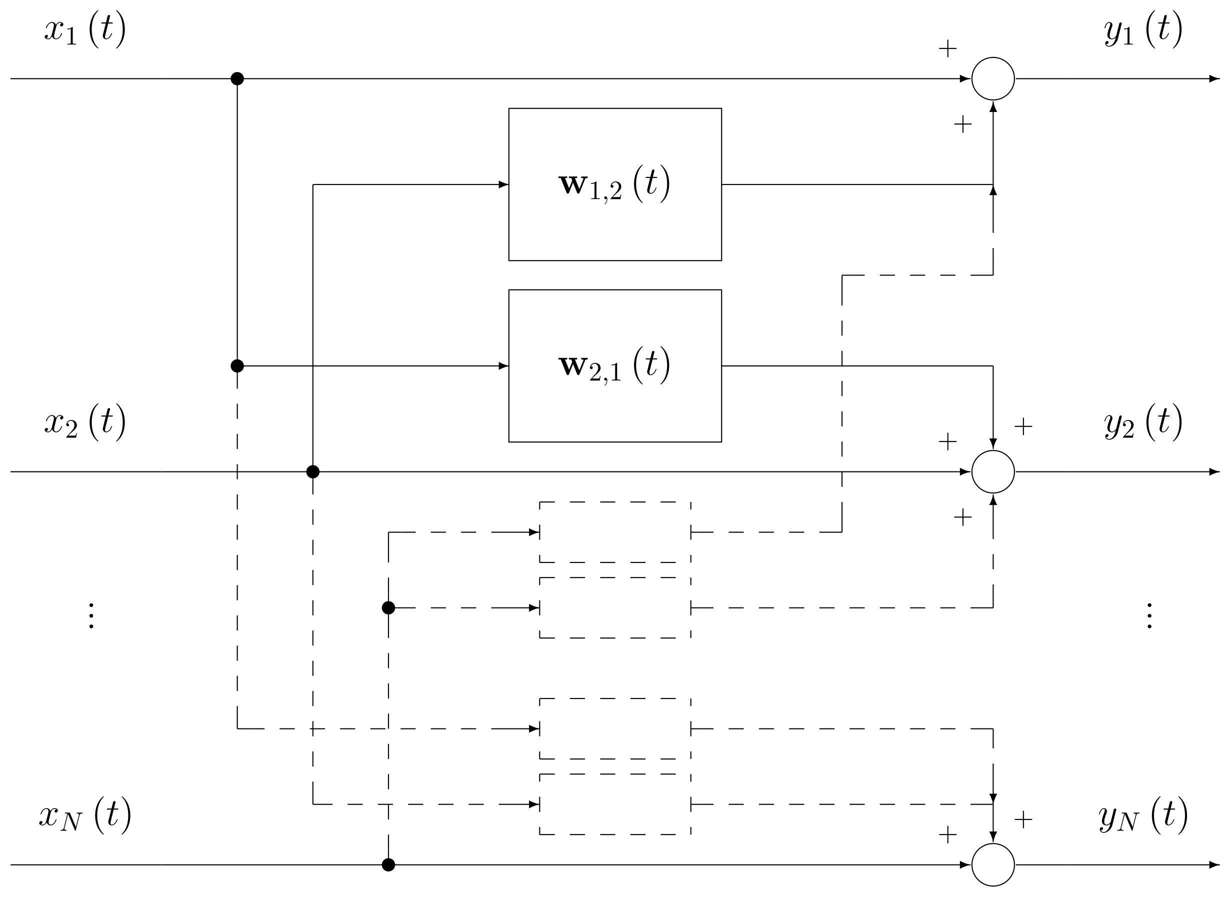


Fig. 3.6: The demixing system.

Likewise a *demixing* system can be written as

$$y_i(t)=\sum_{j=0}^{N-1} \mathbf{x}_j^T(t)\,\mathbf{w}_{i,j}(t) \tag{3.15}$$

For equations (3.14) and (3.15) to properly match the mixing and demixing systems of Fig. 3.5 and Fig. 3.6, we define the direct channels as

$$\mathbf{h}_{i,i} = \mathbf{w}_{i,i}(t)=\begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix}^T.$$

If suitable constraints are placed on the mixing system, one possible way of minimizing the crosstalk between outputs of the demixing system is by minimizing the cross-correlation between the outputs. Between two outputs $y_i(t)$ and $y_j(t)$ the cross-correlation is defined as

$$C_{y_j,y_i}(m) = E\left[y_i(t+m)\, y_j(t)\right] \qquad \forall m \in \mathbb{Z} \tag{3.16}$$

To remove crosstalk introduced by sources $s_i(t)$ and $s_j(t)$ from output $y_j(t)$ and $y_i(t)$ respectively, it is necessary to find the values of the filter vectors $\mathbf{w}_{i,j}(t)$ and $\mathbf{w}_{i,j}(t)$ such that the cross-correlation defined in (3.16) is minimized.

However, due to the simplification of the mixing system represented by Fig. 3.5, we can see that the values of m in the cross-correlation are bounded such that we can redefine the cross-correlation between $y_i(t)$ and $y_j(t)$ due to the crosstalk from $s_j(t)$ into $x_i(t)$ as

$$C_{y_j,y_i}(m) = E\left[y_i(t)\, y_j(t-m)\right] \qquad 0 < m < L. \tag{3.17}$$

Given that the k^{th} element of the filter $\mathbf{w}_{i,j}$ only directly affects the $x_j(t-k)$ component of $y_i(t)$, it also only directly affects $C_{y_j,y_i}(k)$.

Thus (3.17) should be used as the cost function for updating the filter weight $w_{i,j}^{m}(t)$. Newton's method states that in order to converge to a root of a function $f(x)$, the value can be updated according to the following equation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{3.18}$$

which, after substituting the relevant variables, becomes

$$w_{i,j}^{m}(t+1) = w_{i,j}^{m}(t) - \frac{C_{y_i y_j}(m)}{C'_{y_i y_j}(m,t)} \tag{3.19}$$

where $w_{i,j}^{m}(t)$ is the m^{th} element of $\mathbf{w}_{i,j}(t)$ and

$$C'_{y_iy_j}(m,t)=\frac{\partial C_{y_iy_j}(m)}{\partial w_{i,j}^{m}(t)} \tag{3.20}$$

Substituting (3.14) and (3.15) into (3.17) then rearranging according to Appendix B.1.1 gives

$$C_{y_iy_j}(m)=\sum_{p=0}^{N-1}\sigma_p^2\sum_{k=0}^{N-1}\sum_{n=0}^{L-1}\sum_{l=0}^{N-1}\sum_{o=0}^{L-1}\sum_{q=0}^{L-1}h_{k,p}^{q}w_{i,k}^{n}(t)\,h_{l,p}^{n+q-m-o}w_{j,l}^{o}(t-m) \tag{3.21}$$

where σ_j^2 is the power of the j^{th} source.

However, when $p=k$,

$$h_{k,p}^{q}=\begin{cases}1, & q=0\\ 0, & q\neq 0\end{cases}$$

and when $p\neq k$,

$$h_{k,p}^{0}=0$$

After this substitution and subsequent rearrangement as shown in Appendix B.1.2, (3.21) becomes

$$
\begin{aligned}
C_{y_j,y_i}(m) =& \sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1} \sigma_p^2 \sum_{q=1}^{2L-2} \delta_{p,i}^{q}(t)\,\delta_{p,j}^{q-m}(t-m) \\
&+\sigma_i^2 \sum_{\substack{k=0\\k\neq i}}^{N-1} \sum_{n=1}^{L-1} \sum_{q=1}^{2L-2} h_{k,i}^{q-n} w_{i,k}^{n}(t)\,\delta_{i,j}^{q-m}(t-m) \\
&+\sigma_j^2 \sum_{\substack{k=0\\k\neq j}}^{N-1} \sum_{n=1}^{L-1} \sum_{q=1}^{2L-2} h_{k,j}^{q-m-n} w_{j,k}^{n}(t-m)\,\delta_{j,i}^{q}(t) \\
&+\sigma_j^2 \delta_{j,i}^{m}(t) \qquad (3.22)
\end{aligned}
$$

where

$$
\delta_{i,j}^{m}(t) = h_{i,j}^{m} + w_{i,j}^{m}(t) + \sum_{\substack{k=0\\k\neq i\\k\neq j}}^{N-1} \sum_{n=1}^{L-1} h_{k,j}^{m-n} w_{i,k}^{n}(t) \qquad (3.23)
$$

defines the error of the filter tap from its optimum value.

We will now find the derivative of this with respect to the mth separating filter $w_{j,i}^{m}$

$$
\frac{\partial C_{y_i,y_j}(m)}{\partial w_{i,j}^{m}} = \sum_{\substack{p=0\\p\neq j}}^{N-1} \sigma_p^2 \sum_{q=1}^{L-1} h_{p,j}^{q} \delta_{p,j}^{q}(t-m) + \sigma_j^2 \qquad (3.24)
$$

Detailed information on finding this gradient term is given in Appendix B.1.3.

Substituting (3.24) into (3.19) in order to find the $w_{i,j}^{m}$ that gives zero cross-correlation results in

$$w_{i,j}^{m}\left(t+1\right)=w_{i,j}^{m}\left(t+1\right)-\frac{y_{i}\left(t\right)y_{j}\left(t-m\right)}{\sigma_{j}^{2}+\sum_{\substack{p=0\\p\neq j}}^{N-1}\sigma_{p}^{2}\left[\sum_{q=1}^{L-1}h_{j,p}^{q}\delta_{j,p}^{q}\left(t-m\right)\right]} \tag{3.25}$$

Because the cross-correlation is a linear function of the tap weights, approximating the denominator of the update term in Equation (3.25) as σ_j^2 will still result in convergence provided that

$$\left|\sum_{\substack{p=0\\p\neq j}}^{N-1}\sigma_{p}^{2}\left[\sum_{q=1}^{L-1}h_{j,p}^{q}\delta_{j,p}^{q}\left(t-m\right)\right]\right|<\sigma_{j}^{2} \tag{3.26}$$

Under the assumption that the powers of the source signals are approximately equal, we can simplify (3.26) to

$$\left|\sum_{\substack{p=0\\p\neq j}}^{N-1}\sigma_{p}^{2}\left[\sum_{q=1}^{L-1}h_{j,p}^{q}\delta_{j,p}^{q}\left(t-m\right)\right]\right|<1 \tag{3.27}$$

When substituting $N=2$, $\delta_{i,j}^{m}\left(t\right)$ becomes $\left(w_{i,j}^{m}\left(t\right)+h_{i,j}^{m}\right)$, thus the constraint in Equation (3.27) simplifies to

$$\left|\sum_{q=1}^{L-1}h_{j,i}^{q}\left(w_{j,i}^{q}\left(t-m\right)+h_{j,i}^{q}\right)\right|<1$$

which is identical to the constraint given for the TITO decorrelator in [13].

With this constraint, (3.25) simplifies to

$$w_{i,j}^{m}\left(t+1\right)=w_{i,j}^{m}\left(t\right)-\frac{y_{i}\left(t\right)y_{j}\left(t-m\right)}{\sigma^{2}}$$

and finally an adaptive step-size constant is added to improve stability

$$w_{i,j}^{m}\left(t+1\right)=w_{i,j}^{m}\left(t\right)-\frac{\mu}{\sigma^2}y_i\left(t\right)y_j\left(t-m\right) \tag{3.28}$$

The pseudocode for this algorithm is shown in Program 1.

Program 1 MIMO SAD Pseudocode

```
Superscript $^T$ denotes the transpose operation.
$n$                                   ▷ the number of observed signal samples
$N$                                   ▷ the number of observed signals
$L$                                   ▷ the filter size
$x_1, x_2, \ldots, x_N$               ▷ the observed signals
$y_1, y_2, \ldots, y_N$               ▷ the outputs
$\mathbf{w}_{i,j}$                    ▷ the vectors containing the filter weights

for $t = 0 \to n$ do
    for $i = 1 \to N$ do
        $\mathbf{x}_i(t)$ = the next $L$ samples of $x_i$ in reverse order.
    end for

    for $i = 1 \to N$ do
        $y_i(t) = 0$
        for $j = 1 \to N, j \neq i$ do
            $y_i(t) += \mathbf{w}_{i,j}^T(t)\,\mathbf{x}_j(t)$
        end for

    end for
    for $i = 1 \to N$ do

        for $j = 1 \to N, i \neq j$ do
            $\mathbf{w}_{i,j}(t) = \mathbf{w}_{i,j}(t) - y_i(t)\,\mathbf{y}_j(t)$
        end for
    end for
end for
```

3.3 Convergence of MIMO SAD

Upon investigation of the mixing and demixing systems shown in Fig. 3.5 and Fig. 3.6, one may see that the demixing system becomes the inverse of the mixing system when

$$w_{i,j,\mathrm{opt}}^{m}(t) = -h_{i,j}^{m} - \sum_{\substack{l=0\\ l\neq i\\ l\neq j}}^{N-1} \sum_{o=1}^{m-1} h_{i,l}^{m-o} w_{l,j}^{o}(t-m) \quad \forall i, j \neq i$$

Therefore, we define the deviation from the ideal solution as

$$\begin{aligned} w_{i,j}^{m}(t) - w_{i,j,\mathrm{opt}}^{m}(t) &= w_{i,j}^{m}(t) + h_{i,j}^{m} \\ &\quad + \sum_{\substack{l=0\\ l\neq i\\ l\neq j}}^{N-1} \sum_{o=1}^{m-1} h_{i,l}^{m-o} w_{l,j}^{o}(t-m) \quad \forall i, j \neq i \\ &= \delta_{i,j}^{m}(t) \end{aligned} \tag{3.29}$$

which will be zero on separation.

One may note that $\delta_{i,j}^{m}(t)$ in (3.23) is slightly different from that defined in (3.29) in that the limit on the second sum is $L-1$ as compared to $m-1$ in the earlier definition. The justification for this is that because the mixing filters are strictly causal, $h_{i,l}^{m-o} = 0$ when $o \geq m$.

For convergence in the mean, if

$$\left| E\left[\delta_{i,j}^{m}(t)\right] \right| \leq \varepsilon \tag{3.30}$$

then the following must also be true,

$$\left|E\left[\delta_{i,j}^{m}\left(t+k\right)\right]\right|<\varepsilon \qquad \forall k\in\mathbb{Z}^{+} \tag{3.31}$$

for any real positive ε.

Using the definition in (3.23) for $\delta_{i,j}^{m}(t)$, and under the assumption that $w_{l,j}^{o}(t)\approx w_{l,j}^{o}(t+1)$, $l\neq i,j$, the demixing filter tap update equation in (3.28) then becomes

$$\delta_{i,j}^{m}\left(t+1\right)=\delta_{i,j}^{m}\left(t\right)-\frac{\mu}{\sigma^{2}}y_{i}\left(t\right)y_{j}\left(t-m\right) \tag{3.32}$$

Taking expected values, and using (3.22) as the cross-correlation between y_i and y_j, the expectation of (3.32) reduces to

$$E\left[\delta_{i,j}^{m}\left(t+1\right)\right]=E\left[\delta_{i,j}^{m}\left(t\right)-\mu\left\{\delta_{i,j}^{m}\left(t\right)+\tilde{\gamma}_{i,j}^{m}\left(t\right)\right\}\right] \tag{3.33}$$

where

$$\begin{aligned}\tilde{\gamma}_{i,j}^{m}\left(t\right)=\Bigg[&\sum_{\substack{p=0\\p\neq j\\p\neq i}}^{N-1}\sum_{q=1}^{2L-2}\delta_{p,j}^{q}\left(t\right)\delta_{p,i}^{q-m}\left(t-m\right)\\&+\sum_{q=1}^{2L-2}\delta_{j,i}^{q-m}\left(t-m\right)\sum_{\substack{k=0\\k\neq j}}^{N-1}\sum_{n=1}^{L-1}h_{j,k}^{q-n}w_{k,j}^{n}\left(t\right)\\&+\sum_{q=1}^{2L-2}\delta_{i,j}^{q}\left(t\right)\sum_{\substack{k=0\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}h_{i,k}^{q-m-n}w_{k,i}^{n}\left(t-m\right)\Bigg]\end{aligned} \tag{3.34}$$

Taking absolute values of (3.33) gives the following inequality

$$\left|E\left[\delta_{i,j}^{m}(t+1)\right]\right|\leq\left|(1-\mu)E\left[\delta_{i,j}^{m}(t)\right]\right| \\ +\left|\mu E\left[\tilde{\gamma}_{i,j}^{m}\right]\right| \tag{3.35}$$

If all the weights of all of the filters have fallen to within ε of the desired solution, $\tilde{\gamma}_{i,j}^{m}(t)$ can be linearized by ignoring all second order terms of ε. This results in the following approximation

$$\tilde{\gamma}_{i,j}^{m}(t)\approx\varepsilon\gamma_{i,j}^{m}(t)$$

where

$$\gamma_{i,j}^{m}(t)=\left|\sum_{\substack{k=0\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}\sum_{q=1}^{L-1-m}h_{i,k}^{q}w_{k,i}^{n}(t-m) \right. \\ \left. +\sum_{\substack{k=0\\k\neq j}}^{N-1}\sum_{n=1}^{L-1}\sum_{q=1}^{L-1}h_{j,k}^{q}w_{k,j}^{n}(t)\right| \tag{3.36}$$

and in general,

$$\left|\tilde{\gamma}_{i,j}^{m}(t)\right|\leq\varepsilon\left|\gamma_{i,j}^{m}(t)\right| \tag{3.37}$$

Substituting (3.37) into the inequality in (3.35) results in

$$\left|E\left[\delta_{j,i}^{m}(t+1)\right]\right|\leq\left|(1-\mu)E\left[\delta_{i,j}^{m}(t)\right]\right| \\ +\varepsilon\left|\mu E\left[\gamma_{i,j}^{m}(t)\right]\right| \tag{3.38}$$

Assuming that $E\left[\delta_{i,j}^{m}(t)\right]$ has also fallen to within ε of the desired solution, (3.38) reduces to

$$\left|E\left[\delta_{i,j}^{m}\left(t+1\right)\right]\right|\leq\left\{\left|\left(1-\mu\right)\right|+\left|\mu E\left[\gamma_{i,j}^{m}\left(t\right)\right]\right|\right\}\varepsilon \tag{3.39}$$

In order for convergence in the mean to be satisfied, (3.31) must be satistfied. From (3.39), it can be seen that convergence to the desired solution will occur when

$$\left|\mu E\left[\gamma_{i,j}^{m}\left(t\right)\right]\right|-1<1-\mu<1-\left|\mu E\left[\gamma_{i,j}^{m}\left(t\right)\right]\right| \tag{3.40}$$

There is no solution when μ is negative or zero, therefore (3.40) simplifies to

$$\left|E\left[\gamma_{i,j}^{m}\left(t\right)\right]\right|<1<\frac{1}{\mu}-\left|E\left[\gamma_{i,j}^{m}\left(t\right)\right]\right| \tag{3.41}$$

which gives the following constraints on $\gamma_{i,j}^{m}\left(t\right)$ and μ

$$\left|E\left[\gamma_{i,j}^{m}\left(t\right)\right]\right|<1 \tag{3.42}$$

$$0<\mu<\frac{2}{1+\left|E\left[\gamma_{i,j}^{m}\left(t\right)\right]\right|} \tag{3.43}$$

These are also satisfied with the following simpler constraints

$$\left|\gamma_{i,j}^{m}\left(t\right)\right|<1 \tag{3.44}$$

$$0<\mu<1 \tag{3.45}$$

However, this only guarantees convergence *in the mean*. It is still possible to have a system that converges in the mean but where the error is oscillating from a negative extreme to a positive extreme. Therefore, in order to guard against this possibility, we must also show that the magnitude of the error does not diverge.

The following constraint is used to restrict the error's magnitude: if

$$\left|E\left\{\left[\delta_{i,j}^{m}(t)\right]^{2}\right\}\right|\leq\varepsilon^{2} \tag{3.46}$$

then the following must be true,

$$\left|E\left\{\left[\delta_{i,j}^{m}(t+k)\right]^{2}\right\}\right|<\varepsilon^{2} \qquad \forall k\in\mathbb{Z}^{+} \tag{3.47}$$

for any real positive ε^{2}.

Using (3.33), we get

$$\begin{aligned}E\left[\left\{\delta_{i,j}^{m}(t+1)\right\}^{2}\right]=&E\left[\left\{\delta_{i,j}^{m}(t)\right.\right.\\&\left.\left.-\mu\left[\delta_{i,j}^{m}(t)+\tilde{\gamma}_{i,j}^{m}(t)\right]\right\}^{2}\right]\\=&E\left[\left\{\delta_{i,j}^{m}(t)\right\}^{2}\right]\left(1-2\mu+\mu^{2}\right)\\&-2\mu E\left[\delta_{i,j}^{m}(t)\,\tilde{\gamma}_{i,j}^{m}(t)\right](1-\mu)\\&+\mu^{2}E\left[\left\{\tilde{\gamma}_{i,j}^{m}(t)\right\}^{2}\right]\end{aligned} \tag{3.48}$$

But under the approximation made of $\tilde{\gamma}_{i,j}^{m}(t)$ in (3.37) and the constraints placed on $\gamma_{i,j}^{m}(t)$ and μ in (3.44) and (3.45), (3.48) can also be written as the following inequality

$$\begin{aligned}E\left[\left\{\delta_{i,j}^{m}(t+1)\right\}^{2}\right]<&E\left[\left\{\delta_{i,j}^{m}(t)\right\}^{2}\right]\left(1-2\mu+\mu^{2}\right)\\&+2\mu\varepsilon\left|E\left[\left\{\delta_{i,j}^{m}(t)\right\}\right]\right|(1-\mu)\\&+\mu^{2}\varepsilon^{2}\end{aligned} \tag{3.49}$$

If we make the assumption that the mean square error of the filter taps has fallen to within ε^2, then the maximum possible value for $\left|E\left[\delta_{i,j}^m(t)\right]\right|$ is ε. Therefore, if (3.49) is true, the following is also true:

$$\begin{aligned} E\left[\left\{\delta_{i,j}^m(t+1)\right\}^2\right] &< \left(1-2\mu+2\mu^2\right)\varepsilon^2 \\ &\quad +2\mu\varepsilon^2(1-\mu) \\ &< \varepsilon^2 \end{aligned} \tag{3.50}$$

Therefore, under the constraints on $\gamma_{i,j}^m(t)$ and μ in (3.44) and (3.45), asymptotic stability is guaranteed.

3.4 Experiments

In order to evaluate the performance of the proposed algorithm, the following experiment was set up. Four speakers and four mics were set up in a 5m×3m room, as is shown in Fig. 3.7. Human speech, taken from the 'Lunatick-20080326-cc.tgz' package from the VoxForge speech corpus, was played through one of the speakers, and independent samples of car assembly line noise, taken from the NOISEX database, were played through the other three speakers. In Fig. 3.7, source 1 is the speech source; sources 2, 3, and 4 are all noise sources.

Each of the sources were played independently, then mixed later with appropriate time-lags. The reason for doing the experiment this way rather than playing all four sources simultaneously was that the SNR of the observed signals and separated signals could be

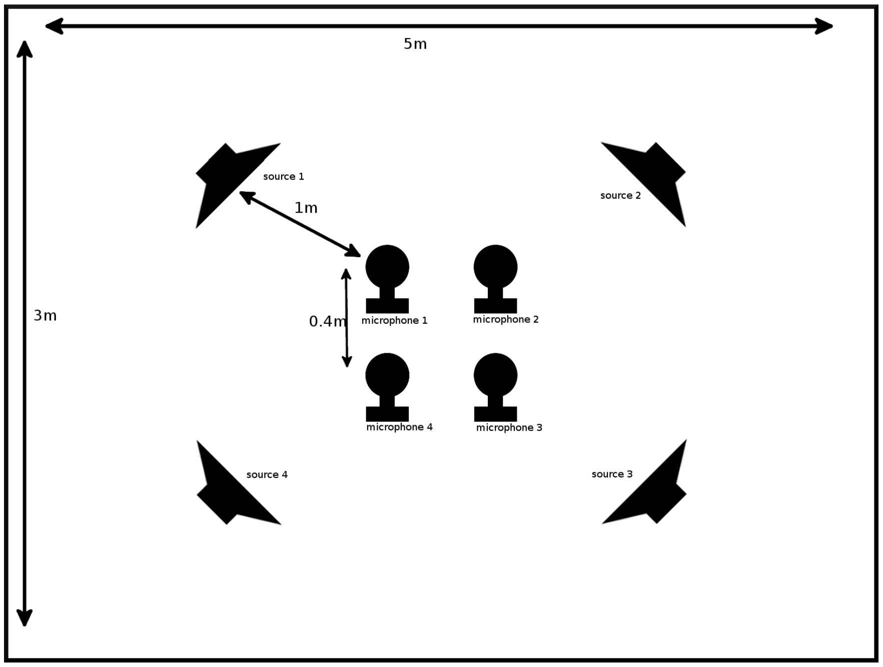


Fig. 3.7: The recording layout.

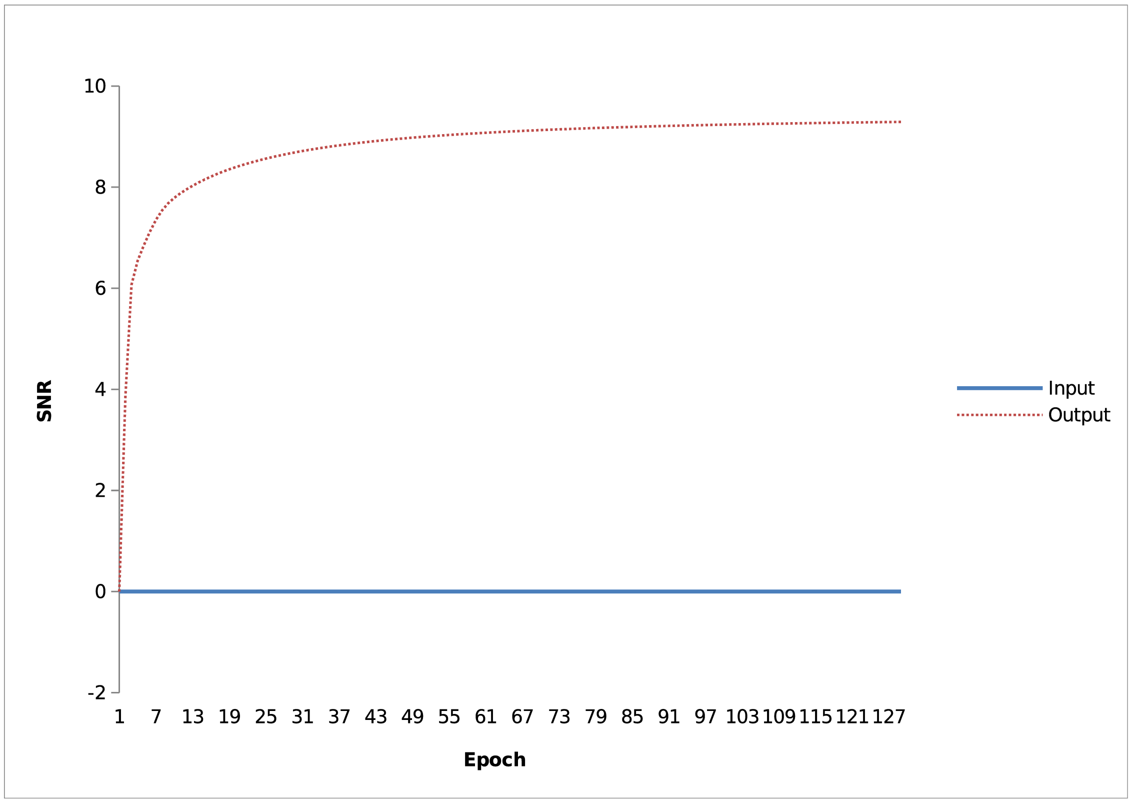


Fig. 3.8: The *signal to noise ratio* (SNR) converging over time.

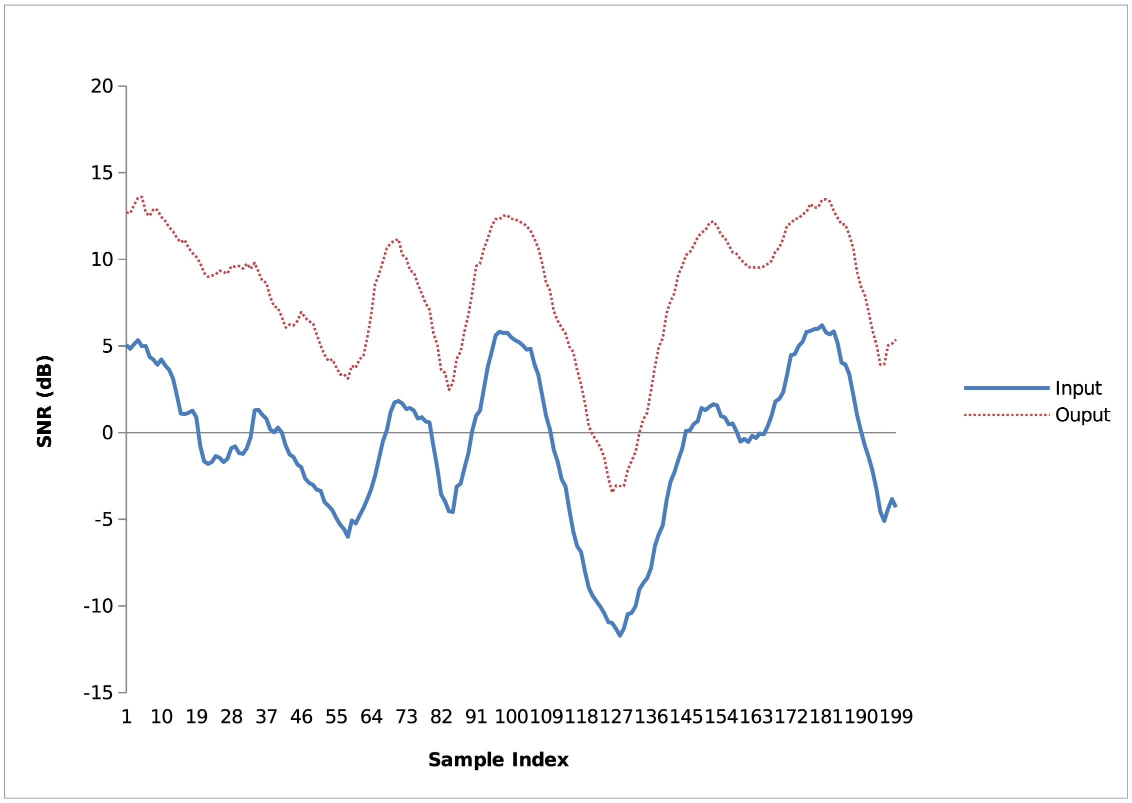


Fig. 3.9: The SNR over time for the final run.

better evaluated. It was calculated using

$$\text{SNR} = 10 \log_{10} \left(\frac{P_S}{P_N} \right) \tag{3.51}$$

where P_S is the power of the speech component of the signal and P_N was the power of the noise component of the signal. Because of the recording technique, the actual power of the speech and noise components was available, and there was no need for any approximations.

In each channel, the desired and noise components were each mixed such that the average SNR of each observed signal was $0dB$. A four second sample of each mixture was passed into the separating system. After the first pass through the data, the final separating filter weights were stored and then used as the initial values for a second pass. This iterative process continued until the SNR had converged to a steady state. The SNR after each iteration is plotted against the number of pass-throughs in Fig. 3.8.

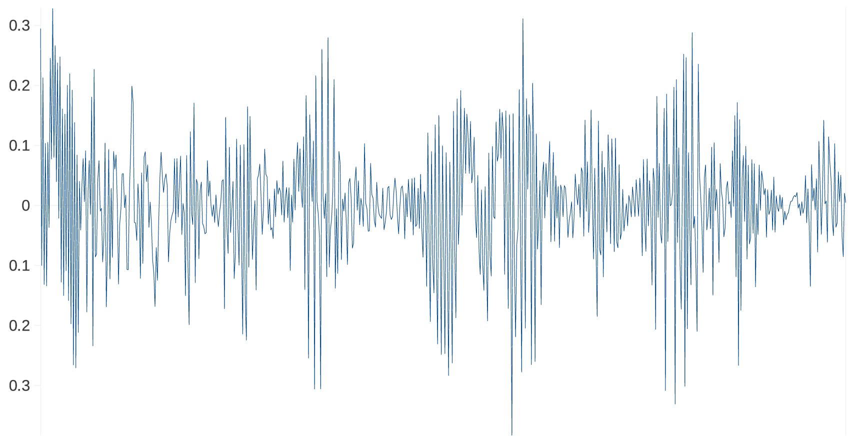
After the final pass-through, the SNR was calculated on the output using a sliding window of length 2000. A comparison to the observed signal SNR is shown in Fig. 3.9. Of note is the fact that the increase in SNR is moderately invariant to the observed signal SNR.

The actual waveform of the desired signal, mixture, and separated signal are shown in figures 3.10a, 3.10b, and 3.10c respectively. One can see that the output in Fig. 3.10c more closely matches the desired signal in Fig. 3.10a than the observed signal in Fig. 3.10b. It also appears to be substantially less noisy than the observed signal.

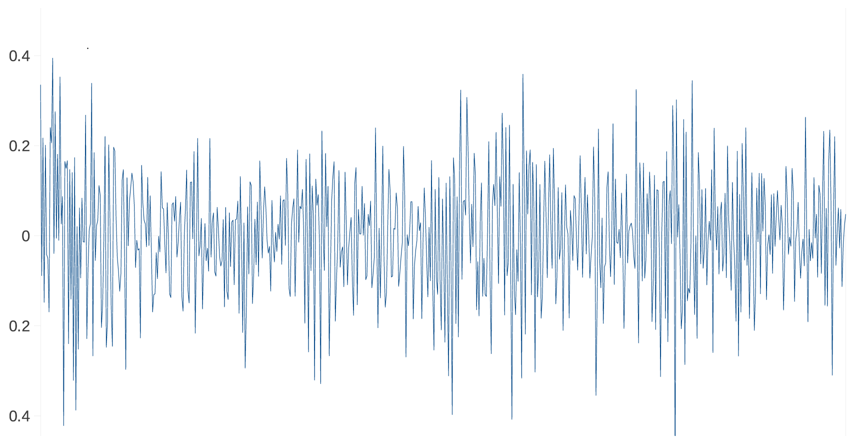
3.5 Conclusion

In this chapter, the derivation and convergence analysis of the MIMO SAD are outlined. Asymptotic stability is guaranteed provided that the step-size and mixing filter taps satisfy certain constraints, which cannot be simply extended from the TITO case, but have been re-derived. Experiments on real recordings show an increase in SNR of up to $9dB$.

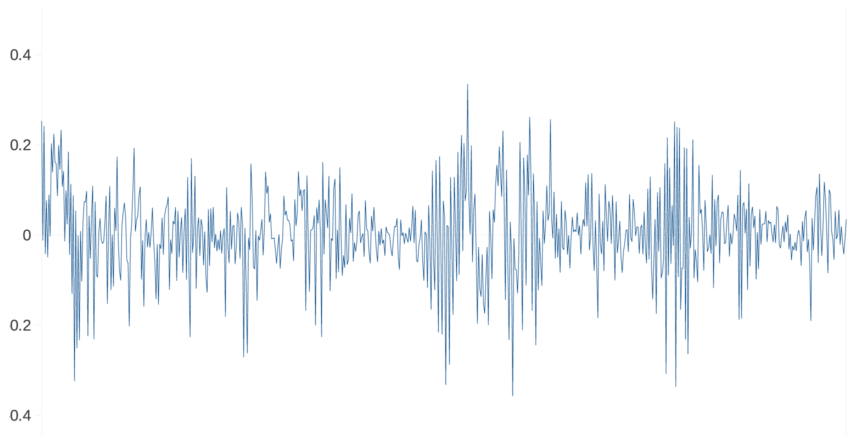
If this algorithm is to be used in a real-time application, then its computational efficiency needs to be addressed. Because the filtering and update operations are based respectively on convolution and correlation, the complexity as a function of length is $\mathcal{O}\left(L^2\right)$. And because there needs to be a filter from every mixture to all but one output, the complexity as a function of number of inputs is also $\mathcal{O}\left(N^2\right)$. This means that as filter size and number of sources increase the complexity of the algorithm increases significantly. We attempt to find



(a)



(b)



(c)

Fig. 3.10: The fig:MimoSad:DesiredSignal desired, fig:MimoSad:MixedSignal mixed, and fig:MimoSad:SeparatedSignal separated signals.

suitable solutions to this problem in following chapters.

4. MULTIBRANCHED RECURSIVE CROSSTALK RESISTANT ADAPTIVE NOISE CANCELLATION

The *multiple-input multiple-output* (MIMO) *symmetric adaptive decorrelation* (SAD) algorithm in chapter 3 was shown to be a potentially viable method of separating convolutely mixed signals. However, with increasing numbers of sources and filter sizes, the computational cost of the algorithms increased significantly. In this chapter, we endeavor to show that using a cross-talk resistant noise canceller based on *vector least mean squares* (VLMS) can improve computational efficiency without significantly compromising the degree of separation.

4.1 Background Theory

4.1.1 The Mixing System

We will first consider the case of a *two-input two-output* (TITO) system. Consider the mixing system shown in Fig. 2.2. This shows how the received signals $x_1(t)$ and $x_2(t)$ are mixtures of filtered versions of $\tilde{s}_1(t)$ and $\tilde{s}_2(t)$. In matrix-polynomial form this is

$$\mathbf{x}(t) = \mathbf{H}^T \tilde{\mathbf{s}}(t) \tag{4.1}$$

where $\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) \end{bmatrix}^T$, the superscript "$T$" denotes the transpose operator, t denotes the time index,

$$\mathbf{H} = \begin{bmatrix} \begin{bmatrix} h_{1,1}^0 & h_{1,2}^0 \\ h_{2,1}^0 & h_{2,2}^0 \end{bmatrix} & H^1 & \dots & H^{L-1} \end{bmatrix}^T$$

is the mixing matrix with L taps (the superscript indicates the tap number), and

$$\tilde{\mathbf{s}}(t) = \Big[\tilde{s}_1(t) \quad \tilde{s}_2(t) \quad \tilde{s}_1(t-1) \quad \tilde{s}_2(t-1) \quad \dots \\ \tilde{s}_1(t-L+1) \quad \tilde{s}_2(t-L+1) \Big]^T$$

This can be considered the convolutive analogue of (2.7). While similar to (2.9), it is different as it interleaves the source signals rather than concatenating them.

Fig. 2.3 shows the simplified mixing system. In relation to the actual mixing system shown in Fig. 2.2, we have

$$\begin{aligned} s_1 &= h_{1,1}\left(z^{-1}\right)\tilde{s}_1 \\ s_2 &= h_{2,2}\left(z^{-1}\right)\tilde{s}_2 \\ g_{1,2}\left(z^{-1}\right) &= \frac{h_{1,2}\left(z^{-1}\right)}{h_{2,2}\left(z^{-1}\right)} \\ g_{2,1}\left(z^{-1}\right) &= \frac{h_{2,1}\left(z^{-1}\right)}{h_{1,1}\left(z^{-1}\right)} \end{aligned} \tag{4.2}$$

The equations in (4.2) show how the separation will not result in the original sources $\tilde{s}_1$ and $\tilde{s}_2$. Rather, at best it will separate the sources to s_1 and s_2, which are filtered versions of the originals.

The SAD algorithm can be used to solve this problem as shown in chapter 3, both for the TITO case and the MIMO case. We now propose adaptation to the SAD algorithm which increases its efficiency as the number of sources increases.

4.2 The Cross-Coupled Vector-LMS

In order to show the working of the *crosstalk resistant adaptive noise canceller* (CTRANC) based on VLMS, we will consider the situation of four inputs and four outputs. In Fig. 4.1 we have a matrix polynomial representation of the mixing system, where $\tilde{\mathbf{s}}_1(t) = \begin{bmatrix} \tilde{s}_1(t), & \tilde{s}_2(t) \end{bmatrix}^T$ and $\tilde{\mathbf{s}}_2(t) = \begin{bmatrix} \tilde{s}_3(t), & \tilde{s}_4(t) \end{bmatrix}^T$ are the four inputs multiplexed into two vectors, $\mathbf{x}_1(t) = \begin{bmatrix} x_1(t), & x_2(t) \end{bmatrix}^T$ and $\mathbf{x}_2(t) = \begin{bmatrix} x_3(t), & x_4(t) \end{bmatrix}^T$ are the four outputs multiplexed into two vectors, and $\mathbf{H}_{1,1}$, $\mathbf{H}_{1,2}$, $\mathbf{H}_{2,1}$, and $\mathbf{H}_{2,2}$ are all mixing polynomial matrices representing the entire mixing system. Note that these should not be confused with their scalar counterparts.

Using the same reasoning as with the ordinary CTRANC, we derive the following update equations for the separating polynomial matrices $\mathbf{W}_{1,2}$ and $\mathbf{W}_{2,1}$.

$$\mathbf{W}_{1,2}(t+1) = \mathbf{W}_{1,2}(t) + \mu \mathbf{Y}_2(t)\, \mathbf{y}_1^T(t)$$
$$\mathbf{W}_{2,1}(t+1) = \mathbf{W}_{2,1}(t) + \mu \mathbf{Y}_1(t)\, \mathbf{y}_2^T(t)$$

where μ_1 and μ_2 are convergence weights, $\mathbf{y}_1(t)$ and $\mathbf{y}_2(t)$ are the length-2 output vectors $\begin{bmatrix} y_1(t), & y_2(t) \end{bmatrix}$ and $\begin{bmatrix} y_3(t), & y_4(t) \end{bmatrix}$ respectively, and the length-$2L$ vectors $\mathbf{Y}_1(t)$ and

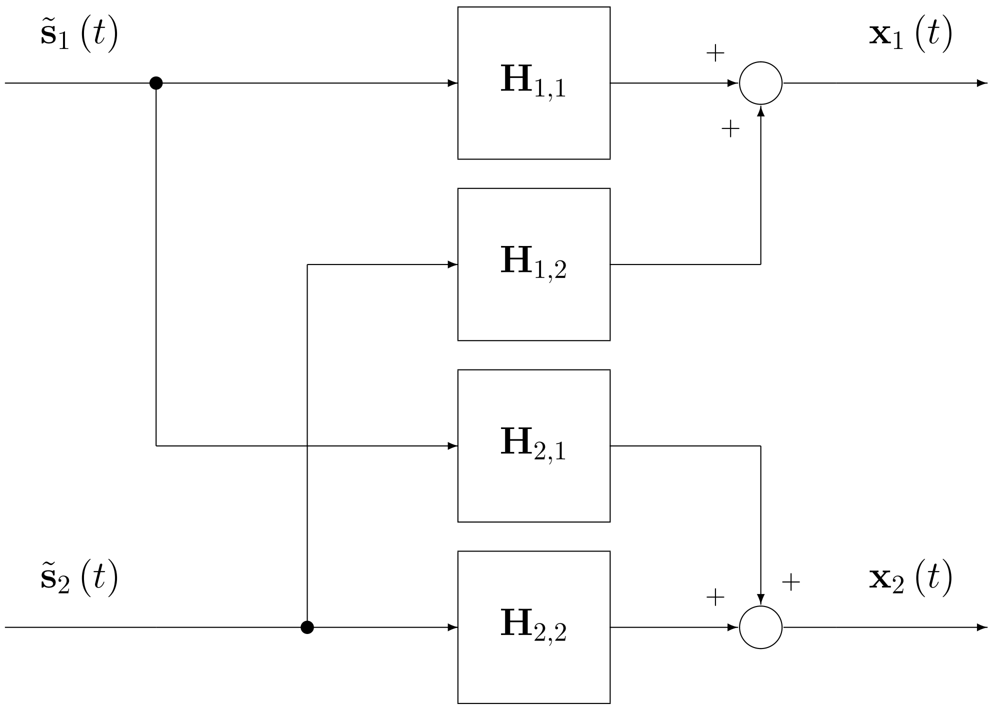


Fig. 4.1: The four input mixing system.

$\mathbf{Y}_2(t)$ are defined by

$$\mathbf{Y}_1(t) = \left[\begin{array}{cccc} \mathbf{y}_1^T(t-1) & \mathbf{y}_1^T(t-2) & \dots & \mathbf{y}_1^T(t-L+1) \end{array} \right]$$
$$\mathbf{Y}_2(t) = \left[\begin{array}{cccc} \mathbf{y}_2^T(t-1) & \mathbf{y}_2^T(t-2) & \dots & \mathbf{y}_2^T(t-L+1) \end{array} \right]$$

The essential operation of this algorithm is to separate a system of four mixed sources into two systems of two mixed sources. One can then apply the algorithm from an ordinary CTRANC to separate each of the sources into approximations of the original individual signals.

The pseudocode for this algorithm is shown in Program 2.

Program 2 Multibranched SAD Pseudocode

```
Superscript $^T$ denotes the transpose operation.
Any numeric superscripts denote vector indices.
$n$                                   ▷ the number of *observed signal* samples
$N_0$                                 ▷ the number of observed signals
$L$                                   ▷ the filter size
$x_1, x_2, \ldots, x_{N_0}$           ▷ the observed signals
$y_1, y_2, \ldots, y_{N_0}$           ▷ the *outputs*
$\mathbf{W}_{i,j,k}(t)$               ▷ the $k \times kL$ matrix consisting of all demixing matrix taps
                                        from one quadrant of the demixing matrix
$\mathbf{x}_0(t) = \begin{bmatrix} x_1(t) & x_2(t) & \ldots & x_{N_0}(t) \end{bmatrix}^T$   ▷ an array of all observed signal samples at $t$
$\mathbf{X}_0(t) = \begin{bmatrix} \mathbf{x}_0^T(t) & \mathbf{x}_0^T(t-1) & \ldots & \mathbf{x}_0^T(t-L+1) \end{bmatrix}^T$

function MimoSad($\mathbf{X}(t)$,$N$,t)
    for $i = 0 \to L-1$ do
        for $j = 1 \to \frac{N}{2}$ do
            $\mathbf{X}_1^{\frac{N}{2}i+j} = \mathbf{X}^{Ni+j}$
            $\mathbf{X}_2^{\frac{N}{2}i+j} = \mathbf{X}^{N\left(i+\frac{1}{2}\right)+j}$
        end for
    end for
    $\mathbf{y}_1(t) = \mathbf{x}_1(t) - \mathbf{W}_{1,2,\frac{N}{2}}^T \mathbf{X}_2(t)$
    $\mathbf{y}_2(t) = \mathbf{x}_2(t) - \mathbf{W}_{2,1,\frac{N}{2}}^T \mathbf{X}_1(t)$
    $\mathbf{Y}_1(t) = \begin{bmatrix} \mathbf{y}_1^T(t) & \mathbf{y}_1^T(t-1) & \ldots & \mathbf{y}_1^T(t-L+1) \end{bmatrix}^T$
    $\mathbf{Y}_2(t) = \begin{bmatrix} \mathbf{y}_2^T(t) & \mathbf{y}_2^T(t-1) & \ldots & \mathbf{y}_2^T(t-L+1) \end{bmatrix}^T$

    $\mathbf{W}_{1,2,\frac{N}{2}} = \mathbf{W}_{1,2,\frac{N}{2}} + \mu \mathbf{Y}_2(t) \mathbf{y}_1^T(t)$
    $\mathbf{W}_{2,1,\frac{N}{2}} = \mathbf{W}_{2,1,\frac{N}{2}} + \mu \mathbf{Y}_1(t) \mathbf{y}_2^T(t)$

    if $N > 1$ then
        $\mathbf{Y}_1(t)$ = MimoSad($\mathbf{Y}_1(t)$,$\frac{N}{2}$,$t$)
        $\mathbf{Y}_2(t)$ = MimoSad($\mathbf{Y}_2(t)$,$\frac{N}{2}$,$t$)
    end if
    return $\begin{bmatrix} \mathbf{Y}_1(t) \\ \mathbf{Y}_2(t) \end{bmatrix}$
end function

MimoSad($\mathbf{X}_0(t)$,$N$,$t$)
```

4.2.1 Computational Efficiency

While [44] may have already proposed a CTRANC algorithm for more than two sources, the advantage in using a multibranched recursive approach is that it is potentially more efficient for more sources.

We will take the case when there are $L = 4$ sources. We will therefore need one 2×2 matrix CTRANC (which translates to two multivariable *least mean squares* (LMS) algorithms) to separate the mixture into two mixtures of two signals, then we will need two scalar CTRANCs (which translates to four scalar LMS algorithms) to separate the systems into the individual signals.

For each multivariable LMS, there are

- $2LM^2 + M$ multiplications
- $LM^2 + M$ additions/subtractions

where L is the number of taps in the filter and M is the number of inputs of the multivariable LMS. However, it is important to note that with an N-input matrix CTRANC, each multivariable LMS only has $M = N/2$ inputs.

For each scalar LMS, there are

- $2L + 3$ multiplications
- $L + 2$ additions/subtractions

Therefore, for $N = 4$ there will be a total of

- $24L + 32$ multiplications
- $12L + 20$ additions/subtractions

for the multibranched CTRANC.

For the method proposed in [44], it is necessary to have $N\,(N-1)$ scalar LMS algorithms. When $N = 4$, this means that there will be 12 scalar LMS algorithms, which requires

- $24L + 36$ multiplications
- $12L + 24$ additions/subtractions

For the full separation of four sources, the method proposed in this chapter only gives a computational advantage of four multiplications and four additions/subtractions.

However, as the number of inputs increases, so does the savings in number of computations. For the multivariable CTRANC with 8 inputs we have

- $112L + 136$ multiplications
- $56L + 80$ additions/subtractions

as compared to the

- $112L + 168$ multiplications
- $56L + 112$ additions/subtractions

for the ordinary CTRANC, giving a computational advantage of 32 multiplications and 32 additions/subtractions.

Where the real computational efficiency is gained, however, is if there is only one desired signal amongst the noise, and we know which channel it is on the output of the CTRANC. If this is the case, then we only need one of each of the matrix-polynomial CTRANC steps, and only one TITO CTRANC. For $L = 4$, we save $4L + 10$ multiplications and $2L + 8$ additions/subtractions. When $L = 8$, we save $28L + 70$ multiplications and $14L + 56$ additions/subtractions.

While present, these gains are not overly significant when attempting to separate all of the sources. For this reason, in subsequent chapters we will move away from the multi-branched approach and toward a frequency-domain approach which offers better computational savings.

4.3 Separation Performance

We conducted a simple experiment to discover the relative separation of the proposed method to the method in [44].

4.3.1 Experimental Procedure

The experiment was set up as follows: four microphones were placed as four corners of a 0.2m$\times$0.2m square near the middle of a 4m$\times$7m room furnished with a lounge suite, a piano and a dining room suite. There were three noise sources, all samples of a car assembly line from the file labelled 'factory floor noise 2' from the NOISEX database. The speech was created by using a loudspeaker in front of the computer monitor, playing the speech sample in the package 'Lunatick-20080326–cc.tgz' from the VoxForge speech corpus.

Using the described set-up, we used the proposed algorithm to reduce the noise level. Each filter had 1000 weights. We chose this number because increasing the number of weights beyond 1000 increased computational complexity with a negligible increase in *signal to noise ratio* (SNR), while decreasing the number of weights adversely affected the results.

Tab. 4.1: Increases in SNR

	Input SNR	Output SNR	Increase in SNR
Proposed Method	7.7 dB	14.2 dB	6.5 dB
Mei *et al.* Method	7.7 dB	13.9 dB	6.2 dB

Because we do not have the power of the noise by itself, to calculate the SNR, we need to use the following formula:

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{P_{SN} - P_N}{P_N} \right) \tag{4.3}$$

where P_{SN} is the combined power of the speech with the noise and P_N is the power of the noise. This is based on the assumption that the noise and the speech are statistically independent.

4.3.2 Results

Using (4.3), we obtained the results as shown in Table 4.1. In an informal listening test, we also found that the speech was more comprehensible in the separated signals than in the mixed signals.

The difference in SNR between the different approaches is negligible. This shows that, while the proposed method is faster to execute, there is no drop in the degree of separation.

4.4 Conclusion

One solution to the problem of blind source separation is to use a cross-talk resistant noise canceller to separate the signals. This chapter describes an adaptation to the CTRANC algorithm to increase its computational efficiency. Experimental data shows that there is no drop in performance in spite of the decreased computational cost. It also has the advantage that it is potentially even more computationally efficient if there is only one desired source, and it is known which channel it will be separated to. However, while present, the computational savings are not overly substantial. Therefore, in future chapters other methods will be investigated to increase computational efficiency in the aim of getting more significant gains.

5. FREQUENCY DOMAIN SAD

An adaptation to the *symmetric adaptive decorrelation* (SAD) algorithm that significantly increases computational efficiency is proposed in this chapter. Because it is based on the *least mean squares* (LMS) algorithm, if the bulk of the processing of the SAD algorithm can be done in the frequency domain, the computational benefits are similar to those seen with the LMS algorithm. While there are computational savings with an increase in filter order as expected, it is shown that there are also computational savings relative to the time domain implementation when the number of sources increases.

5.1 Background Information

The computational efficiency of the LMS algorithm can be significantly increased by transferral of the bulk of the work into the frequency domain. This is mainly due to two operations, convolution when passing the noise signal $n(t)$ through filter $\mathbf{w}(t)$, and correlation in the gradient term of the update equation. If the fast Fourier transform (FFT) is used to efficiently convert the signals and filters into the frequency domain, the convolution and correlation time-domain operations become element-by-element multiplication.

The fast least mean squares (FLMS) algorithm was first proposed by Dentino *et al.* in [21], but failed to zero-pad the filters in order to get the *linear* convolution/correlation. Rather, the algorithm incorporates *circular* convolution, which results in an algorithm that does not converge to the correct solution. Later, Ferrara in [22] proposed an overlap-save frequency domain algorithm that performed the convolution and correlation in the frequency domain, but utilized zero padding in the time domain in order to ensure that operations were linear rather than circular. The resulting algorithm is less computationally efficient than [21] due to the longer Fourier transforms, plus additional conversions to and from the time domain in order to zero-pad the signals. However, it is still substantially more efficient that the time-domain algorithm (for larger filter sizes), and correctly converges to the same solution.

In order to show the FLMS algorithm, the following notation will be used. For variables that are implemented in both the time domain and frequency domain, time domain versions are written in lower case (x) and frequency domain variables are denoted by an upper case character (X). Scalars are in italics, and vectors are in bold; vectors which are in the time domain are all of size L, those in the frequency domain are all of size $2L$, where $L-1$ is the order of the demixing filters. The symbol t is used to indicate the iteration number in the time domain, and k is used to indicate the block number when operating in the frequency domain. The superscript ^{T} denotes the transpose operation and the overline ($\overline{X}$) denotes complex conjugation. F is used to denote the discrete Fourier transform (DFT) matrix, and F^{-1} its inverse. It should be noted that although the DFT matrix is used in the derivations, this is purely for notational convenience. Actual implementation should use the FFT for its computational efficiency. The derivation here closely follows that shown in [23].

For a frequency domain implementation, the LMS algorithm must first be converted to a block algorithm. This is an important distinction between the two algorithms, as it means that the filter $\mathbf{w}\left(t\right)$ cannot significantly change over L samples, and can be assumed constant over this time period. Therefore the LMS update equation in (3.4) becomes

$$\begin{aligned}\mathbf{w}\left(kL+L\right)=&\mathbf{w}\left(kL\right)-\mu\left[e\left(kL\right)\mathbf{x}\left(kL\right)+e\left(kL+1\right)\mathbf{x}\left(kL+1\right)\right.\\&\left.+\ldots+e\left(kL+L-1\right)\mathbf{x}\left(kL+L-1\right)\right]\\=&\mathbf{w}\left(kL\right)-\mu\boldsymbol{\nabla}\left(kL\right)\end{aligned}\tag{5.1}$$

which gives us the update term

$$\boldsymbol{\nabla}\left(kL\right)=\begin{bmatrix}\sum_{i=0}^{L-1}e\left(kL+i\right)x\left(kL+i\right)\\\sum_{i=0}^{L-1}e\left(kL+i\right)x\left(kL+i-1\right)\\\vdots\\\sum_{i=0}^{L-1}e\left(kL+i\right)x\left(kL+i-L+1\right)\end{bmatrix}\tag{5.2}$$

which is the *cross-correlation* vector between $e\left(t\right)$ and $x\left(t\right)$. In the frequency domain, this is calculated using

$$\begin{bmatrix}\boldsymbol{\nabla}\left(kL\right)\\\mathbf{0}\end{bmatrix}=GF^{-1}\left\{\overline{F\left[G\mathbf{e}_{2L}\left(kL\right)\right]}\circ F\left[\mathbf{x}_{2L}\left(kL\right)\right]\right\}\tag{5.3}$$

where

$$G=\begin{bmatrix}I_{L\times L}&0_{L\times L}\\0_{L\times L}&0_{L\times L}\end{bmatrix}$$

is a zero-padding matrix, the overline indicates complex conjugation, which maps to time reversal in the time domain, the operator '$\circ$' denotes element-by-element multiplication, and the subscript $2L$ means that the regressor vectors are of size $2L$ rather than L.

When converting to the frequency domain for fast convolution, zero-padding is an important step. Any signal that is bounded in the time-domain (such as a block used in block-processing) needs to be represented by an unbounded frequency-domain series. This is impractical, so instead the assumption is made that the time-domain block is not a block, but an infinite series. The question then arises as to what this infinite time-domain signal looks like outside of that defined by the block. The usual assumption (as is taken in the FFT) is that the block simply repeats itself infinitely, thus allowing a bounded frequency domain representation of the signal.

However, simply using elementwise multiplication in the frequency domain maps to a convolution of these infinite series, which *is not* the same as a linear convolution in the time domain, but what is known as *circular* convolution. This erroneous result can be avoided by zero-padding in the time domain before performing the frequency domain transform. While this is still technically a circular convolution, because portions of the convolved signals are now zero, the result is in fact the same as a linear convolution of the original sequences. Thus the spurious results that would normally occur due to circular convolution are avoided.

For the calculation of the error to be performed in the frequency domain, it too must be implemented as a block algorithm. Equation (3.2) then becomes

$$\mathbf{e}\left(t\right)=\mathbf{x}\left(t\right)-\begin{bmatrix}\mathbf{w}^{T}\left(t-1\right)\mathbf{n}\left(t\right)\\ \mathbf{w}^{T}\left(t-2\right)\mathbf{n}\left(t-1\right)\\ \vdots\\ \mathbf{w}^{T}\left(t-L\right)\mathbf{n}\left(t-L+1\right)\end{bmatrix} \tag{5.4}$$

However, because $\mathbf{w}\left(t\right)$ is only updated once every L time samples, $\mathbf{w}\left(t\right)$ remains unchanged over L updates. Therefore (5.4) becomes

$$\mathbf{e}(kL) = \mathbf{x}(kL) - \begin{bmatrix} \mathbf{w}^T((k-1)L)\,\mathbf{n}(kL) \\ \mathbf{w}^T((k-1)L)\,\mathbf{n}(kL-1) \\ \vdots \\ \mathbf{w}^T((k-1)L)\,\mathbf{n}(kL-L+1) \end{bmatrix} \tag{5.5}$$

This can be calculated in the frequency domain as

$$\begin{aligned} \begin{bmatrix} \mathbf{e}(kL) \\ \mathbf{0} \end{bmatrix} &= G\mathbf{x}_{2L}(kL) \\ &\quad - GF^{-1}\left\{ F\begin{bmatrix} \mathbf{w}(kL-L) \\ \mathbf{0} \end{bmatrix} \circ F\left[\mathbf{n}_{2L}(kL)\right] \right\}. \end{aligned} \tag{5.6}$$

The block equations (5.1), (5.3) and (5.6) can then be used in lieu of the time domain equations (3.2) and (3.4) as the updates for the filters. The main disadvantage of the frequency domain algorithm over the time-domain algorithm is that it does not converge quite as quickly, though further adaptations can be made to reduce this decrease in performance [23].

5.2 Frequency Domain SAD

Similar to how a frequency domain implementation can be found for the LMS algorithm, the *frequency-domain symmetric adaptive decorrelation* (FD-SAD) algorithm can also be derived for the SAD algorithm.

In order to implement the algorithm in the frequency domain, it must first be converted to a block algorithm that is only updated once every L iterations, where L is the filter order.

(3.28) therefore becomes

$$\mathbf{w}_{i,j}\left(kL+L\right)=\mathbf{w}_{i,j}\left(kL\right)-\frac{\mu}{\sigma^2}\nabla\mathbf{w}_{i,j}\left(kL\right) \tag{5.7}$$

where

$$\nabla\mathbf{w}_{i,j}\left(t\right)=\sum_{l=0}^{L-1}y_i\left(t+l\right)\mathbf{y}_j\left(t+l\right) \tag{5.8}$$

The update equation in (5.7) can be rewritten in the frequency domain as

$$\mathbf{W}_{i,j}\left(k+1\right)=\mathbf{W}_{i,j}\left(k\right)-\frac{\mu}{\sigma^2}\nabla\mathbf{W}_{i,j}\left(k\right) \tag{5.9}$$

where

$$\mathbf{W}_{i,j}\left(k\right)=F\begin{bmatrix}\mathbf{w}_{i,j}\left(kL\right)\\ \mathbf{0}_L\end{bmatrix}$$
$$\nabla\mathbf{W}_{i,j}\left(k\right)=F\begin{bmatrix}\nabla\mathbf{w}_{i,j}\left(kL\right)\\ \mathbf{0}_L\end{bmatrix}$$

where F is the DFT matrix.

Because the update term in (5.8) is acquired by finding the correlation between $y_i\left(t\right)$ and $y_j\left(t\right)$, the frequency domain representation is found by element-wise multiplication in the frequency domain then zero-padding it in the time domain

$$\nabla\mathbf{W}_{i,j}\left(k\right)=F\left[G_{2L}F^{-1}\left\{\overline{G_{2L}\mathbf{Y}_i\left(k\right)}\circ\mathbf{Y}_j\left(k\right)\right\}\right] \tag{5.10}$$

where $\circ$ represents element-wise multiplication, the overline denotes complex conjugation which maps to time-domain reversal, the frequency-domain *output* vector

$$\mathbf{Y}_i\left(k\right)=F\begin{bmatrix}\mathbf{y}_i\left(kL\right)\\ \mathbf{y}_i\left(kL-L\right)\end{bmatrix}, \tag{5.11}$$

and the zero padding matrix

$$G_{2L}=\begin{bmatrix}I_{L\times L} & 0_{L\times L}\\ 0_{L\times L} & 0_{L\times L}\end{bmatrix}$$

is used to prevent circular correlation.

Therefore, using (5.9) and (5.10) the frequency domain update equation is found.

A similar approach can also be taken to find the filter outputs $\mathbf{y}_i\left(kL\right)$, and with the definition in (5.11) find $\mathbf{Y}_i\left(k\right)$.

$$\mathbf{y}_i\left(kL\right)=\begin{bmatrix}\sum_{l=0}^{N-1}\mathbf{x}_l^T\left(kL\right)\mathbf{w}_{i,l}\left(kL\right)\\ \sum_{l=0}^{N-1}\mathbf{x}_l^T\left(kL-1\right)\mathbf{w}_{i,l}\left(kL-1\right)\\ \vdots\\ \sum_{l=0}^{N-1}\mathbf{x}_l^T\left(kL-L+1\right)\mathbf{w}_{i,l}\left(kL-L+1\right)\end{bmatrix} \tag{5.12}$$

where

$$\mathbf{x}_l\left(kL\right)=\begin{bmatrix}x_l\left(kL\right) & x_l\left(kL-1\right) & \dots & x_l\left(kL-L+1\right)\end{bmatrix}^T.$$

Under the assumption that

$$\begin{aligned}\mathbf{w}_{i,j}(t) &= \mathbf{w}_{i,j}(t-1) \\ &= \mathbf{w}_{ij}(t-2) \\ &\vdots \\ &= \mathbf{w}_{i,j}(t-L+1) \end{aligned} \tag{5.13}$$

we can simplify (5.12) to

$$\mathbf{y}_i(kL) = \sum_{l=0}^{N-1} \begin{bmatrix} \mathbf{x}_l^T(kL) \\ \mathbf{x}_l^T(kL-1) \\ \vdots \\ \mathbf{x}_l^T(kL-L+1) \end{bmatrix} \mathbf{w}_{i,l}(kL) \tag{5.14}$$

This can be obtained in the frequency domain using the following sum

$$\begin{bmatrix} \mathbf{y}_i(kL) \\ \mathbf{0}_L \end{bmatrix} = G_{2L} F^{-1} \left[\sum_{l=0}^{N-1} \overline{\mathbf{W}_{i,l}(k)} \circ \mathbf{X}_l(k) \right] \tag{5.15}$$

where

$$\mathbf{X}_l(k) = F \begin{bmatrix} \mathbf{x}_l(kL) \\ \mathbf{x}_l(kL-L) \end{bmatrix}$$

5.3 Comparison to the Time Domain Implementation

There is one fundamental difference between the time-domain and the frequency-domain implementations of the SAD algorithm. This is the assumption that while the separating

Program 3 The FD-SAD Pseudocode

```
$\circ$ denotes entry-wise multiplication.
The overline denotes complex conjugation.
$n$                          ▷ the number of *observed signal* samples
$N$                          ▷ the number of observed signals
$L$                          ▷ the filter size
$x_1, x_2, \ldots, x_N$      ▷ the observed signals
$\mathbf{W}_{i,j}$           ▷ the length $2L$ vectors that contain the filter weights initialized to zero

for $k = 0 \to \frac{n}{L}$ do
    for $i = 1 \to N$ do
        $\mathbf{x}_i(k)$ = the next $L$ samples of $x_i$ in reverse order.
        $\mathbf{X}_i(k) = \text{FFT}\begin{bmatrix} \mathbf{x}_i(k) \\ \mathbf{x}_i(k-1) \end{bmatrix}$
    end for
    for $i = 1 \to N$ do
        $\mathbf{tmp}_{2L} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}^T$.
        for $j = 1 \to N, j \neq i$ do
            $\mathbf{tmp}_{2L} += \overline{\mathbf{W}_{i,j}(k)} \circ \mathbf{X}_j(k)$
        end for
        $\begin{bmatrix} \mathbf{y}_i(k) \\ \mathbf{v} \end{bmatrix} = \text{FFT}^{-1}[\mathbf{tmp}_{2L}]$
    end for

    for $j = 1 \to N$ do
        $\mathbf{Y}_j(k) = \text{FFT}\begin{bmatrix} \mathbf{y}_j(k) \\ \mathbf{y}_j(k-1) \end{bmatrix}$
    end for
    for $i = 1 \to N$ do
        $\mathbf{Y}'_i(k) = \text{FFT}\begin{bmatrix} \mathbf{y}_i(k) \\ \mathbf{0} \end{bmatrix}$

        for $j = 1 \to N, i \neq j$ do
            $\nabla\mathbf{W}_{i,j}(k) = \overline{\mathbf{Y}'_i(k)} \circ \mathbf{Y}_j(k)$
            $\mathbf{W}_{i,j}(k+1) = \mathbf{W}_{i,j}(k) + \mu\nabla\mathbf{W}_{i,j}(k)$
            $\begin{bmatrix} \mathbf{w}_{i,j}(k) \\ \mathbf{v} \end{bmatrix} = \text{FFT}^{-1}[\mathbf{W}_{i,j}(k)]$
            $\mathbf{W}_{i,j}(k) = \text{FFT}\begin{bmatrix} \mathbf{w}_{i,j}(k) \\ \mathbf{0} \end{bmatrix}$
        end for
    end for
end for
```

filters w_1 and w_2 do change over L samples in the time-domain implementation, this change is insignificant enough that they can be assumed constant, as described by Equation (5.13).

In [101], Widrow *et al.* show the calculation of the time constant for convergence of an LMS filter. Because of its similarity to LMS, the SAD algorithm has comparable calculations for the time constant.

The LMS filter (and therefore also the SAD algorithm) is based on the assumption that the mean square error can be approximated by the instantaneous squared error. This results in *nonstationarity* of the performance surface even for a stationary system. Although the LMS algorithm converges in mean to the optimal solution, if μ is chosen too high, then a significant amount of noise is introduced into the system due to convergence to the shifting minimum. This was termed the *misadjustment* of the LMS filter by Widrow *et al.* in [101]. It is therefore important to choose μ such that this gradient noise is significantly reduced.

In [101], Widrow *et al.* define the misadjustment as

$$M \triangleq \frac{\text{average excess mean square error}}{\text{minimum mean square error}} \tag{5.16}$$

and suggest that a value of ten percent is a suitable value for most engineering applications. In order to obtain such a value, the number of samples required for the transients to settle needs to be roughly ten times as long as the filter size, meaning that, although w_1 will change over L time samples in the *time-domain symmetric adaptive decorrelation* (TD-SAD) algorithm, it will not be enough to significantly compromise the performance of the frequency domain implementation. However, it is quite possible to choose μ such that the system is stable, but the time taken for the outputs to converge is comparable to the time memory of the filters.

If μ is chosen such that it is on the upper end of this limit, stability is ensured but the assumption that the filters w_1 and w_2 do not change significantly over time *cannot* be justified. However, in these cases, the only detrimental effect on the frequency domain algorithm will be a slight increase in convergence time over the time domain implementation, not in the degree of separation.

5.4 Computational Cost

A major advantage in processing the signals in the frequency domain is that there can be significant computational savings both in the filtering of the signals and in the updates. We will consider both the time domain and the frequency domain implementations and compare the number of additions and multiplications required for each algorithm.

For the time-domain case, there are $(N-1)$ separation filters for every *source* to separate, and therefore as many update equations. For each update equation, (assuming μ/σ^2 is a constant), there are

$$L + 1 \quad \text{multiplications}$$

and each demixing filter requires

$$L \quad \text{multiplications}$$

to produce the separated outputs.

Therefore, L iterations of the time domain demixing system requires

$$L\,(N-1)\,(2L+1) \quad \text{multiplications}$$

for every input. More detail on this derivation can be found in Appendix C.1.

For the frequency domain algorithm, it is important to keep in mind that each filter needs to be twice the length of the time domain filters to combat problems arising from circular convolution. Examining the code in Program 3, it can be seen that there are $2N+2$ FFTs per input. Assuming real signals [102] each requires

$$2L\log_2 L - 5L + 6 \quad \text{real multiplications}$$

Aside from these multiplications, there are also $4L\,(N-1)$ complex multiplications and L real multiplications per output, which equates to a total of

$$2\,(2L\log_2 L + 3L + 6)\,(N+1) - 31L \qquad \text{real multiplications}$$

per output. See Appendix C.2 for more details on how this was attained.

Although the frequency domain method has more multiplications per update of the demixing filter coefficients, it should be noted the update only needs to occur once per L time samples.

Table 5.1 shows a comparison of multiplications required for two sources for varying filter lengths, and table 5.2 shows the ratio of multiplications required in the frequency domain

Tab. 5.1: Multiplications Required for $N = 2$

Filter order	TD-SAD	FD-SAD
8	0.32×10^3	0.44×10^3
16	1.22×10^3	1.19×10^3
32	4.61×10^3	3.08×10^3
64	1.77×10^4	0.76×10^4
128	6.86×10^5	0.18×10^5
2048	1.69×10^7	0.05×10^7

Tab. 5.2: FD-TD Ratio of Multiplications Required

Filter order	2 sources	4 sources	8 sources	16 sources
8	1.38	1.19	1.13	1.10
16	0.98	0.77	0.70	0.67
32	0.67	0.49	0.43	0.41
64	0.43	0.30	0.26	0.24
128	0.27	0.18	0.15	0.14
2048	0.029	0.018	0.015	0.013

to those required in the time domain.

From these tables, it can be seen that not only does the relative computational efficiency of the FD-SAD algorithm increase as the separating filter order increases, it also increases as the number of sources increases. The primary reason for this is that during the filtering stage of the algorithm, the time domain implementation requires $N\,(N-1)$ convolution operations, but the frequency domain implementation only requires N FFTs. This is clear from the pseudocode in Program 3.

5.5 *Simulations*

Simulations were undertaken in order to compare the difference in performances of the time-domain and the frequency-domain SAD algorithms. Although it is shown in section 5.4 that the separating filters need to be at least 32 taps long for any improvement in computational

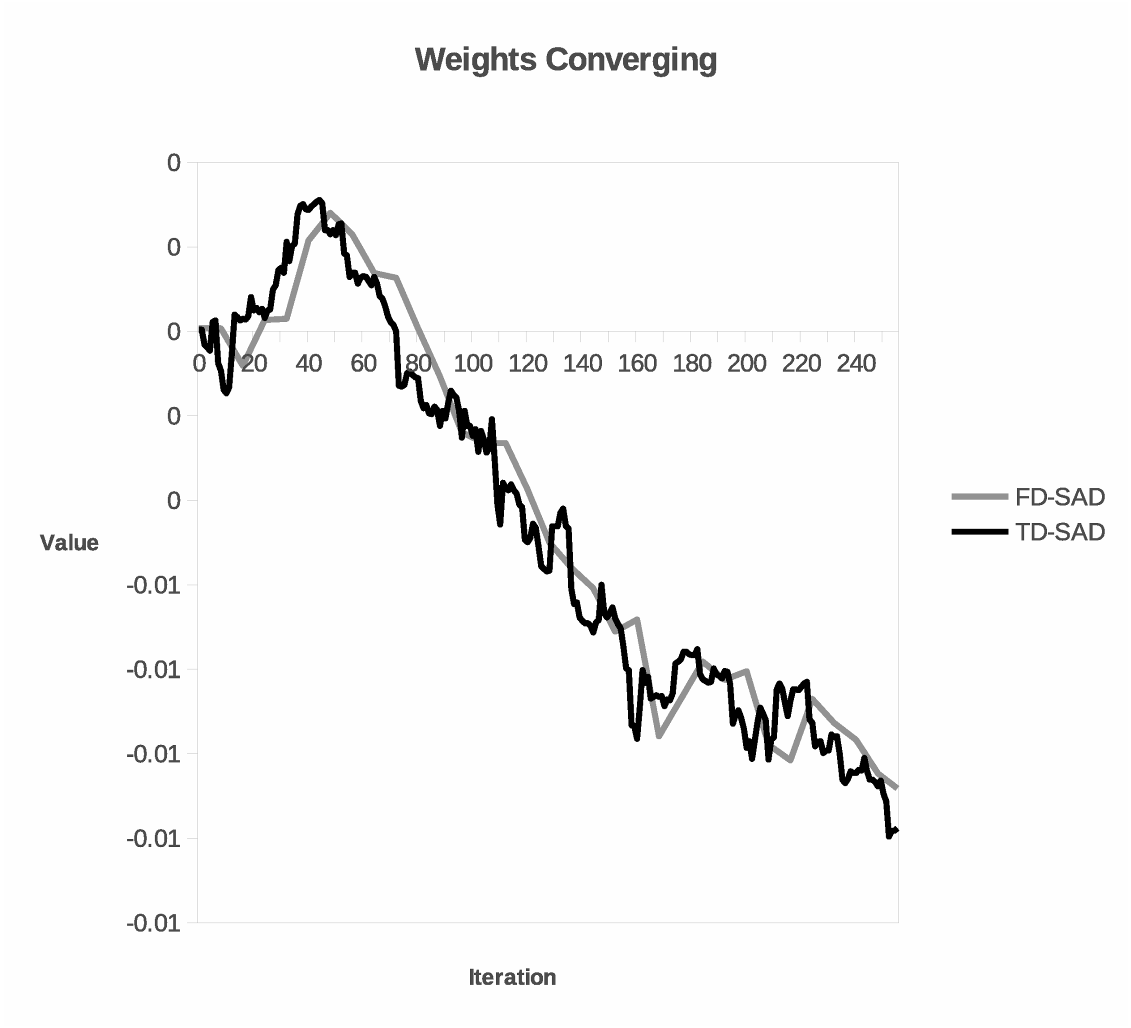


Fig. 5.1: The filter weights after 256 time samples with $\mu = 0.02$.

complexity for 2 sources, for clarity's sake we will only show here filters of length 8. Other simulations have shown that the algorithm works equally well for separating filters beyond 32 taps. The data and adaptive step size μ were identical for both the time domain and the frequency domain algorithms.

Figure 5.1 shows the convergence of the tap at index 1 of w_1 over 256 time samples for both algorithms, and figure 5.2 shows the entire filter for both algorithms after 256 time samples. From these figures one can soon see that although the convergence of the frequency

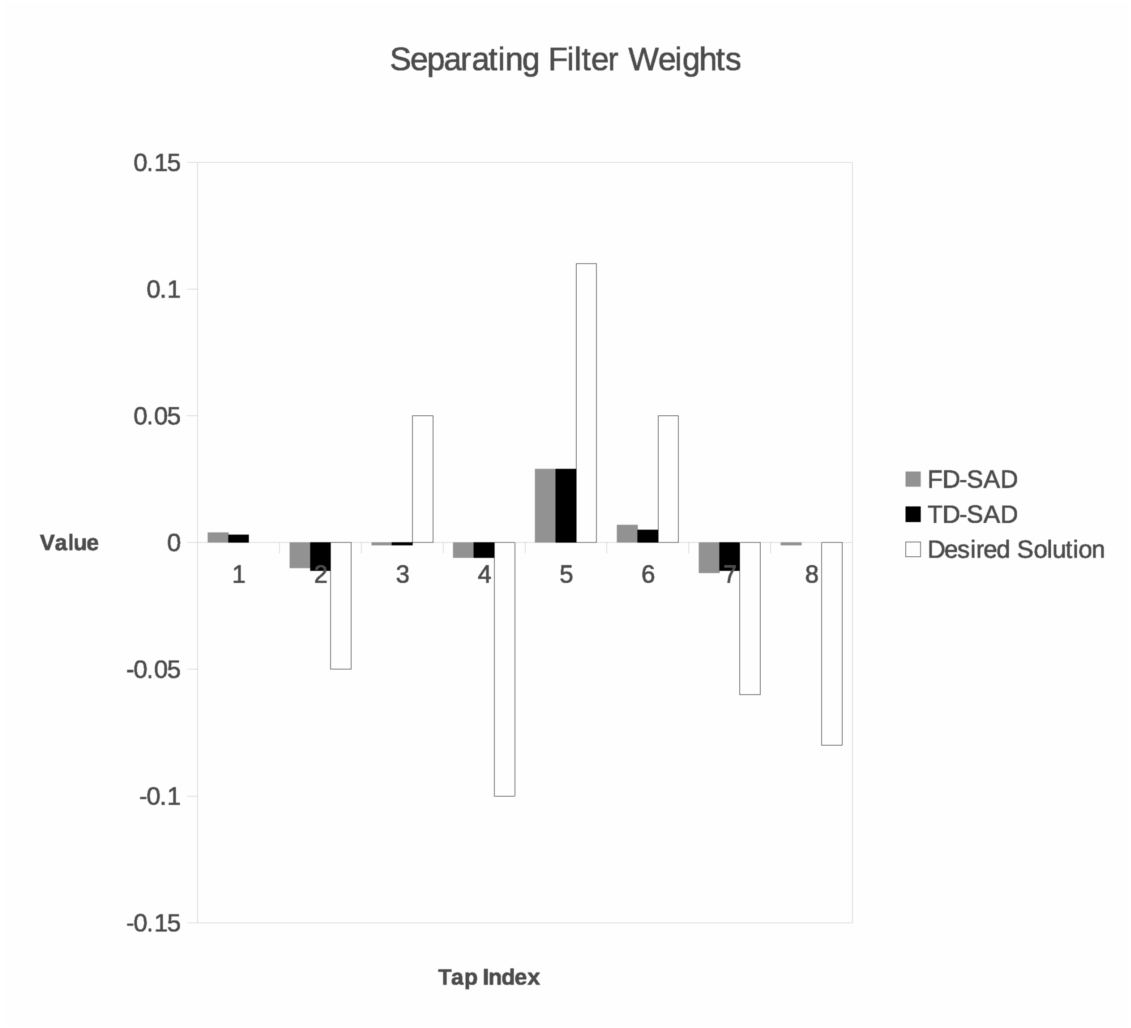


Fig. 5.2: The convergence of the tap at index one over 256 time samples when $\mu = 0.02$.

domain implementation of the SAD algorithm does lag the time domain implementation by approximately 8 samples, this has very little effect on the actual filter value. This is because $\mu = 0.02$ has been chosen such that the number of iterations for convergence is far higher than the number of taps in the filter. Figure 5.3 shows that when the demixing filters converge, they both closely match the desired solution.

In order to show the problem that arises due to the assumed semi-stationarity of the demixing filters, the step sizes μ_1 and μ_2 for both algorithms were set to 0.2. It was found

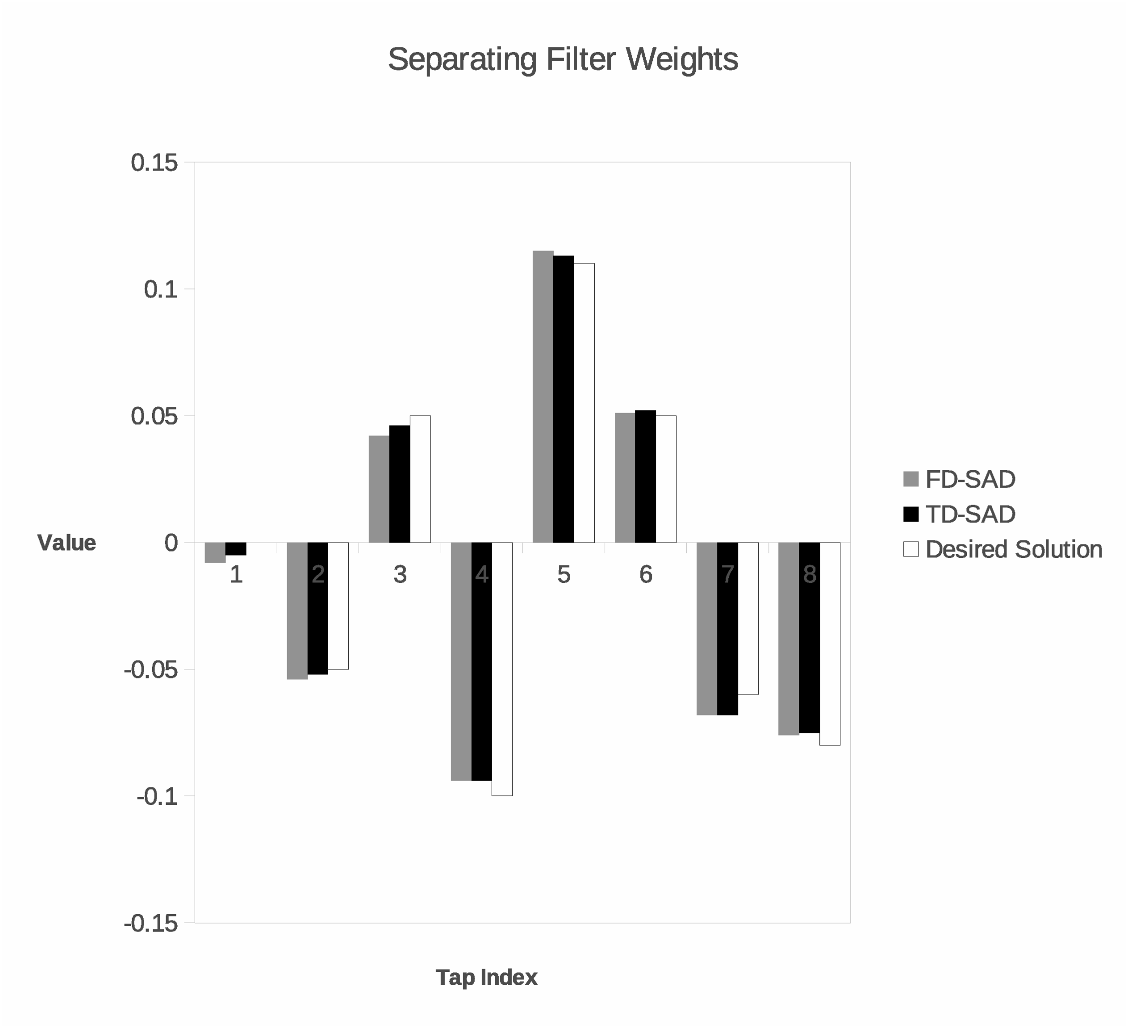


Fig. 5.3: The filter weights after 32 000 time samples with $\mu = 0.02$.

that the system lost stability when the adaptive steps sizes were increased to more than 0.5, so a value of 0.2 ensured stability while illustrating the problem described in section 5.3.

From Figure 5.4, one can see that the frequency domain algorithm lags the time domain algorithm by roughly 8 samples, similar to the case for a small adaptive step size (Fig. 5.1). In contrast to the earlier example, the difference between the values of the weights for the time-domain and frequency-domain algorithms is substantially larger. This can be seen in Fig. 5.5, where the time domain result does appear to match the desired result better than

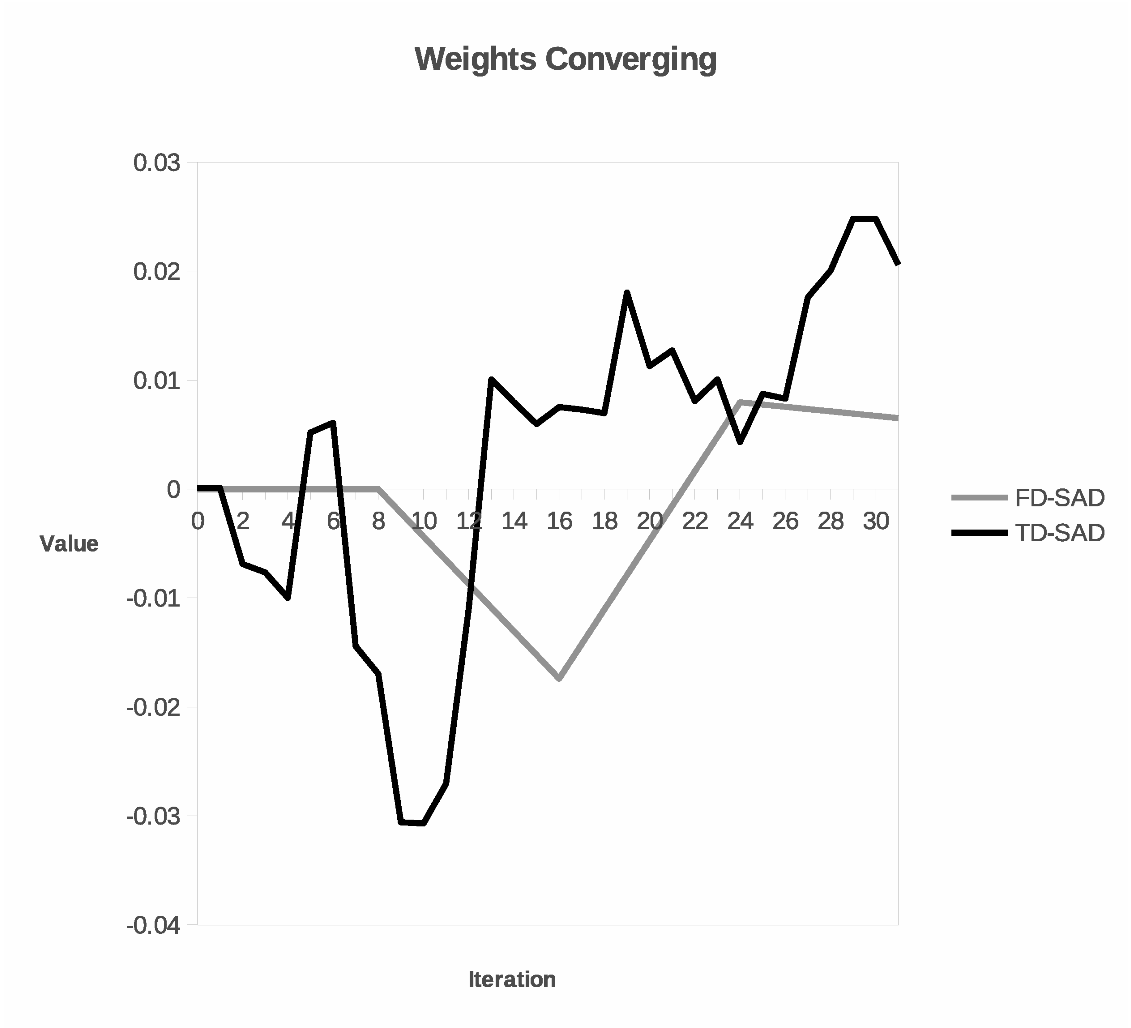


Fig. 5.4: The convergence of the tap at index one over 32 time samples when $\mu = 0.2$.

the frequency domain algorithm. However, figure 5.6 shows that when such an adaptive step size is chosen, the misadjustment (as described in [101]) means that there is still a significant error in the values of the filter taps long after the algorithms should have converged. The case as described in section 5.3 where the time domain implementation would have a significant advantage over the frequency domain implementation is therefore impractical from a convergence standpoint.

One case where the time-domain implementation still has an advantage over the frequency-

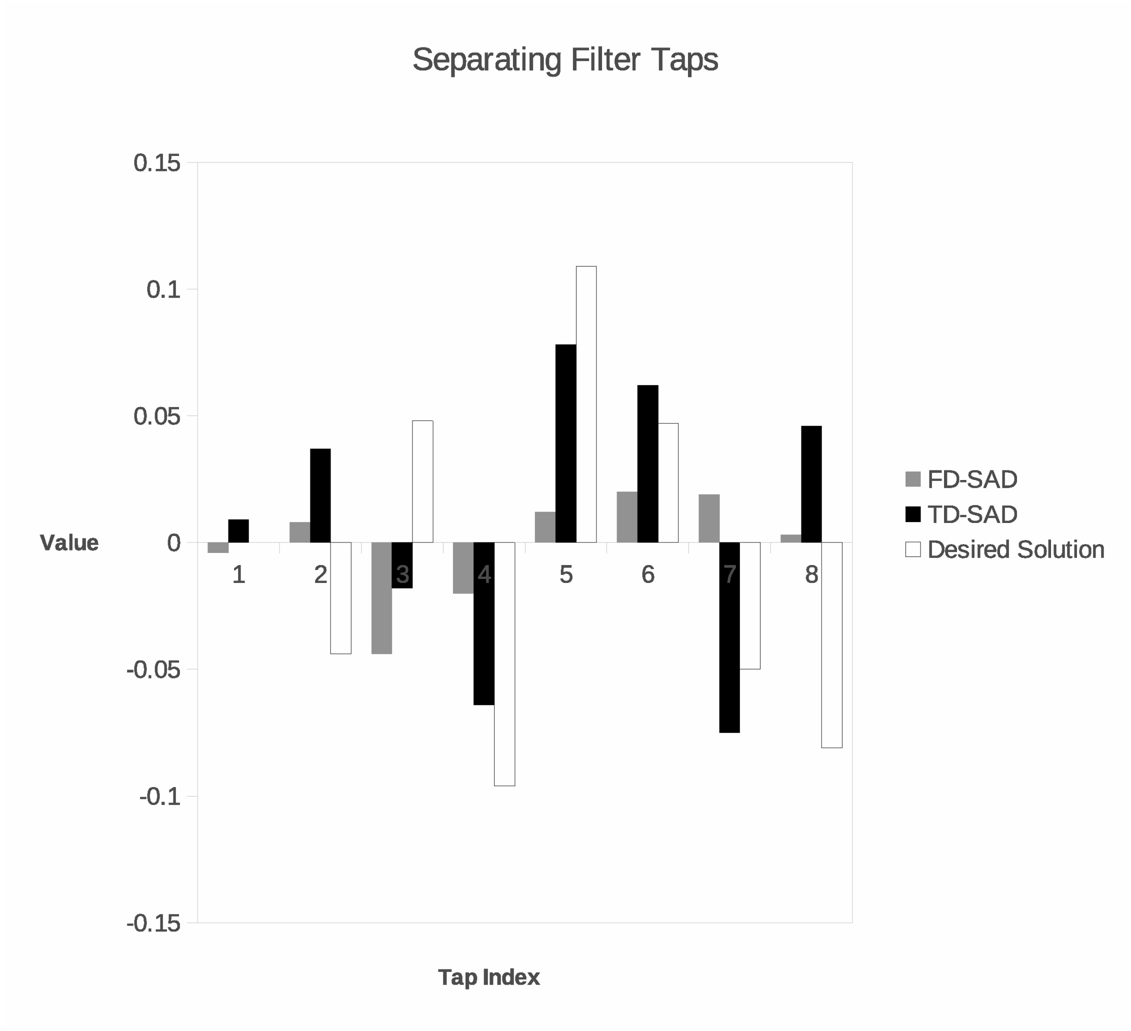


Fig. 5.5: The filter weights after 32 time samples with $\mu = 0.2$.

domain implementation is with applications where the separated signals are needed in real-time. Because of the block structure of the frequency domain implementation, the outputs are delayed by L samples, whereas in the time domain implementation the outputs are only delayed by 1 sample. For example, with an audio signal sampled at 44.1 kHz and filters of size 1024, the lag for the frequency domain algorithm will be 23 ms compared to the time-domain's 23 μs. Therefore the frequency-domain SAD algorithm may be ill-suited for some time-critical applications.

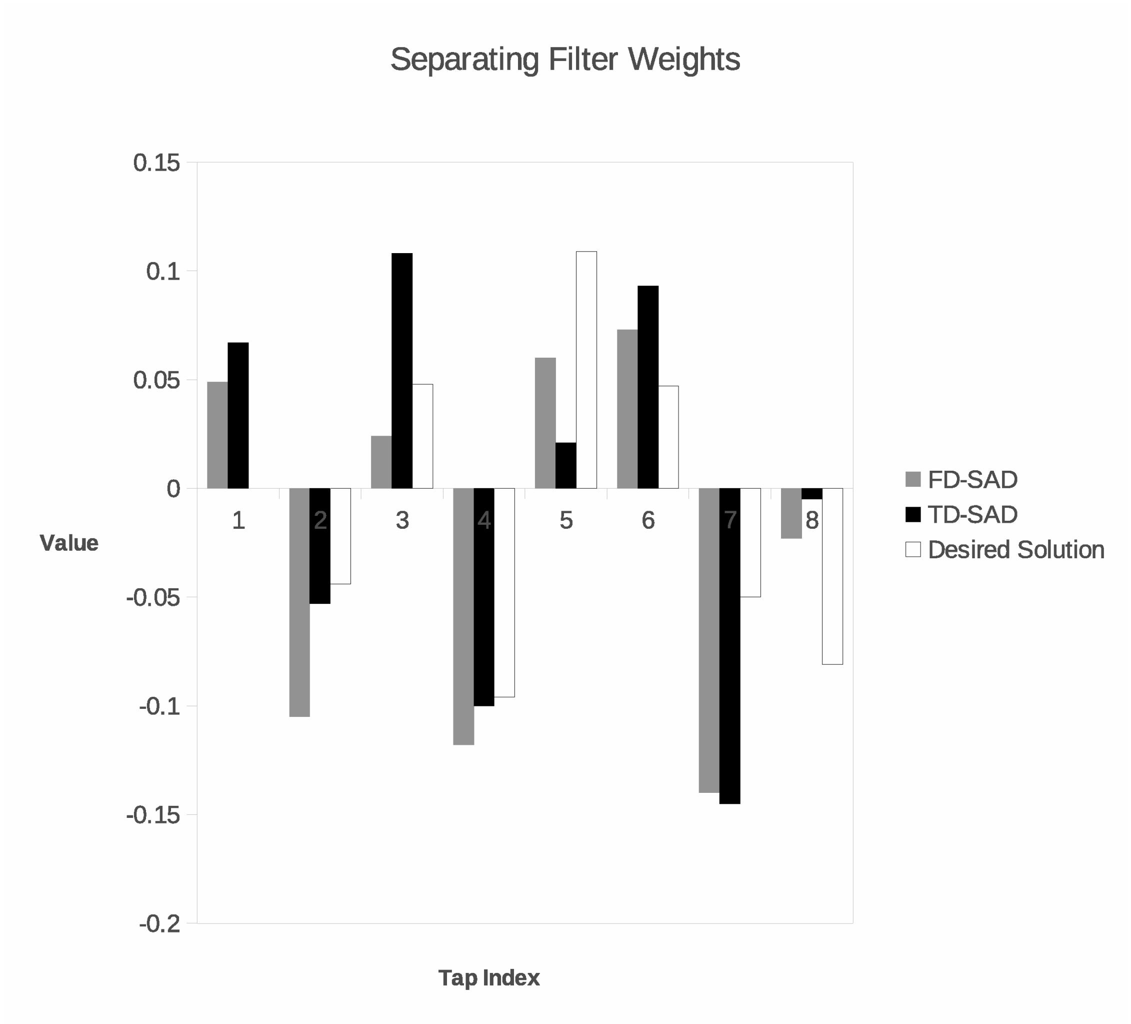


Fig. 5.6: The filter weights after 32 000 time samples with $\mu = 0.2$.

5.6 Conclusion

Because of the computational efficiency of the FFT, a frequency-domain implementation of the SAD algorithm is proposed which is more efficient than the time-domain equivalent, especially for higher-order separating filters and a greater number of sources. It was found that the frequency domain implementation was more efficient than the time-domain implementation when the filters were at least 16 samples long when separating convolutely mixed sources. Simulations showed that there is no significant difference in separation performance between the time domain and frequency domain implementations.

This, however introduces an input-output lag that means that the straight FD-SAD may be unsuitable for time-sensitive applications. For this reason, we investigate a hybrid method in the next chapter in order to reduce this lag without significantly affecting computational efficiency.

6. HYBRID FREQUENCY-DOMAIN TIME-DOMAIN SAD

A major advantage of the *frequency-domain symmetric adaptive decorrelation* (FD-SAD) algorithm over the *time-domain symmetric adaptive decorrelation* (TD-SAD) algorithm is that it is substantially more computationally efficient. However, it has the downside that the *outputs* lag the *observed signals* by L samples, whereas with the TD-SAD, the lag is only dependent on the speed of whatever is processing the signals. Therefore the TD-SAD is far better suited to real-time applications that need the output as early as possible. For example, with live music reproduction, even an echo that arrives greater than 20 ms after the line of sight signal can be intolerable for the audience [103]. With a sample rate of 44100 kHz a filter of length 1024 means that that the loudspeakers are reproducing the audio 23 ms after receiving it. This could potentially disconcerting to the audience, and even more so to the performers so should be avoided.

In this chapter is proposed an adaptation to the *symmetric adaptive decorrelation* (SAD) algorithm that exploits the fast Fourier transform (FFT) to create an algorithm that is more efficient than the TD-SAD but does not suffer from the same lag that the FD-SAD does. It would therefore be suitable for real-time applications as it has increased computational efficiency over the TD-SAD, but the delay from the observed signal to the output is only dependent on the speed of the processor.

6.1 Background Information

The hybrid SAD algorithm in this chapter is based heavily on the ordinary SAD algorithm, but is more computationally efficient than the time-domain counterpart (chapter 3) without having the delayed output associated with the FD-SAD algorithm (chapter 5).

The primary advantage of performing the separation in the frequency domain is that there is potential for significant computational savings. The main cause of these savings is due to the fact that the convolution and correlation operations in the time domain map to element-wise multiplication in the frequency domain. For the SAD algorithm, there are two instances where the computational savings of this transformation could be advantageous. The first is in the convolution operation for the filtration of each observed signal through each separating filter, and the second is in the correlation operation in the actual update equations.

However, there are also a few disadvantages in implementing the algorithm in the frequency domain. One disadvantage is due to the block processing delaying the output by up to the block size. This may become a problem where the outputs are needed in real-time.

Another problem is that the upper limit on the adaptive step size μ must be scaled down by a factor of L. This results in slow convergence if the observed signal correlation matrix has a large eigenvalue spread.

In this chapter, we propose a hybrid frequency-domain time-domain approach that overcomes the first problem by performing the filtering in the time domain but the update in the frequency domain. While this approach is marginally more computationally expensive than the pure frequency domain approach, it is significantly less computationally expensive than the time-domain approach without being subject to the lag associated with the pure frequency domain approach. Therefore, such a hybrid approach would be suitable for applications such as live sound mixing, but still have substantial computational advantages over a purely time-domain approach.

6.2 The Hybrid Algorithm

The block time-domain update equation is given in (5.7) as

$$\mathbf{w}_{i,j}\left(kL+L\right)=\mathbf{w}_{i,j}\left(kL\right)-\frac{\mu}{\sigma^2}\nabla\mathbf{w}_{i,j}\left(kL\right) \tag{6.1}$$

where

$$\nabla\mathbf{w}_{i,j}\left(t\right)=\sum_{l=0}^{L-1}y_i\left(t+l\right)\mathbf{y}_j\left(t+l\right). \tag{6.2}$$

However, this is only the update for the filter itself, not the outputs. Therefore a trivial solutions would be to update the demixing filters on a block-by-block basis using frequency-domain correlation, but filtering the observed signals at every sample. This would allow for some of the computational savings available to the frequency domain components of the hybrid algorithm, without introducing the length-L lag. Therefore the block output equation in (5.15) can be replaced with the time domain outputs as described in (3.15)

$$y_i\left(t\right)=\sum_{j=0}^{N-1}\mathbf{x}_j^T\left(t\right)\mathbf{w}_{i,j}\left(t\right) \tag{6.3}$$

As the filter size begins to increase, a larger and larger percentage of the computing requirements is needed for the filtering rather than the update because the update becomes more efficient for larger filter sizes. In the TD-SAD algorithm, roughly half of the computing resources are required for the filtering, and the rest are required for the update, but the long-filter efficiency of the new algorithm means that this is not the case. As one would expect, as the filter size increases, the computational requirements of the hybrid algorithm as it stands has a lower bound of half the requirements of the time-domain algorithm.

Another interesting point is that the FD-SAD algorithm actually increases efficiency over the TD-SAD algorithm as the number of *sources* to separate increase. The main reason for this is that there are $N\left(N-1\right)$ convolution operations required for the filtering portion of the algorithm, but because the frequency-domain forms of the input signals can be stored, only $2N$ FFTs are required, cutting down substantially on the computational requirements as the number of sources increase. Due to the fact that the suggested algorithm only exploits frequency domain efficiencies for the update of the adaptive filter, the increase in efficiency as the number of sources increses is not nearly as significant as that for the FD-SAD. If it is possible to utilize some frequency domain techniques for the filtering as well as the update, the computational savings for the hybrid algorithm would be even more substantial. These savings would not only be seen as the demixing filter sizes increase, but also as the number of sources increase.

In [4], Gardner compares filtering in the frequency domain to filtering in the time domain and proposes a novel algorithm that utilizes the FFT while suffering from no input-output lag. He points out that because the convolution operation is a linear operation, and the *discrete Fourier transform* is also a linear operation, the convolution operation can be par-

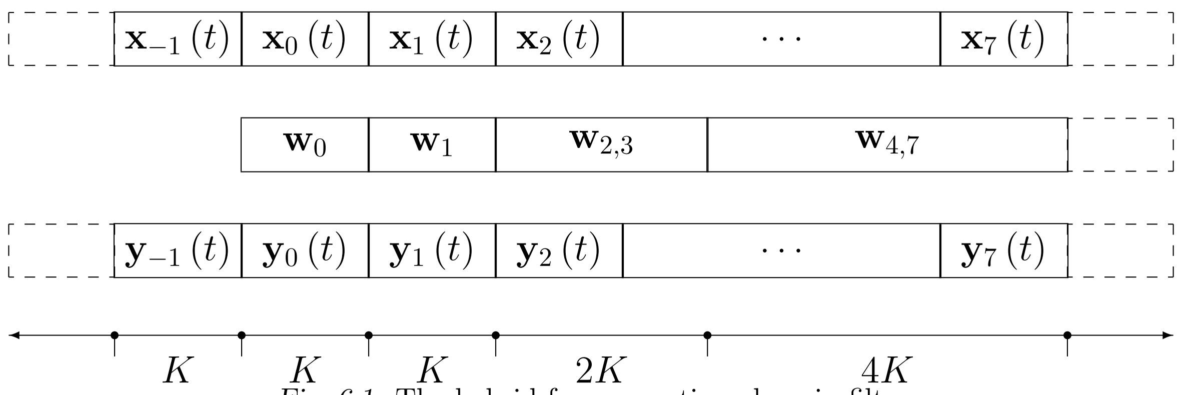


Fig. 6.1: The hybrid frequency-time domain filter.

tially computed in the time domain and partially computed in the frequency domain, then combined later to give the full convolution. However, Gardner's algorithm was designed for simulation, which involves *static* filters. Because SAD algorithm is an adaptive filter, further investigation is needed to ensure that his proposals can still be used.

In fact, for the straight TD-SAD, because the filter is getting adjusted for every time-sample, these frequency-domain techniques can no longer apply, unless the TD-SAD algorithm is first converted into a block algorithm. In which case it is mathematically identical to the hybrid algorithm, but with greater computational requirements for the update.

Consider Fig. 6.1, where the observed signal x, output y and demixing filter w are all divided into blocks of size K, where K is the size at which convolution in the frequency domain becomes more efficient than direct convolution. This will normally occur with a block size of 32 or 64. The double subscript $w_{i,j}$ indicates that it is the block that is made up of all size-K blocks from w_i to w_j.

The output $y(t)$ can be found by the following convolution

$$\mathbf{y}_i(t) = \mathbf{w}_{0,\frac{L-K}{K}} * \mathbf{x}_{i-\frac{L}{K},i}(t) \qquad \forall i \tag{6.4}$$

where $*$ denotes the convolution operation.

Because of the linearity of the convolution operation, and for simplicity taking the case of $i = 0$, (6.4) can be decomposed to the following sum

$$\mathbf{y}_0\left(t\right)=\mathbf{w}_0 * \mathbf{x}_{-1,0}\left(t\right) + \mathbf{w}_1 * \mathbf{x}_{-2,-1}\left(t\right) + \ldots + \mathbf{w}_{\frac{L-K}{K}} * \mathbf{x}_{-\frac{L}{K},\frac{K-L}{K}}\left(t\right). \tag{6.5}$$

Each of these individual convolutions do not have to be calculated the same way. If K is the number at which convolution in the frequency domain becomes more efficient than convolution in the time domain, then it is desirable to do as many of these convolutions as possible in the frequency domain. To have zero lag, the first convolution must be done in the time-domain, as the block $\mathbf{x}_0\left(t\right)$ is not known until after $\mathbf{y}_0\left(t\right)$ is needed. However, all other convolutions can be completed in the frequency domain, increasing computational efficiency.

As the size of the block increases, the computational savings in using frequency-domain convolution also increases. Using this, (6.5) can also be calculated more efficiently.

$$\begin{aligned}\mathbf{y}_0\left(t\right)=&\mathbf{w}_0 * \mathbf{x}_{-1,0}\left(t\right)\\ &+\mathbf{w}_1 * \mathbf{x}_{-2,-1}\left(t\right)\\ &+\mathbf{w}_{2,3} * \mathbf{x}_{-4,-2}\left(t\right)\\ &+\mathbf{w}_{4,7} * \mathbf{x}_{-8,-4}\left(t\right)\\ &+\ldots\\ &+\mathbf{w}_{\frac{L}{2K},\frac{L-K}{K}} * \mathbf{x}_{-\frac{L}{K},\frac{-L}{2K}}\left(t\right)\end{aligned} \tag{6.6}$$

Note that for all of the convolutions in (6.6), only K samples of the result will be needed. However, many of the unused samples will be needed in the calculation of future blocks. For example in calculating the term

$$\mathbf{w}_{2,3} * \mathbf{x}_{-4,-2},$$

one can actually calculate the following

$$\mathbf{w}_{2,3} * \mathbf{x}_{-4,-1},$$

but use the following K samples of the result to aid in the calculation of $\mathbf{y}_1(t)$, thus increasing computational efficiency further. The pseudocode for the hybrid algorithm is shown in Appendix B.4.1.

6.3 Experiments

6.3.1 Comparison between SAD approaches

In order to evaluate the efficacy of the hybrid algorithm in comparison to the time domain and frequency domain algorithms, all three algorithms were used to increase the *signal to noise ratio* (SNR) of a ten-channel live recording that was 2 minutes, 40 seconds long. The channels were as follows:

1. Bass drum
2. Snare drum
3. Hi-hat
4. Hanging tom
5. Floor tom

6. Bass guitar
7. Electric guitar
8. Backing vocal
9. Backing vocal
10. Lead vocal

and the separating filters were all of length 1024, as smaller filter sizes did not perform as well, but there was no significant increase in performance for larger filter sizes.

Because the individual components of each channel were not available (as in chapter 3) the SNR could not be directly calculated, but had to be estimated. The SNR formula in (3.51) is given as

$$\mathrm{SNR} = 10\log_{10}\left(\frac{P_S}{P_N}\right) \tag{6.7}$$

The power of the signal P_S and the power of the noise P_N have to be estimated by taking regions that are approximately noise-free (for P_S) and signal-free (for P_N). While it was easy to find periods where the observed signal was signal-free, it was more difficult to find regions that had only the desired signal with no noise. What compounded this problem was the fact that both the noise and desired signal were nonstationary, so even using the noise power from the signal-free sections as an estimate of the noise component of the power when there is also a desired signal component could not be completely justified.

However, because the main purpose of these experiments is to compare the performance of the three different algorithms against each other, rather than with the variety of other algorithms available, the absolute value of the SNR is less important that the consistency of its calculation between the three cases. As a result, for the purposes of this evaluation, we

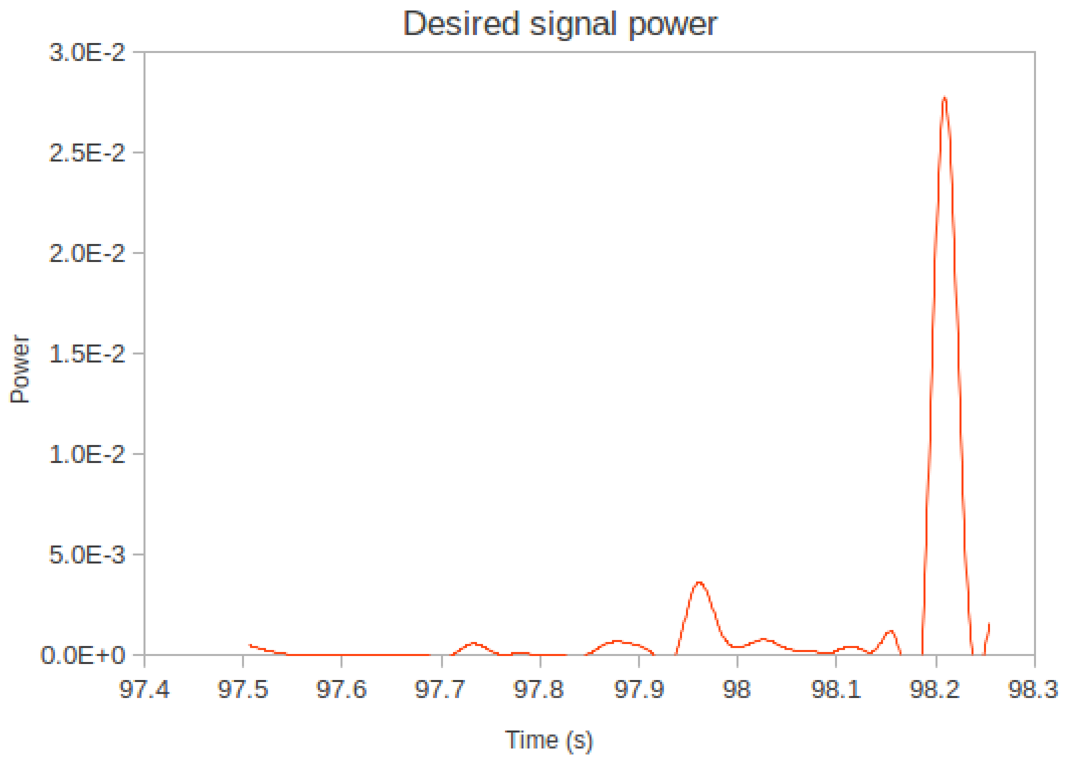


Fig. 6.2: The desired component of the hanging tom microphone.

define the approximate SNR as

$$SNR_{approx} = 10\log_{10}\left(\frac{P_{S+N,1}}{P_{N,2}}\right) \tag{6.8}$$

where $P_{S+N,1}$ is the combined power of the signal and noise at time t_1 and $P_{N,2}$ is the power of only the noise at time t_2, and t_1 and t_2 are chosen such that the desired signal is the primary component at t_1, and there is no desired component at t_2. The power of the observed signals and outputs was calculated using exponential smoothing with a smoothing factor of 0.01.

Two of the ten channels will be analysed; a backing vocal microphone, and the hanging tom drum microphone. The backing vocal microphone selected was the vocal microphone that audibly had the most crosstalk from the other instruments, primarily the drums. The

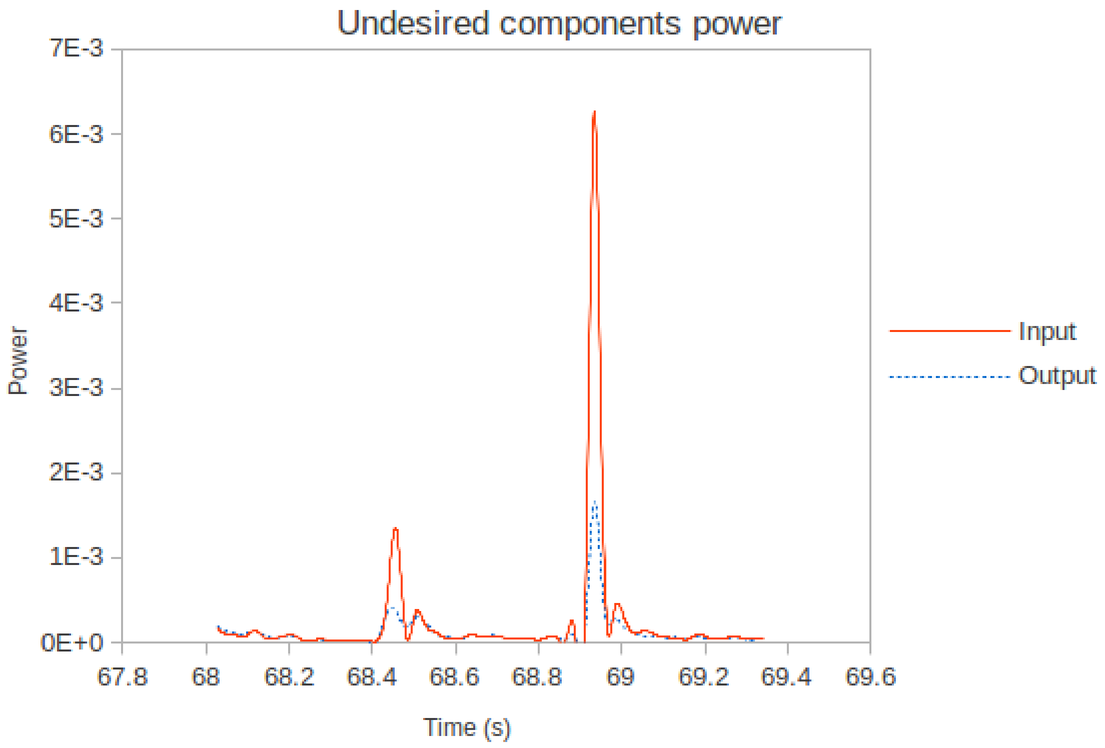


Fig. 6.3: A signal free portion of the hanging tom microphone.

tom drum microphone was chosen because it was one of the microphones most dissimilar to the vocal microphone; its close proximity to the other drums meant that the SNR was substantially lower, and also the time taken for the noise from the other undesired drums to reach it was far less, meaning that earlier taps of the demixing filters would be used.

Figures 6.2-6.5 show the powers of short selections of output signals for two different channels of the demixing system. Fig. 6.2 shows a sample of the exponentially smoothed power of the tom microphone where the main component of the signal is the tom drum itself. Although this only actually shows the observed signal, the outputs of all of the algorithms were visually indistinguishable from it. Due to the high SNR of the observed signal, this is actually a good sign as it indicates that the desired component of the output is a fairly close representation of the desired component of the observed signal. This would be a highly

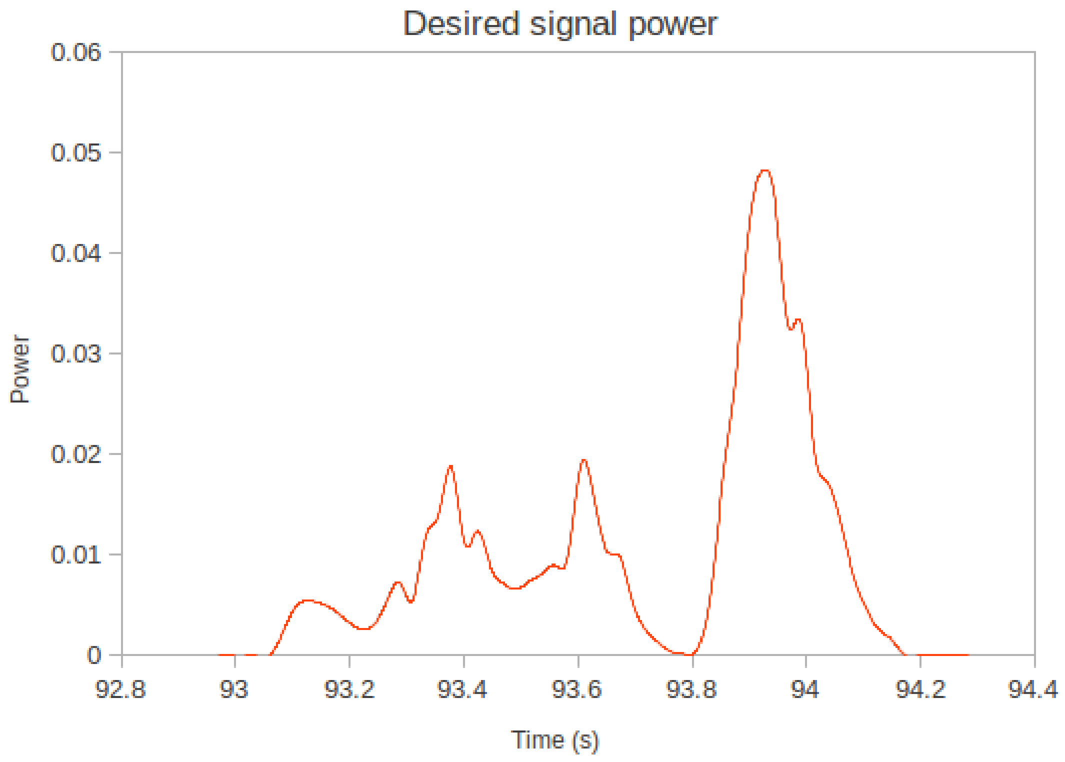


Fig. 6.4: The desired component of the backing vocal microphone.

desirable characteristic of an algorithm that is used for sound reproduction.

Fig. 6.3 shows a portion of the tom drum channel where there is no desired component. Again, because the algorithms behaved so similarly, there was no reason to put all of them on the same graph, as it would be nearly impossible to differentiate them. Therefore, only the observed signal and output powers have been plotted.

Fig. 6.4 and 6.5 show the desired and undesired powers respectively of the backing vocal microphone. There is not quite the same attenuation in the undesired component when compared to the tom microphone, but it is still fairly significant. There are a couple of possible reasons for this. The power of the undesired component was far less significant at the backing vocalist's microphone than what it was at the tom microphone. This would have

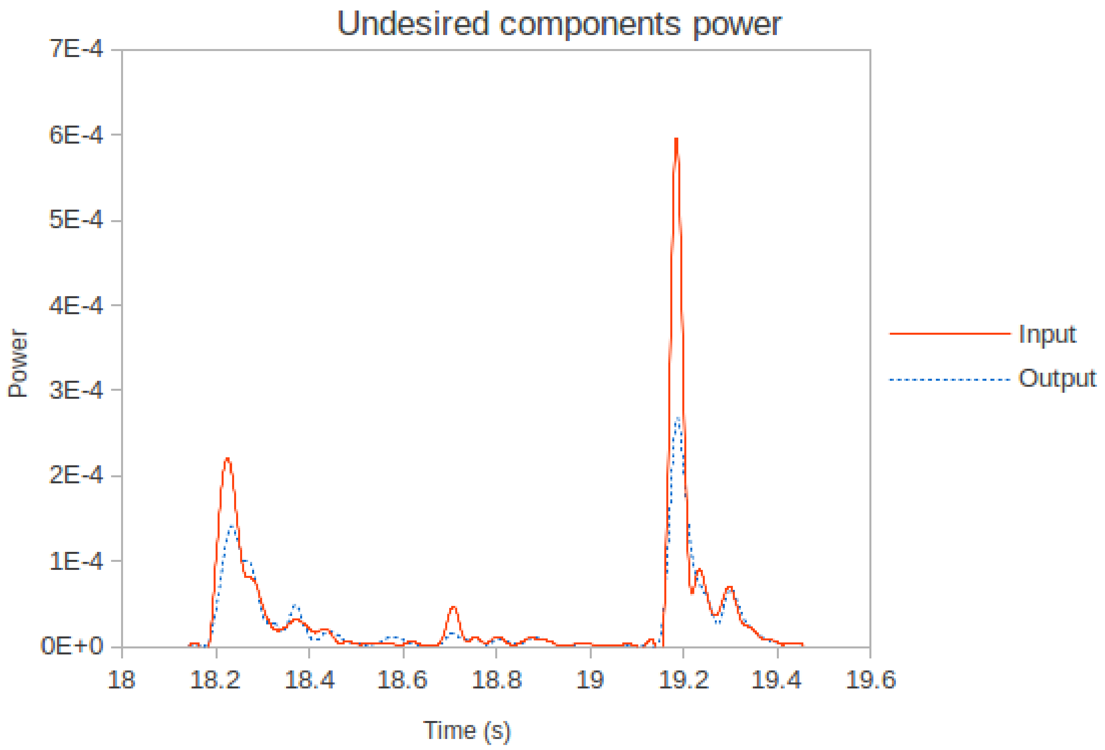


Fig. 6.5: A signal free portion of the backing vocal microphone

Tab. 6.1: Comparison of SNR$_{approx}$ of a tom drum microphone

Input	6.33 dB
TD-SAD	11.79 dB
FD-SAD	11.77 dB
Hybrid SAD	11.77 dB

contributed toward slower convergence of the algorithm, as its convergence rate depends on the power of the observed signals. Also, when listening to the track, it could be heard that the most significant undesired noise came from the drums. The time lag for this would be more significant than with the tom drum's microphone, and the separating filters were limited to 1024 taps, which may not have been enough to remove all of the drum's residual reverberation.

Tab. 6.2: Comparison of SNR$_{approx}$ of a backing vocal microphone

Input	18.59 dB
TD-SAD	20.98 dB
FD-SAD	20.96 dB
Hybrid SAD	20.96 dB

Tables 6.2 and 6.1 show the SNR$_{approx}$ of the observed signal and outputs for each of the algorithms. This was calculated using (6.8) where the power of the signal plus noise was taken at the peak at 98.2s for the tom microphone, and the peak at 94.9s for the backing vocal microphone. The power of the noise alone was taken as the peak at 68.9s for the tom microphone and at 19.2s for the backing vocal microphone. These tables show that there is only a trivial difference between the performance of the algorithms.

6.4 Computational Cost

In Appendix C.1, it is shown that the computational cost for L updates of the TD-SAD algorithm with N observed signals and demixing filter length L is

$$LN(N-1)(2L+1) \quad \text{real multiplications} \tag{6.9}$$

per observed signal, whereas the equivalent for the FD-SAD algorithm is shown in Appendix C.2 to be

$$2(N+1)(2L\log_2 L + 3L + 6) - 31L \quad \text{real multiplications} \tag{6.10}$$

per observed signal.

Tab. 6.3: Multiplications Required for $N = 2$

Filter order	TD-SAD	FD-SAD	Hybrid SAD
16	1.22E3	1.19E3	1.23E3
32	4.61E3	3.08E3	3.18E3
64	1.77E4	0.76E4	0.79E4
128	6.86E4	1.82E4	1.87E4
256	2.69E5	0.42E5	0.44E5
512	1.06E6	0.10E6	0.10E6
1024	4.23E6	0.22E6	0.22E6
2048	1.69E7	0.05E7	0.05E7

According to the calculations shown in Appendix C.3, the algorithm needs

$$\begin{aligned}
&2 \sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k + \log_2 K\right) - 5L + \frac{6L}{2^k K} \\
&+ (N-1)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1} K\left(k + \log_2 K\right) - 5 \times 2^k K + 6\right) \\
&+8NL\log_2 L + \left(\hat{K} - 4\log_2 \hat{K} - 2\right) NL \\
&-4L\log_2 L - \left(\hat{K} - 4\log_2 \hat{K} + 7\right) L + 12N \qquad \text{real multiplies} \qquad (6.11)
\end{aligned}$$

for L time samples of each observed signal, where K is the block size at which frequency-domain convolution becomes more efficient than time-domain convolution. It was found that depending on L and N, this was either 8, 16, or 32. Because there was little difference, we will assume that $K = 16$. $\hat{K}$ is used to denote the minimum of K and L.

Table 6.3 shows a comparison of the multiplications required for varying filter lengths when separating two sources. Even for small filter sizes the hybrid SAD algorithm shows similar computational requirements to the TD-SAD algorithm. As the filter sizes get larger, the hybrid SAD algorithm nearly matches the FD-SAD algorithm in computational complexity.

Tab. 6.4: Ratio of Multiplications Required

Length	2 sources		4 sources		8 sources	
	FD/TD	Hyb/TD	FD/TD	Hyb/TD	FD/TD	Hyb/TD
16	0.98	1.01	0.77	0.93	0.70	0.90
32	0.67	0.69	0.49	0.61	0.43	0.58
64	0.43	0.45	0.30	0.38	0.26	0.36
128	0.27	0.27	0.17	0.23	0.15	0.22
256	0.16	0.16	0.10	0.13	0.09	0.13
512	0.09	0.09	0.06	0.08	0.05	0.07
1024	0.05	0.05	0.03	0.04	0.03	0.04
2048	0.03	0.03	0.02	0.02	0.01	0.02

Table 6.4 shows the ratios of multiplications required for each of the algorithms against the TD-SAD algorithm for varying filter sizes and number of observed signals. This table also show that it is for the longer filter sizes that the computational savings of the hybrid algorithm start to take effect. It also shows that as the number of sources increase the efficiency (in comparison with the TD-SAD algorithm) increases. While the increase in efficiency as the number of inputs increases does not match the increases seen for the FD-SAD algorithm, they are still significant. The reason for this is that for the $\mathcal{O}(N^2)$ convolutions required for the filtering portion of each algorithm, the FD-SAD algorithm only needs $\mathcal{O}(N)$ FFTs to transform the observed signals into the frequency domain. The hybrid algorithm also only needs $\mathcal{O}(N)$ FFTs to transform the observed signals into the frequency domain. However, the FD-SAD algorithm already has all of the $\mathcal{O}(N^2)$ *filters* in the frequency domain from the update portion of the algorithm. Because the hybrid SAD algorithm cannot use the length-$2L$ frequency domain representations of the demixing filters, it also must perform $\mathcal{O}(N^2)$ FFTs to transform the filters to the frequency domain for frequency-domain convolution.

Both algorithms require $\mathcal{O}(N^2)$ FFTs for the update. For the filtering portion of the algorithm, the FD-SAD algorithm only requires $\mathcal{O}(N)$ FFTs, while the hybrid SAD algorithm requires $\mathcal{O}(N^2)$ FFTs.

6.5 Future Work

One problem with the hybrid algorithm is that it does not have a constant computational load. For example, it needs to be able to calculate the FFT for the previous block before the first sample of the next block is actualy received. This means that, while it is substantially more efficient overall than the time-domain approach, the computational load changes with time. To run such an algorithm on low-specification hardware would require an input-output delay due to that fact that it wouldn't be able to reliably calculate the output within one sample. There would be two ways to approach this problem.

The first would be to actually introduce an input-output delay that gave time for the hardware to perform the necessary computations. This approach would still reduce the computational requirements of the algorithm when compared to the time-domain approach, but it would be at the cost of an input-output delay.

The second (and more robust) approach would be to increase the minimum subblock size [4]. This would slightly increase the *overall* computational complexity of the algorithm, but also gives more time during computationally-intensive periods to perform the necessary calculations.

A fundamental flaw with the current approach is that SAD is one of the earlier attempts to achieve *blind source separation* (BSS). As a result, its separation performance in comparison with more modern algorithms is somewhat lacking. Due to its reliance on only second-order statistics (SOS), and its basis in the *least mean squares* (LMS) algorithm, it *is* a good starting point, as, in comparison with more recent attempts at BSS, it tends to be quite computationally efficient. If computational efficiency is the desire, this justifies its use as a test-bed for future approaches. The first step in development of a better approach would be to find a more recent algorithm that also does its update and filtering in the frequency domain and has a computational cost on the same order as SAD, but separates with markedly better results.

One such approach could be *Triple-N ICA for Convolutive mixtures* (TRINICON) [1], which is a BSS method that can also be based solely on SOS. TRINICON is a method that exploits the three 'N's (non-stationarity, non-Gaussianity, and non-whiteness) in an attempt to improve separation performance. The following section shows a comparison of TRINICON and SAD, with respect to separation performance, convergence rate, and computational complexity.

6.5.1 Experimentation

Because it is a more recent approach, and is fairly well-known, TRINICON was deemed a good choice to be a comparison. Also, due to the fact that it can be implemented using solely SOS, this similarity to SAD made it a good option as an alternative algorithm. The implementation itself was made available by Anderson *et al.* [47], and was run using MATLAB.

The experimentation for comparing the proposed algorithm with TRINICON was completed on audio samples obtained by Sawada *et al.* in [104]. These samples included separate recordings for each channel; there was a file for each component of every observed signal.

Because of the restriction on SAD that each source must be the closest source to a unique receiver, this allowed the reconstruction of a mixing system with an arbitrary distance between microphones. For the case of this experiment, the distance chosen was one meter. This meant that there could be an accurate calculation of the SNR, as each output could be decomposed into components that were dependent only on individual sources. Therefore the equation in (6.7) could be used directly, rather than the approximation in (6.8).

Whereas the previous comparison between the different hybrid SAD approaches could simply use the same step size, the choice of step sizes for the comparison between hybrid SAD and TRINICON may greatly influence the results. There needs to be a method of choosing step sizes which lessens the effect of its choice.

For this reason, the separation of the mixed signals was performed using both the hybrid SAD and TRINICON approaches, with varying step sizes. The increase in SNR for each case was measured. It was found that the increase in SNR hit an upper limit, and as the step size was increased further, the system lost stability. The step size was then taken as the value which gave a maximum SNR near this limit, and had a high convergence rate, but did not have any visual signs of instability.

Both algorithms were passed multiple times over the mixture, each time initializing the filters to the final values of the previous run. The SNR was calculated for each epoch, and the results are plotted on Fig. 6.6.

This shows how TRINICON produces far better results than the proposed algorithm. The increase in SNR is on the order of 20 dB, whereas for the hybrid SAD algorithm it is on the order of 5 dB. From this graph, the only apparent advantage that the hybrid SAD

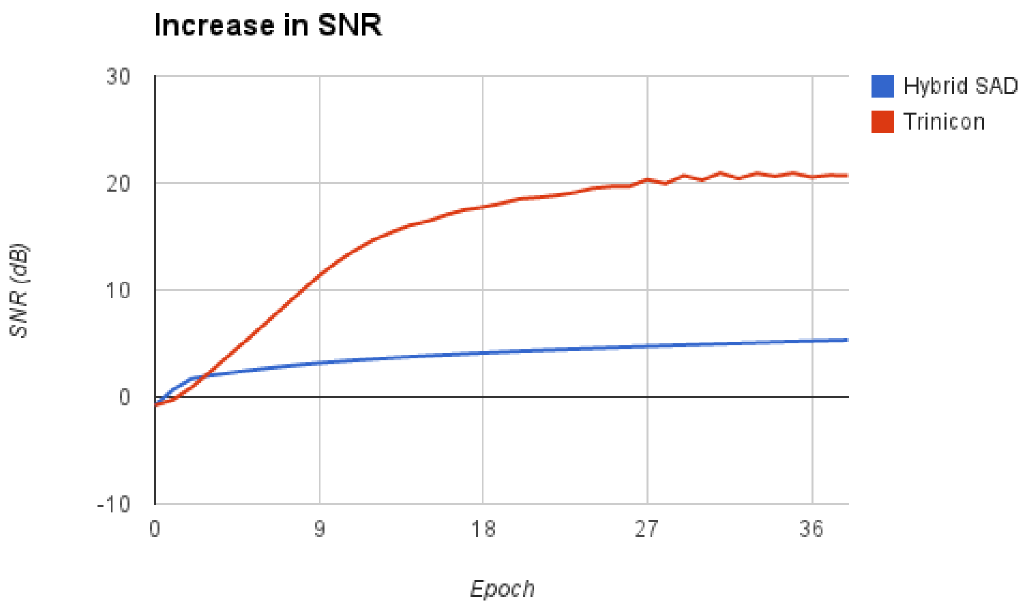


Fig. 6.6: The increase in SNR for repeated runs over a sample mixture.

approach has is that it appears to converge to its final value more quickly, especially right at the beginning. With informal listening tests, the TRINICON method also appeared to produce markedly better results.

It is important to note, however, that the hybrid SAD approach is a truly real-time approach. Both methods exploit the computational advantages provided by the FFT algorithm, but it is only with TRINICON that there is an input-output delay.

It is, however, feasible that the same frequency-domain adaptations made to the SAD algorithm to eliminate the input-output delay could also be made to the TRINICON algorithm. This would require further investigation.

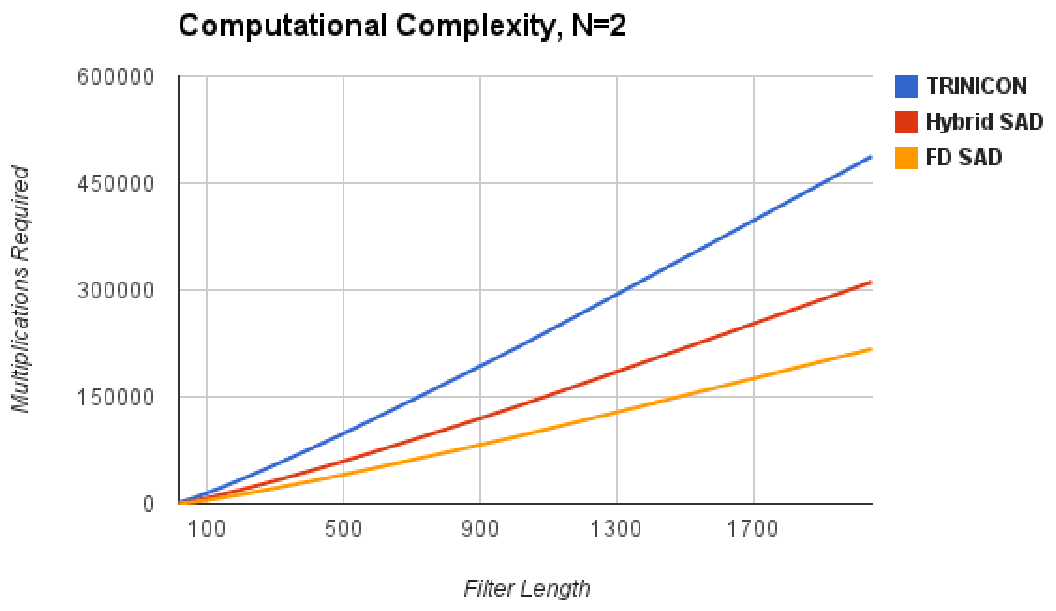


Fig. 6.7: Multiplications required for separating two sources.

6.5.2 Computational Complexity

In order for an approach such as TRINICON to be viable as an alternative to SAD as a basis for these frequency domain adaptations, its computational complexity needs to be roughly the same as that of FD-SAD. In this section, we show that it is.

The formulae for calculation of computational complexity for both FD-SAD and hybrid SAD are given in (6.10) and (6.11). For TRINICON, the computational complexity as calculated in Appendix C.4 is

$$6LN^2 \log_2 L + 81LN^2 + 18N^2 - 2LN \log_2 L - 43LN - 6N \qquad \text{real multiplies} \quad (6.12)$$

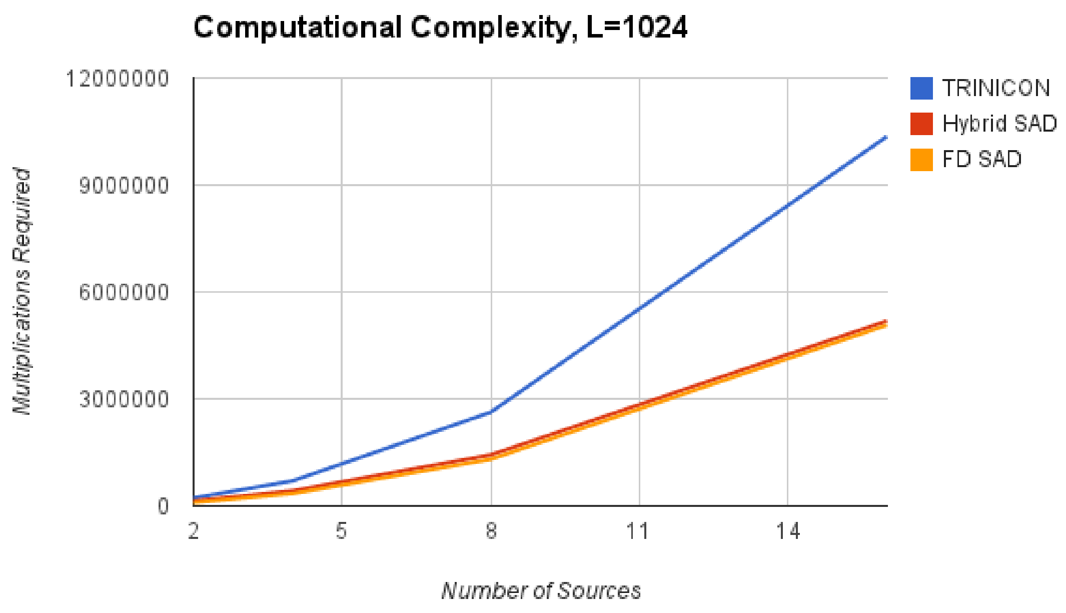


Fig. 6.8: Multiplications required for separating with filter length = 1024.

The computational complexity for TRINICON was higher than that of both the FD-SAD and the hybrid SAD. Fig. 6.7 shows the number of multiplications required as the filter size increases for all three. This was based on a mixing system with two sources. Trinicon requires the most computations, but it still appears to be on the same order as both FD-SAD and hybrid SAD.

Fig. 6.8 shows the number of multiplications required for all three algorithms as the number of sources increase. This was calculated using a filter length of 1024. Again, TRINICON requires the most computations, however, we see the gap between FD-SAD and hybrid SAD decrease as the number of sources increases. If a hybrid TRINICON approach was constructed, one might anticipate to see a similar merging of computational complexity of the hybrid TRINICON and pure frequency-domain TRINICON implementations.

6.6 Conclusion

The hybrid frequency-domain time-domain SAD algorithm exploits some of the computational efficiency of the FD-SAD algorithm without having the associated delay in output. This makes it more relevant for time-critical applications than the FD-SAD algorithm, while being more computationally efficient than the TD-SAD algorithm. Experiments have shown that there is no decrease in separation performance, and a computational complexity analysis shows that it nearly matches the increase in efficiency seen by the FD-SAD algorithm, especially when the number of sources to separate is small. Future work could include the incorporation of frequency-domain techniques investigated in this chapter to other, more modern algorithms (for example, TRINICON).

7. CONCLUSION

This work describes a number of extensions to the *symmetric adaptive decorrelation* (SAD) algorithm. The first extension focuses on the scalability of the algorithm by increasing the number of inputs and outputs. Constraints that were on the *two-input two-output* (TITO) system no longer hold for the *multiple-input multiple-output* (MIMO) case, so a new set of constraints was defined. An analysis was undertaken on the convergence properties of the algorithm which ensures that stability is maintained if the new constraints were met. Simulations were run to check the effectiveness of the multi-channel algorithm, and these simulations showed good increases in *signal to noise ratio* (SNR).

A SAD algorithm based on *vector least mean squares* (VLMS) is also described, which increases in computational efficiency as the number of sources increase. It splits the sources using a divide-and-conquer approach similar to how the fast Fourier transform (FFT) algorithm achieves its computational efficiency. Simulations again show increases in SNR at least as substantial as the scalar SAD algorithm.

The second aim of this work was to provide adaptations to the algorithm that increased its computational efficiency with applicability to real-life situations. The main cause of the increases in efficiency was due to frequency domain processing. Fast convolution and fast correlation could be achieved by exploiting the FFT, which the transforms relatively computationally intensive convolution and correlation operations into the far simpler operation of element-wise multiplication.

The straight *frequency-domain symmetric adaptive decorrelation* (FD-SAD) algorithm has a significant downside, however. While the *time-domain symmetric adaptive decorrelation* (TD-SAD) algorithm is substantially more computationally intensive, especially for longer filter sizes, it is possible to obtain the output with minimal lag to the input. However, because it is block-processed, the FD-SAD algorithm must wait for the entire block of input data to be processed before it can obtain the separated signals. This could potentially be a problem in real-time applications.

The first step in overcoming this is through the realization that it is only the filtering portion of the SAD algorithm that means that the output is delayed. The update itself has no bearing on the delay so can successfully use blockwise processing without any lag. For the filtering portion of the algorithm, the filter can be divided up into subblocks of increasing length and still use frequency domain techniques on most of these subbblocks without introducing any delay. This new hybrid algorithm has a computational complexity that is only slightly greater than the FD-SAD, but has the potential to be used in a real-time environment.

All three algorithms were run on real recordings, and the results compared. There was no discernible difference in separation performance, but as expected, the two latter algorithms completed substantially more quickly.

Possible future work could include combining the frequency-domain adaptations with the VLMS adaptations in order to increase the performance of the algorithm even further. Because the VLMS primarily gets an increase in computational efficiency as the number of sources increase (rather than an increase in the length of the filters), it would be of interest how this translated when also using the frequency domain techniques which see performance gains for increases in both. It would also give the algorithm a more modular structure, which would give greater flexibility in real environments.

Another significant aspect of the hybrid approach is that, although it can have no lag, it does not have a constant computational load. The responsibility of this irregular load is primarily on the update portion of the algorithm. When the final sample of a block is received, a FFT needs to be performed on that block before the first sample of the next block is received. If it is possible to filter the signal with a method similar to how the update is performed for the hybrid algorithm, the computational load would likely be more uniform. This would also offer the opportunity to update some taps in the filters more regularly than once every block, which could potentially improve convergence properties.

Now that the basis for frequency-domain adaptations to a *blind source separation* (BSS) algorithm has been established, these techniques could also be extended to other frequency-domain algorithms. *Triple-N ICA for Convolutive mixtures* (TRINICON), for example, is a BSS approach that can also be based on second-order statistics (SOS), thus the convolution operations in the time domain can also be calculated using element-wise multiplication in the frequency-domain. This particular approach could also utilize the proposed adaptations to allow an efficient algorithm with zero input-out lag, and potentially have significantly greater performance than is acheived by SAD.

One application which could use such hardware would be live sound mixing. The main constraint of the SAD algorithm is that each *source* must be the closest source to a unique receiver, and for live sound mixing this requirement will almost always be met. Because the hybrid algorithm is substantially more efficient than the TD-SAD algorithm, it would provide a means of separation that would require less expensive hardware. And although not quite as efficient as the pure frequency domain algorithm, the hybrid algorithm does not suffer from the end-to-end lag, making it far more suitable for a real-time application such as live sound mixing. However, in order to have a truly real-time implementation on low-specification hardware, the problem of constant computational load would need to be solved.

APPENDIX

A. THE NON-MINIMUM PHASE ENVIRONMENT

A non-minimum phase environment is an environment which is itself stable, but its inverse is not. This occurs when the zeros in the *Z-domain* occur outside the unit circle. For example, consider the following *finite impulse response* (FIR) filter

$$1 + z^{-1} - 1.5z^{-2} + z^{-3} \tag{A.1}$$

This transfer function has the roots as defined in figure A.1a. This, being a FIR filter with no poles, is naturally stable. However, if it is inverted, the zeros become poles, therefore making instability possible. The inverted system is defined in the complex plane by figure A.1b. This system is not stable, as one of its poles lies at -2, which is outside the unit circle. In a practical situation, it has been claimed that non-minimum phase environments occur

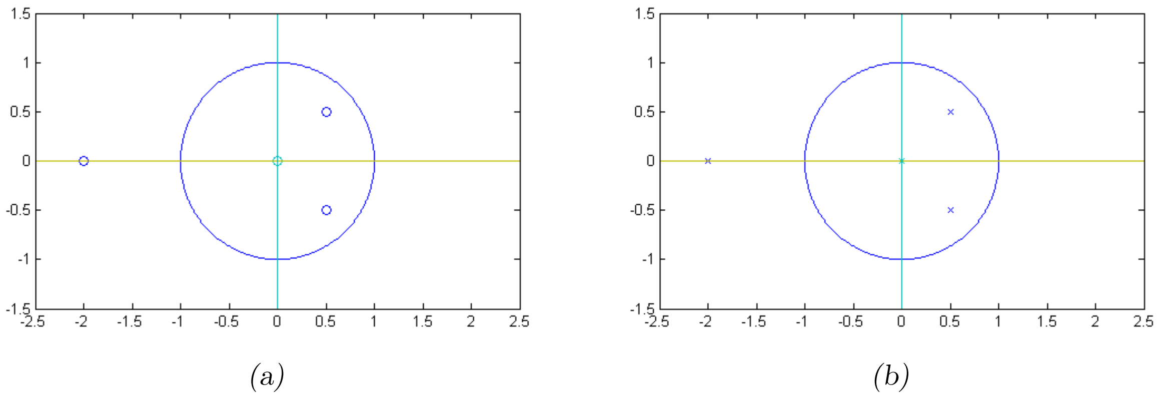


Fig. A.1: A fig:NonminimumPhaseSystem *non-minimum phase* system and fig:UnstableSystem its inverse.

when one of the reflections has a greater magnitude than the direct signal [45]. However, this is not the case, which can be shown by the following reasoning. Take for example, a simple first order FIR filter.

$$X_1 = 1 + a_1 z^{-1} \tag{A.2}$$

In order for this system to be minimum phase, $|a_1|$ must be less than 1, meaning that the direct signal is greater in amplitude than the reflection. However, if we then take a second order FIR filter,

$$\begin{aligned} X_2 &= \left(1 + a_1 z^{-1}\right)\left(1 + a_2 z^{-1}\right) \\ &= 1 + \left(a_1 + a_2\right) z^{-1} + a_1 a_2 z^{-2} \end{aligned} \tag{A.3}$$

The requirement for the system in (A.3) to be minimum phase is that the magnitude of both a_1 and a_2 are less than 1. While this may guarantee that the coefficient in front of z^{-2} is less than the direct signal, it does not guarantee that $a_1 + a_2$ is less than 1. Therefore we cannot state that the requirement for minimum phase systems is that the direct signal has the greatest amplitude. However, we can state that they will tend to be weighted toward earlier reflections over later reflections.

If there is a desired frequency response, there exists more than one filter if all that is desired is a specified *magnitude* response. Each of these filters will also have a phase response, which will not be identical. The minimum phase filter is the filter which has minimum group delay for the given magnitude response. For example, take the following first order system

$$\begin{aligned} H\left(s\right) &= s + a \\ H\left(j\omega\right) &= j\omega + a \end{aligned} \tag{A.4}$$

where a is an arbitrary positive value. We then find the frequency magnitude response of this system

$$\begin{aligned} H_{dB}\left(j\omega\right) &= 20\log_{10}\left|j\omega + a\right| \\ &= 20\log_{10}\left(\sqrt{\omega^2 + a^2}\right) \\ &= 10\log_{10}\left(1 + \frac{\omega^2}{a^2}\right) \end{aligned} \tag{A.5}$$

If we then take the similar, but non-minimum phase case of

$$H\left(s\right) = s - a \tag{A.6}$$

we find that we get an identical magnitude response. This shows how two unique filters can have the same magnitude response, yet one can have zeros in the left half s-plane, and one can have zeros in the right half of the s-plane. Similar proofs can be shown in the Z-domain.

If we now look at the phase response of the system in Equation (A.4), we get

$$\begin{aligned} \phi\left\{H\left(j\omega\right)\right\} &= \phi\left\{j\omega + a\right\} \\ &= \tan^{-1}\left(\frac{\omega}{a}\right) \end{aligned} \tag{A.7}$$

Therefore, as $\omega \to 0$, $\phi \to 0$, and as $\omega \to \infty$, $\phi \to \pi/2$. If we now consider the non-minimum phase case in Equation (A.6), we get

$$\begin{aligned}\phi\left\{H\left(j\omega\right)\right\}=&\phi\left\{j\omega-a\right\}\\=&\tan^{-1}\left(\frac{\omega}{-a}\right)\end{aligned}\tag{A.8}$$

In this case, as $\omega \to 0$, $\phi \to \pi$, and as $\omega \to \infty$, $\phi \to \pi/2$. From these two examples of the phase response, we can see that the system with its zero in the left half plane advances the phase within the range $0 \leq \phi < \pi/2$, whereas the system with its zero in the right half plane advances the phase by $\pi \leq \phi < \pi/2$.

If we then extend this to an nth order case, there will be four potential filters of the same magnitude response, in the form

$$\left(s \pm a_1\right)\left(s \pm a_2\right)\ldots\left(s \pm a_n\right).\tag{A.9}$$

The filter that results in minimal change of phase will be the filter

$$\left(s + a_1\right)\left(s + a_2\right)\ldots\left(s + a_n\right).\tag{A.10}$$

Therefore, systems with zeros only in the left half-plane will have the minimum group delay in phase across all frequencies for a given magnitude response. Hence the term non-minimum phase. In a practical application, this means that a minimum phase system is one that for a given frequency response has the weights as early as possible in its transfer function. This, however, does *not* mean that the first weight must the heaviest weight.

B. DERIVATIONS

B.1 MIMO SAD

B.1.1 Proof 1

In order to find the derivative of the cross-correlation with respect to each demixing tap, we must first rearrange the expression in (3.17) to be a function of the source powers. By using the identities in (3.14) and (3.15), we get

$$\begin{aligned}
C_{y_i y_j}(m) &= E\left[\sum_{k=0}^{N-1} \mathbf{x}_k^T(t)\,\mathbf{w}_{i,k}(t) \sum_{l=0}^{N-1} \mathbf{x}_l^T(t-m)\,\mathbf{w}_{j,l}(t-m)\right] \\
&= E\left[\sum_{k=0}^{N-1}\sum_{n=0}^{L-1} x_k(t-n)\,w_{i,k}^n(t) \sum_{l=0}^{N-1}\sum_{o=0}^{L-1} x_l(t-m-o)\,w_{j,l}^o(t-m)\right] \\
&= E\left[\left(\sum_{k=0}^{N-1}\sum_{n=0}^{L-1}\sum_{p=0}^{N-1} \mathbf{s}_p^T(t-n)\,\mathbf{h}_{k,p} w_{i,k}^n(t)\right)\right. \\
&\quad \left.\left(\sum_{l=0}^{N-1}\sum_{o=0}^{L-1}\sum_{q=0}^{N-1} \mathbf{s}_q^T(t-m-o)\,\mathbf{h}_{l,q} w_{j,l}^o(t-m)\right)\right] \\
&= E\left[\sum_{k=0}^{N-1}\sum_{n=0}^{L-1}\sum_{p=0}^{N-1}\sum_{l=0}^{N-1}\sum_{o=0}^{L-1}\sum_{q=0}^{L-1}\sum_{r=0}^{L-1}\sum_{u=0}^{L-1} s_p(t-n-r)\,h_{k,p}^r w_{i,k}^n(t)\right. \\
&\quad \left. s_q(t-m-o-u)\,h_{l,q}^u w_{j,l}^o(t-m)\right]
\end{aligned} \tag{B.1}$$

which is only nonzero when $p = q$ and $u = n + r - m - o$ assuming stationarity, whiteness, and no correlation between the sources. Thus (B.1) becomes

$$
\begin{aligned}
=&E\left[\sum_{k=0}^{N-1}\sum_{n=0}^{L-1}\sum_{p=0}^{N-1}\sum_{l=0}^{N-1}\sum_{o=0}^{L-1}\sum_{r=0}^{L-1} s_p\left(t-n-r\right)h_{k,p}^{r}w_{i,k}^{n}\left(t\right)\right.\\
&\left. s_p\left(t-n-r\right)h_{l,p}^{n+r-m-o}w_{j,l}^{o}\left(t-m\right)\right]\\
=&E\left[\sum_{k=0}^{N-1}\sum_{n=0}^{L-1}\sum_{p=0}^{N-1}\sum_{l=0}^{N-1}\sum_{o=0}^{L-1}\sum_{q=0}^{L-1} s_p\left(t-n-q\right)h_{k,p}^{q}w_{i,k}^{n}\left(t\right)\right.\\
&\left. s_p\left(t-n-q\right)h_{l,p}^{n+q-m-o}w_{j,l}^{o}\left(t-m\right)\right]\\
=&\sum_{p=0}^{N-1}\sigma_p^2\sum_{k=0}^{N-1}\sum_{n=0}^{L-1}\sum_{l=0}^{N-1}\sum_{o=0}^{L-1}\sum_{q=0}^{L-1} h_{k,p}^{q}w_{i,k}^{n}\left(t\right)h_{l,p}^{n+q-m-o}w_{j,l}^{o}\left(t-m\right) \qquad \text{(B.2)}
\end{aligned}
$$

which gives the result in equation (3.21).

B.1.2 Proof 2

Here we calculate the derivative of the cross-correlation with respect to the demixing filter taps. The cross-correlation in (3.21) has many terms that reduce to zero, due to the constraints on the mixing system.

When $p = k$,

$$
h_{k,p}^{q}=\begin{cases} 1, & q=0\\ 0, & q\neq 0\end{cases}
$$

and when $p \neq k$,

$$
h_{k,p}^{0}=0
$$

The equivalent taps in the demixing filters also reduce to zero.

If we go through all of the terms in (3.21), methodically removing all that reduce to zero because of the above constraints, it becomes

$$\begin{aligned}
C_{y_i,y_j}(m) = &\sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1} \sigma_p^2 \sum_{\substack{k=0\\k\neq p\\k\neq i}}^{N-1} \sum_{\substack{l=0\\l\neq p\\l\neq j}}^{N-1} \sum_{n=1}^{L-1} \sum_{o=1}^{L-1} \sum_{q=1}^{L-1} h_{k,p}^{q} w_{i,k}^{n}(t)\, h_{l,p}^{n+q-m-o} w_{j,l}^{o}(t-m) \\
&+ \sigma_i^2 \sum_{\substack{k=0\\k\neq i}}^{N-1} \sum_{\substack{l=0\\l\neq i\\l\neq j}}^{N-1} \sum_{n=1}^{L-1} \sum_{o=1}^{L-1} \sum_{q=1}^{L-1} h_{k,i}^{q} w_{i,k}^{n}(t)\, h_{l,i}^{n+q-m-o} w_{j,l}^{o}(t-m) \\
&+ \sigma_j^2 \sum_{\substack{k=0\\k\neq i\\k\neq j}}^{N-1} \sum_{\substack{l=0\\l\neq j}}^{N-1} \sum_{n=1}^{L-1} \sum_{o=1}^{L-1} \sum_{q=1}^{L-1} h_{k,j}^{q} w_{i,k}^{n}(t)\, h_{l,j}^{n+q-m-o} w_{j,l}^{o}(t-m) \\
&+ \sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1} \sigma_p^2 \sum_{\substack{k=0\\k\neq p\\k\neq i}}^{N-1} \sum_{n=1}^{L-1} \sum_{q=1}^{L-1} h_{k,p}^{q} w_{i,k}^{n}(t)\, h_{j,p}^{n+q-m} \\
&+ \sigma_i^2 \sum_{\substack{k=0\\k\neq i}}^{N-1} \sum_{n=1}^{L-1} \sum_{q=1}^{L-1} h_{k,i}^{q} w_{i,k}^{n}(t)\, h_{j,i}^{n+q-m} \\
&+ \sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1} \sigma_p^2 \sum_{\substack{l=0\\l\neq p\\l\neq j}}^{N-1} \sum_{o=1}^{L-1} \sum_{q=1}^{L-1} h_{i,p}^{q} h_{l,p}^{q-m-o} w_{j,l}^{o}(t-m) \\
&+ \sigma_j^2 \sum_{\substack{l=0\\l\neq j}}^{N-1} \sum_{o=1}^{L-1} \sum_{q=1}^{L-1} h_{i,j}^{q} h_{l,j}^{q-m-o} w_{j,l}^{o}(t-m) \\
&+ \sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1} \sigma_p^2 \sum_{q=1}^{L-1} h_{i,p}^{q} h_{j,p}^{q-m}
\end{aligned}$$

$$+\sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1}\sigma_p^2\sum_{\substack{k=0\\k\neq p\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}\sum_{o=1}^{L-1}h_{k,p}^{m+o-n}w_{i,k}^{n}\left(t\right)w_{j,p}^{o}\left(t-m\right)$$

$$+\sigma_i^2\sum_{\substack{k=0\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}\sum_{o=1}^{L-1}h_{k,i}^{m+o-n}w_{i,k}^{n}\left(t\right)w_{j,i}^{o}\left(t-m\right)$$

$$+\sigma_j^2\sum_{\substack{k=0\\k\neq j\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}h_{k,j}^{m-n}w_{i,k}^{n}\left(t\right)$$

$$+\sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1}\sigma_p^2\sum_{o=1}^{L-1}h_{i,p}^{m+o}w_{j,p}^{o}\left(t-m\right)$$

$$+\sigma_j^2h_{i,j}^{m}$$

$$+\sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1}\sigma_p^2\sum_{n=1}^{L-1}\sum_{\substack{l=0\\l\neq p\\l\neq j}}^{N-1}\sum_{o=1}^{L-1}w_{i,p}^{n}\left(t\right)h_{l,p}^{n-m-o}w_{j,l}^{o}\left(t-m\right)$$

$$+\sigma_j^2\sum_{n=1}^{L-1}\sum_{\substack{l=0\\l\neq j}}^{N-1}\sum_{o=1}^{L-1}w_{i,j}^{n}\left(t\right)h_{l,j}^{n-m-o}w_{j,l}^{o}\left(t-m\right)$$

$$+\sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1}\sigma_p^2\sum_{n=1}^{L-1}w_{i,p}^{n}\left(t\right)h_{j,p}^{n-m}$$

$$+\sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1}\sigma_p^2\sum_{n=1}^{L-1}w_{i,p}^{n}\left(t\right)w_{j,p}^{n-m}\left(t-m\right)$$

$$+\sigma_j^2w_{i,j}^{m}\left(t\right)$$

Looking through this, we can see that it can be factorized into the following.

$$
\begin{aligned}
=&\sum_{\substack{p=0\\p\neq i\\p\neq j}}^{N-1}\sigma_p^2\sum_{q=1}^{2L-2}\left(h_{i,p}^{q}+w_{i,p}^{q}(t)+\sum_{\substack{k=0\\k\neq p\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}h_{k,p}^{q-n}w_{i,k}^{n}(t)\right)\\
&\left(h_{j,p}^{q-m}+w_{j,p}^{q-m}(t-m)+\sum_{\substack{l=0\\l\neq p\\l\neq j}}^{N-1}\sum_{o=1}^{L-1}h_{l,p}^{q-m-o}w_{j,l}^{o}(t-m)\right)\\
&+\sigma_i^2\sum_{\substack{k=0\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}\sum_{q=1}^{2L-2}h_{k,i}^{q-n}w_{i,k}^{n}(t)\left(h_{j,i}^{q-m}+w_{j,i}^{q-m}(t-m)+\sum_{\substack{l=0\\l\neq i\\l\neq j}}^{N-1}\sum_{o=1}^{L-1}h_{l,i}^{q-m-o}w_{j,l}^{o}(t-m)\right)\\
&+\sigma_j^2\sum_{\substack{k=0\\k\neq j}}^{N-1}\sum_{n=1}^{L-1}\sum_{q=1}^{2L-2}h_{jhlgk}^{q-m-n}w_{j,k}^{n}(t-m)\left(h_{i,j}^{q}+w_{i,j}^{q}(t)+\sum_{\substack{l=0\\l\neq i\\l\neq j}}^{N-1}\sum_{o=1}^{L-1}h_{jhlgl}^{q-o}w_{i,l}^{o}(t)\right)\\
&+\sigma_j^2\left(h_{i,j}^{m}+w_{i,j}^{m}(t)+\sum_{\substack{k=0\\k\neq j\\k\neq i}}^{N-1}\sum_{n=1}^{L-1}h_{k,j}^{m-n}w_{i,k}^{n}(t)\right)
\end{aligned}
$$

which becomes equation (3.22). Note that the terms of the form

$$h+w+\sum\sum hw$$

are actually a measure of the deviation of a tap from its ideal value.

B.1.3 Proof 3

We will now find the derivative of equation (3.22) with respect to the separating filter tap $w_{i,j}^{m}$.

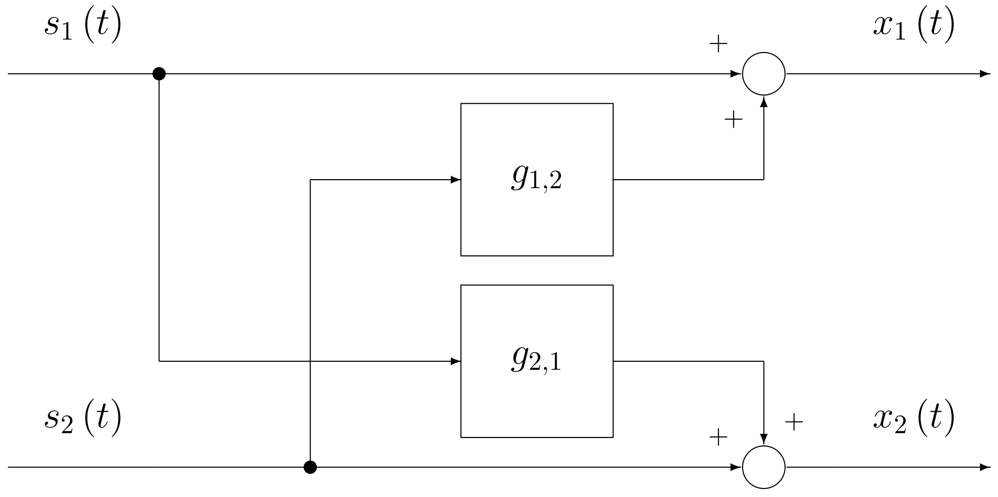


Fig. B.1: The mixing system.

$$
\begin{aligned}
&\frac{\partial C_{y_i,y_j}(m)}{\partial w_{i,j}^{m}} \\
=&\sum_{\substack{p=0\\p\neq j}}^{N-1}\sigma_p^2\sum_{q=1}^{2L-2}h_{j,p}^{q-m}\left(h_{j,p}^{q-m}+w_{j,p}^{q-m}(t-m)+\sum_{\substack{l=0\\l\neq p\\l\neq j}}^{N-1}\sum_{o=1}^{L-1}h_{l,p}^{q-m-o}w_{j,l}^{o}(t-m)\right)+\sigma_j^2 \\
=&\sum_{\substack{p=0\\p\neq j}}^{N-1}\sigma_p^2\sum_{q=1}^{L-1}h_{j,p}^{q}\left(h_{j,p}^{q}+w_{j,p}^{q}(t-m)+\sum_{\substack{l=0\\l\neq p\\l\neq j}}^{N-1}\sum_{o=1}^{L-1}h_{l,p}^{q-o}w_{j,l}^{o}(t-m)\right)+\sigma_j^2
\end{aligned}
$$

which equals (3.24).

B.2 TITO SAD

The formulae for one side of the *symmetric adaptive decorrelation* (SAD) algorithm are

$$y_1(t)=x_1(t)-\mathbf{w}_1^T(t)\,\mathbf{x}_2(t) \tag{B.3}$$

$$\mathbf{w}_1(t+1)=\mathbf{w}_1(t)+\mu y_1(t)\,\mathbf{y}_2(t) \tag{B.4}$$

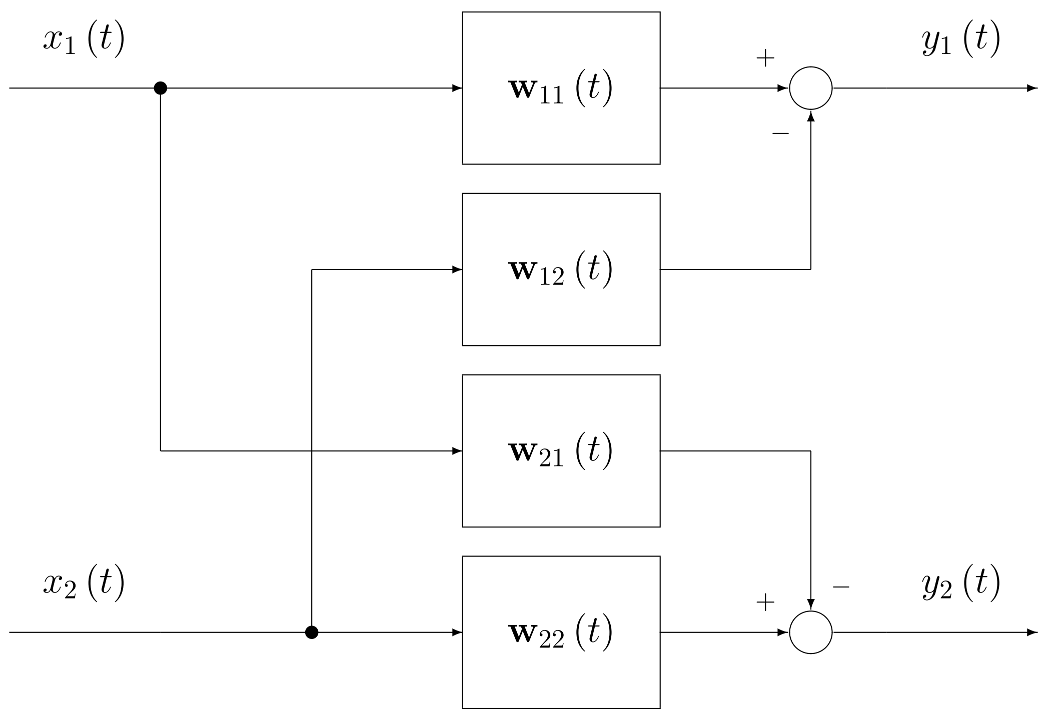


Fig. B.2: The demixing system.

where $\mathbf{w}_1$ is the vector of the filter weights,

$$\mathbf{x}_2(t) = \begin{bmatrix} x_2(t) & x(t-1) & \dots & x_2(t-L+1) \end{bmatrix}^T$$
$$\mathbf{y}_2(t) = \begin{bmatrix} y_2(t) & y(t-1) & \dots & y_2(t-L+1) \end{bmatrix}^T$$

Rewriting (B.4) as a block algorithm becomes

$$\begin{aligned} \mathbf{w}_1(kL+L) =& \mathbf{w}_1(kL) + \mu_1 \left[y_1(kL)\,\mathbf{y}_2(kL) + y_1(kL+1)\,\mathbf{y}_2(kL+1) \right. \\ & \left. + \dots + y_1(kL+L-1)\,\mathbf{y}_2(kL+L-1) \right] \\ =& \mathbf{w}_1(kL) + \mu_1 \boldsymbol{\nabla}_1(kL) \qquad \text{(B.5)} \\ \Rightarrow \text{FFT}\left[\mathbf{w}_1(kL+L)\right] =& \text{FFT}\begin{bmatrix} \mathbf{w}_1(kL) \\ \mathbf{0} \end{bmatrix} + \mu_1 \text{FFT}\begin{bmatrix} \boldsymbol{\nabla}_1(kL) \\ \mathbf{0} \end{bmatrix} \qquad \text{(B.6)} \end{aligned}$$

which can be rewritten as

$$\widehat{\mathbf{w}}_1\,(k+1) = \widehat{\mathbf{w}}_1\,(k) + \mu_1 \widehat{\boldsymbol{\nabla}}_1\,(k) \tag{B.7}$$

where the ˆ denotes the frequency domain implementation of the variables and are defined by the terms in (B.6).

The first thing that we will do is derive a frequency domain representation of the accumulation of update terms in (B.5), which can be rewritten as

$$\boldsymbol{\nabla}_1(kL)=\begin{bmatrix} \sum_{i=0}^{L-1} y_1(kL+i)\, y_2(kL+i) \\ \sum_{i=0}^{L-1} y_1(kL+i)\, y_2(kL+i-1) \\ \vdots \\ \sum_{i=0}^{L-1} y_1(kL+i)\, y_2(kL+i-L+1) \end{bmatrix} \tag{B.8}$$

If we now consider the convolution of the following two vectors

$$\begin{bmatrix} y_2(t+L-1) \\ y_2(t+L-2) \\ \vdots \\ y_2(t) \\ y_2(t-1) \\ y_2(t-2) \\ \vdots \\ y_2(t-L+2) \\ y_2(t-L+1) \end{bmatrix} * \begin{bmatrix} y_1(t) \\ y_1(t+1) \\ \vdots \\ y_1(t+L-2) \\ y_1(t+L-1) \end{bmatrix} \tag{B.9}$$

we get the following sequence

$$\begin{bmatrix} y_1(t)\, y(t+L-1) \\ \sum_{i=0}^{1} y_1(t+i)\, y_2(t+L-2+i) \\ \vdots \\ \sum_{i=0}^{L-2} y_1(t+i)\, y_2(t+1+i) \\ \sum_{i=0}^{L-1} y_1(t+i)\, y_2(t+i) \\ \sum_{i=0}^{L-1} y_1(t+i)\, y_2(t-1+i) \\ \vdots \\ \sum_{i=0}^{L-1} y_1(t+i)\, y_2(t-L+2+i) \\ \sum_{i=0}^{L-1} y_1(t+i)\, y_2(t-L+1+i) \\ \sum_{i=0}^{L-2} y_1(t+i+1)\, y_2(t-L+1+i) \\ \vdots \\ \sum_{i=0}^{1} y_1(t+L-2+i)\, y_2(t-L+1+i) \\ y_1(t+L-1)\, y_2(t-L+1) \end{bmatrix} \tag{B.10}$$

If we discard the first $L-1$ and the last $L-1$ terms of (B.10), we get (B.8).

When implementing convolution by using the frequency domain, it is actually circular convolution, so we can perform this circular convolution on the following vectors, and only take the first L terms.

$$\mathbf{v}_1 = \begin{bmatrix} y_2(t+L-1) \\ y_2(t+L-2) \\ \vdots \\ y_2(t-L) \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ y_1(t) \\ y_1(t+1) \\ \vdots \\ y_1(t+L-1) \end{bmatrix} \tag{B.11}$$

This can be written in the frequency domain as

$$\widehat{\mathbf{v}}_1 = \mathrm{FFT}\begin{bmatrix} \mathbf{y}_2(kL+L-1) \end{bmatrix} \circ \overline{\mathrm{FFT}\begin{bmatrix} \mathbf{y}_1(kL+L-1) \\ \mathbf{0} \end{bmatrix}} \tag{B.12}$$

where the overline denotes complex conjugation and maps to time reversal in the time domain. If we set

$$\begin{aligned} \widehat{\mathbf{y}}_1'(k) &= \mathrm{FFT}\begin{bmatrix} \mathbf{y}_1(kL+L-1) \\ \mathbf{0} \end{bmatrix} \\ \widehat{\mathbf{y}}_2(k) &= \mathrm{FFT}\begin{bmatrix} \mathbf{y}_2(kL+L-1) \\ \mathbf{y}_2(kL-1) \end{bmatrix} \end{aligned} \tag{B.13}$$

we can simplify (B.12) to

$$\widehat{\mathbf{v}}_1(k) = \overline{\widehat{\mathbf{y}}_1'(k)} \circ \widehat{\mathbf{y}}_2(k) \tag{B.14}$$

This completes the update equations for the separating filters.

Now we must derive equations for the filter *outputs*, y_1 and y_2.

$$\begin{aligned} y_1(t) &= x_1(t) - \mathbf{w}_1^T(t)\,\mathbf{x}_2(t) \\ \Rightarrow \mathbf{y}_1(t) &= \mathbf{x}_1(t) - \begin{bmatrix} \mathbf{w}_1^T(t)\,\mathbf{x}_2(t) \\ \mathbf{w}_1^T(t-1)\,\mathbf{x}_2(t-1) \\ \vdots \\ \mathbf{w}_1^T(t-L+1)\,\mathbf{x}_2(t-L+1) \end{bmatrix} \end{aligned} \tag{B.15}$$

Under the assumption that

$$\begin{aligned} \mathbf{w}_1(t) &= \mathbf{w}_1(t-1) \\ &= \mathbf{w}_1(t-2) \\ &\vdots \\ &= \mathbf{w}_1(t-L+1) \end{aligned} \tag{B.16}$$

we can then simplify (B.15) to

$$\mathbf{y}_1(t) = \mathbf{x}_1(t) - \mathbf{u}_1(t) \tag{B.17}$$

where

$$\mathbf{u}_1(t) = \begin{bmatrix} \mathbf{w}_1^T(t)\,\mathbf{x}_2(t) \\ \mathbf{w}_1^T(t)\,\mathbf{x}_2(t-1) \\ \vdots \\ \mathbf{w}_1^T(t)\,\mathbf{x}_2(t-L+1) \end{bmatrix} \tag{B.18}$$

This can be obtained by implementing the following circular convolution

$$\begin{bmatrix} \mathbf{w}_1(t) \\ \mathbf{0} \end{bmatrix} * \begin{bmatrix} x_2(t-2L+1) \\ x_2(t-2L+2) \\ \vdots \\ x_2(t-L) \\ x_2(t-L+1) \\ x_2(t-L+2) \\ \vdots \\ x_2(t-1) \\ x_2(t) \end{bmatrix} \tag{B.19}$$

$$= \begin{bmatrix} \vdots \\ \text{To Discard} \\ \vdots \\ \mathbf{w}_1^T(t)\,\mathbf{x}_2(t-L+1) \\ \mathbf{w}_1^T(t)\,\mathbf{x}_2(t-L+2) \\ \vdots \\ \mathbf{w}_1^T(t)\,\mathbf{x}_2(t-1) \\ \mathbf{w}_1^T(t+L-1)\,\mathbf{x}_2(t) \end{bmatrix} \tag{B.20}$$

of which the last L terms are the time-reversed equivalent of the filtered signal given in (B.18).

In the frequency domain, the circular convolution result in (B.20) can be gained by the following calculation

$$\widehat{\mathbf{w}}_1(k) \circ \overline{\widehat{\mathbf{x}}_2(k)} \tag{B.21}$$

where

$$\begin{aligned}\widehat{\mathbf{w}}_1(k) &= \mathrm{FFT}\begin{bmatrix}\mathbf{w}(kL)\\ \mathbf{0}\end{bmatrix}\\ \widehat{\mathbf{x}}_2(k) &= \mathrm{FFT}\begin{bmatrix}\mathbf{x}_2(kL)\\ \mathbf{x}_2(kL-L)\end{bmatrix}\end{aligned} \tag{B.22}$$

But time reversal in the time domain maps to complex convolution in the frequency domain, therefore

$$\mathbf{u}_1(t) = \text{First } L \text{ terms of} \left\{\mathrm{FFT}^{-1}\left[\overline{\widehat{\mathbf{w}}_1(k)} \circ \widehat{\mathbf{x}}_2(k)\right]\right\} \tag{B.23}$$

B.3 FD-SAMLE

The update equations for the time domain algorithm are given by

$$w_{i,j}^m(t+1) = w_{i,j}^m(t) - \mu \frac{\operatorname{sgn}(y_i(t))\, y_j(t-m)}{E\{|y_i(t)|\}} \tag{B.24}$$

This can be rewritted for the update of the entire vectors.

$$\mathbf{w}_{i,j}(t+1) = \mathbf{w}_{i,j}(t) - \mu \frac{\operatorname{sgn}(y_i(t))\, \mathbf{y}_j(t)}{E\{|y_i(t)|\}} \tag{B.25}$$

where the vector $\mathbf{y}_i(t)$ is defined as

$$\mathbf{y}_i(t) \triangleq \begin{bmatrix} y_i(t) & y_i(t-1) & \ldots & y_i(t-L+1)\end{bmatrix}^T$$

In order to convert (B.25) into a frequency domain function, we must first implement it as a block algorithm.

$$\begin{aligned}\mathbf{w}_{i,j}\left(kL+L\right)=&\mathbf{w}_{i,j}\left(kL\right)-\frac{1}{E\left\{\left|y_i\left(t\right)\right|\right\}}\left[\operatorname{sgn}\left(y_i\left(kL\right)\right)\mathbf{y}_j\left(kL\right)\right.\\&+\operatorname{sgn}\left(y_i\left(kL+1\right)\right)\mathbf{y}_j\left(kL+1\right)+\ldots+\\&\left.+\operatorname{sgn}\left(y_i\left(kL+L-1\right)\right)\mathbf{y}_j\left(kL+L-1\right)\right]\\=&\mathbf{w}_{i,j}\left(kL\right)-\frac{\boldsymbol{\nabla}_{i,j}\left(kL\right)}{\sigma^2}\end{aligned}\tag{B.26}$$

where $\boldsymbol{\nabla}_{i,j}\left(kL\right)$ is self-defined by this equation.

For the conversion to the frequency domain, we take the fast Fourier transform (FFT) of both sides, giving

$$\mathrm{FFT}\begin{bmatrix}\mathbf{w}_{i,j}\left(kL+L\right)\\\mathbf{0}\end{bmatrix}=\mathrm{FFT}\begin{bmatrix}\mathbf{w}_{i,j}\left(kL\right)\\\mathbf{0}\end{bmatrix}-\frac{\mathrm{FFT}\begin{bmatrix}\boldsymbol{\nabla}_{i,j}\left(kL\right)\\\mathbf{0}\end{bmatrix}}{\sigma^2}\tag{B.27}$$

which can be rewritten as

$$\widehat{\mathbf{w}}_{i,j}\left(k+1\right)=\widehat{\mathbf{w}}_{i,j}\left(k\right)+\frac{\widehat{\boldsymbol{\nabla}}_{i,j}\left(k\right)}{\sigma^2}\tag{B.28}$$

where the $\widehat{}$ denotes the frequency domain implementation of the variables as defined by (B.27).

We now consider the time domain block update term $\boldsymbol{\nabla}_{i,j}\left(kL\right)$. This is defined by the vector

$$\boldsymbol{\nabla}_{i,j}\left(kL\right)=\begin{bmatrix}\sum_{l=0}^{L-1}\operatorname{sgn}\left(y_i\left(kL+l\right)\right)y_j\left(kL+l\right)\\ \sum_{l=0}^{L-1}\operatorname{sgn}\left(y_i\left(kL+l\right)\right)y_j\left(kL+l-1\right)\\ \vdots\\ \sum_{l=0}^{L-1}\operatorname{sgn}\left(y_i\left(kL+l\right)\right)y_j\left(kL+l-L+1\right)\end{bmatrix}\tag{B.29}$$

This is equivalent to the first L terms of the circular convolution operation

$$\mathbf{v}_{i,j}\left(k\right)=\begin{bmatrix}y_j\left(kL+L-1\right)\\ y_j\left(kL+L-2\right)\\ \vdots\\ y_j\left(kL-L\right)\end{bmatrix}*\begin{bmatrix}0\\ 0\\ \vdots\\ 0\\ \operatorname{sgn}\left(y_i\left(kL\right)\right)\\ \operatorname{sgn}\left(y_i\left(kL+1\right)\right)\\ \vdots\\ \operatorname{sgn}\left(y_i\left(kL+L-1\right)\right)\end{bmatrix}\tag{B.30}$$

The equivalent frequency domain operation simplifies to

$$\widehat{\mathbf{v}}_{i,j}\left(k\right)=\mathrm{FFT}\begin{bmatrix}\mathbf{y}_j\left(kL+L-1\right)\\ \mathbf{y}_j\left(kL-1\right)\end{bmatrix}\circ\overline{\mathrm{FFT}\begin{bmatrix}\operatorname{sgn}\left(\mathbf{y}_i\left(kL+L-1\right)\right)\\ \mathbf{0}\end{bmatrix}}\tag{B.31}$$

where the overline denotes complex conjugation and maps to time reversal in the time domain. If we set

$$\widehat{\mathbf{y}}'_i(k) = \mathrm{FFT}\begin{bmatrix} \mathrm{sgn}\left(\mathbf{y}_i(kL+L-1)\right) \\ \mathbf{0} \end{bmatrix}$$
$$\widehat{\mathbf{y}}_i(k) = \mathrm{FFT}\begin{bmatrix} \mathbf{y}_i(kL+L-1) \\ \mathbf{y}_i(kL-1) \end{bmatrix}$$

then (B.31) becomes

$$\widehat{\mathbf{v}}_{i,j}(k) = \overline{\widehat{\mathbf{y}}'_i(k)} \circ \widehat{\mathbf{y}}_j(k) \tag{B.32}$$

A similar route can also be taken for the filter outputs $y_i(kL)$.

$$\begin{aligned} y_i(kL) &= \sum_{l=0}^{N-1} \mathbf{x}_l^T(kL)\,\mathbf{w}_{l,i}(kL) \\ \Rightarrow \mathbf{y}_i(kL) &= \begin{bmatrix} \sum_{l=0}^{N-1} \mathbf{x}_l^T(kL)\,\mathbf{w}_{l,i}(kL) \\ \sum_{l=0}^{N-1} \mathbf{x}_l^T(kL-1)\,\mathbf{w}_{l,i}(kL-1) \\ \vdots \\ \sum_{l=0}^{N-1} \mathbf{x}_l^T(kL-L+1)\,\mathbf{w}_{l,i}(kL-L+1) \end{bmatrix} \end{aligned} \tag{B.33}$$

Under the assumption that

$$\begin{aligned} \mathbf{w}_{i,j}(kL) &= \mathbf{w}_{i,j}(kL-1) \\ &= \mathbf{w}i,j(kL-2) \\ &\vdots \\ &= \mathbf{w}_{i,j}(kL-L+1) \end{aligned} \tag{B.34}$$

we can simplify (B.33) to

$$\mathbf{y}_i(kL) = \begin{bmatrix} \sum_{l=0}^{N-1} \mathbf{x}_l^T(kL)\,\mathbf{w}_{l,i}(kL) \\ \sum_{l=0}^{N-1} \mathbf{x}_l^T(kL-1)\,\mathbf{w}_{l,i}(kL) \\ \vdots \\ \sum_{l=0}^{N-1} \mathbf{x}_l^T(kL-L+1)\,\mathbf{w}_{l,i}(kL) \end{bmatrix} = \sum_{l=0}^{N-1} \begin{bmatrix} \mathbf{x}_l^T(kL) \\ \mathbf{x}_l^T(kL-1) \\ \vdots \\ \mathbf{x}_l^T(kL-L+1) \end{bmatrix} \mathbf{w}_{l,i}(kL) \tag{B.35}$$

This can be obtained using the following circular convolution

$$\sum_{l=0}^{N-1} \begin{bmatrix} \mathbf{w}_{l,i}(kL) \\ \mathbf{0} \end{bmatrix} * \begin{bmatrix} x_l(kL-2L+1) \\ x_l(kL-2L+2) \\ \vdots \\ x_l(kL-1) \\ x_l(kL) \end{bmatrix} = \sum_{l=0}^{N-1} \begin{bmatrix} \vdots \\ \text{To Discard} \\ \vdots \\ \mathbf{x}_l^T(kL-L+1)\,\mathbf{w}_{l,i}(kL) \\ \mathbf{x}_l^T(kL-L+2)\,\mathbf{w}_{l,i}(kL) \\ \vdots \\ \mathbf{x}_l^T(kL-1)\,\mathbf{w}_{l,i}(kL) \\ \mathbf{x}_l^T(kL)\,\mathbf{w}_{l,i}(kL) \end{bmatrix} \tag{B.36}$$

of which the last L terms are the time-reversed equivalent of the filtered signal given in (B.35).

In the frequency domain, (B.36) can be obtained using the following point-by-point multiplication

$$\sum_{l=0}^{N-1} \widehat{\mathbf{w}}_{l,i}(k) \circ \overline{\widehat{\mathbf{x}}_l(k)} \tag{B.37}$$

where

$$\begin{aligned} \widehat{\mathbf{w}}_{l,i}(k) &= \text{FFT} \begin{bmatrix} \mathbf{w}_{l,i}(kL) \\ \mathbf{0} \end{bmatrix} \\ \widehat{\mathbf{x}}_l(k) &= \text{FFT} \begin{bmatrix} \mathbf{x}_l(kL) \\ \mathbf{x}_l(kL-L) \end{bmatrix} \end{aligned} \tag{B.38}$$

But reversal in the time domain maps to complex convolution in the frequency domain, therefore

$$\mathbf{y}_i(kL) = \text{First } L \text{ terms of } \left\{ \text{FFT}^{-1} \left[\sum_{l=0}^{N-1} \overline{\widehat{\mathbf{w}}_{l,i}}(k) \circ \widehat{\mathbf{x}}_l(k) \right] \right\} \tag{B.39}$$

B.4 Hybrid SAD

The update equations for the demixing filters of both the feedforward and the feedback SAD algorithm are given as

$$w_1^m(t+1) = w_1^m(t) + \mu y_1(t) y_2(t-m) \tag{B.40}$$

where $w_1^m(t)$ is the m^{th} tap of demixing filter 1 at time t, μ is an adaptive step size, and $y_1(t)$ is the first output at time t. This can also be written in vector form as

$$\mathbf{w}_1(t+1) = \mathbf{w}_1(t) + \mu y_1(t)\,\mathbf{y}_2(t) \tag{B.41}$$

where

$$\begin{aligned}
\mathbf{w}_1(t) &= \begin{bmatrix} w_1^0(t) & w_1^1(t) & \ldots & w_1^{L-1} \end{bmatrix}^T, \\
\mathbf{y}_1(t) &= \begin{bmatrix} y_1(t) & y_1(t-1) & \ldots & y_1(t-L+1) \end{bmatrix}^T
\end{aligned}$$

The outputs of the demixing system can then be given by

$$\begin{aligned}
y_1(t) &= x_1(t) + \mathbf{w}_1^T(t)\,y_2(t) \\
&= x_1(t) + \sum_{i=0}^{L-1} w_1^i(t)\,y_2(t-i) \\
\Rightarrow \mathbf{y}_1(t) &= \mathbf{x}_1(t) + \sum_{i=0}^{L-1} \begin{bmatrix} w_1^i(t) \\ w_1^i(t-1) \\ \vdots \\ w_1^i(t-L+1) \end{bmatrix} \circ \mathbf{y}_2(t-i)
\end{aligned}$$

where $x_1(t)$ is the *observed signal* regressor vector at time t, $\circ$ denotes element-wise multiplication or the *Hadamard product.*

Assuming demixing filter stationarity over a period of L updates, we get

$$\mathbf{y}_1(t) = \mathbf{x}_1(t) + \sum_{i=0}^{L-1} w_1^i(t)\,\mathbf{y}_2(t-i)$$

But with $w_1^0(t) = 0$,

$$\mathbf{y}_1(t) = \mathbf{x}_1(t) + \sum_{i=1}^{L-1} w_1^i(t)\,\mathbf{y}_2(t-i)$$

and likewise for y_2,

$$\mathbf{y}_2(t) = \mathbf{x}_2(t) + \sum_{i=1}^{L-1} w_2^i(t)\,\mathbf{y}_1(t-i)$$

Writing this in terms of the observed signals results in

$$\begin{aligned}
\mathbf{y}_1(t) =& \mathbf{x}_1(t) + \sum_{i=1}^{L-1} w_1^i(t) \Bigg\{ \\
& \mathbf{x}_2(t-i) + \sum_{j=1}^{L-1} w_2^j(t-i) \Bigg\{ \\
& \mathbf{x}_1(t-i-j) + \sum_{k=1}^{L-1} w_1^k(t-i-j) \Bigg\{ \\
& \ldots \Bigg\}\Bigg\}\Bigg\} \\
=& \mathbf{x}_1(t) \\
& + \sum_{i=1}^{L-1} w_1^i(t)\,\mathbf{x}_2(t-i) \\
& + \sum_{i=1}^{L-1}\sum_{j=1}^{L-1} w_1^i(t)\,w_2^j(t-i)\,\mathbf{x}_1(t-i-j) \\
& + \sum_{i=1}^{L-1}\sum_{j=1}^{L-1}\sum_{k=1}^{L-1} w_1^i(t)\,w_2^j(t-i)\,w_1^k(t-i-j)\,\mathbf{x}_2(t-i-j-k) \\
& + \ldots
\end{aligned}$$

Assuming that the observed signals are stationary, and uncorrelated to the demixing filter taps,

$$\begin{aligned}\mathbf{y}_1(t) &= \begin{bmatrix} y_1(t) \\ y_1(t-1) \\ \vdots \\ y_1(t-L+1) \end{bmatrix} \\ &= \begin{bmatrix} y_1(t) \\ y_1(t) \\ \vdots \\ y_1(t) \end{bmatrix}\end{aligned}$$

Therefore, we are only interested in the scalar $y_1(t)$ rather than the output regressor vectors.

$$
\begin{aligned}
E\left[y_1(t)\right] = & E\left[x_1(t)\right] \\
& + E\left[\sum_{i=1}^{L-1} w_1^i(t)\, x_2(t)\right] \\
& + E\left[\sum_{i=1}^{L-1}\sum_{j=1}^{L-1} w_1^i(t)\, w_2^j(t-i)\, x_1(t)\right] \\
& + E\left[\sum_{i=1}^{L-1}\sum_{j=1}^{L-1}\sum_{k=1}^{L-1} w_1^i(t)\, w_2^j(t-i)\, w_1^k(t-i-j)\, x_2(t)\right] \\
& + \ldots \\
= & E\left[x_1(t)\right] \\
& + E\left[\sum_{i=1}^{L-1} w_1^i(t)\right] E\left[x_2(t)\right] \\
& + E\left[\sum_{i=1}^{L-1} w_1^i(t)\right] E\left[\sum_{j=1}^{L-1} w_2^j(t-i)\right] E\left[x_1(t)\right] \\
& + E\left[\sum_{i=1}^{L-1} w_1^i(t) \sum_{k=1}^{L-1} w_1^k(t-i-j)\right] E\left[\sum_{j=1}^{L-1} w_2^j(t-i)\right] E\left[x_2(t)\right] \\
& + \ldots
\end{aligned}
\tag{B.42}
$$

For the feedback structure, there are two forms of stability that are of interest. The first is the convergence of the filter weights, which is also in common with the feedforward case, but there is also whether the demixing system, once converged, is stable due to the feedback structure. We will presently look at the latter case.

Assuming that the weights have converged, (B.42) becomes

$$\begin{aligned}E\left[y_1\left(t\right)\right]=&E\left[x_1\left(t\right)\right]\\&+\sum_{i=1}^{L-1} w_1^i\left(t\right)E\left[x_2\left(t\right)\right]\\&+\sum_{i=1}^{L-1} w_1^i\left(t\right)\sum_{j=1}^{L-1} w_2^j\left(t\right)E\left[x_1\left(t\right)\right]\\&+\sum_{i=1}^{L-1} w_1^i\left(t\right)\sum_{j=1}^{L-1} w_2^j\left(t\right)\sum_{k=1}^{L-1} w_1^k\left(t\right)E\left[x_2\left(t\right)\right]\\&+\ldots\end{aligned} \tag{B.43}$$

Letting

$$\alpha\left(t\right)=\sum_{i=1}^{L-1} w_1^i\left(t\right)\sum_{j=1}^{L-1} w_2^j\left(t\right)$$

means that (B.43) becomes

$$\begin{aligned}E\left[y_1\left(t\right)\right]=&E\left[x_1\left(t\right)\right]\left\{1+\alpha\left(t\right)+\alpha^2\left(t\right)+\ldots\right\}\\&+E\left[x_2\right]\sum_{i=1}^{L-1} w_1^i\left(t\right)\left\{1+\alpha\left(t\right)+\alpha^2\left(t\right)+\ldots\right\}\end{aligned} \tag{B.44}$$

Which will be stable in the mean if

$$\left|\alpha\left(t\right)\right|<1 \tag{B.45}$$

To ensure complete stability

$$E\left[y_1^2\left(t\right)\right] \tag{B.46}$$

must also converge as $t\rightarrow\infty$. Note that again this is investigating the stability of the converged demixing system, not of the filter convergence itself.

$$
\begin{aligned}
E\left[y_2^2(t)\right] =& E\Bigg[\Bigg\{x_1(t) \\
&+\sum_{i=1}^{L-1} w_1^i(t)\,x_2(t-i) \\
&+\sum_{i=1}^{L-1}\sum_{j=1}^{L-1} w_1^i(t)\,w_2^j(t-i)\,x_1(t-i-j) \\
&+\sum_{i=1}^{L-1}\sum_{j=1}^{L-1}\sum_{k=1}^{L-1} w_1^i(t)\,w_2^j(t-i)\,w_1^k(t-i-j)\,x_2(t-i-j-k) \\
&+\ldots\Bigg\}\Bigg\{x_1(t) \\
&+\sum_{i=1}^{L-1} w_1^i(t)\,x_2(t-i) \\
&+\sum_{i=1}^{L-1}\sum_{j=1}^{L-1} w_1^i(t)\,w_2^j(t-i)\,x_1(t-i-j) \\
&+\sum_{i=1}^{L-1}\sum_{j=1}^{L-1}\sum_{k=1}^{L-1} w_1^i(t)\,w_2^j(t-i)\,w_1^k(t-i-j)\,x_2(t-i-j-k) \\
&+\ldots\Bigg\}\Bigg] \\
=& E\Bigg[x_1^2(t) \\
&+2x_1(t)
\end{aligned} \tag{B.47}
$$

B.4.1 Hybrid SAD Pseudocode

Program 4 Hybrid SAD Pseudocode

```
$\circ$ denotes entry-wise multiplication.
The overline denotes complex conjugation.
Superscript $^T$ denotes the transpose operation.
Any other vector superscript denotes the vector length.
All vectors are length $K$ unless otherwise defined.
$n$                                  ▷ the number of observed signal samples
$N$                                  ▷ the number of observed signals
$K$                                  ▷ the minimum block size for frequency-domain convolution
$L$                                  ▷ the filter size
$x_1, x_2, \ldots, x_N$              ▷ the observed signals
$y_1, y_2, \ldots, y_N$              ▷ the outputs
$\mathbf{w}_{i,j,k}$                 ▷ the vectors containing the filter weights for sub-block index $k$

for $t = 0 \rightarrow n$ do
    $k = \frac{t}{K}$
    for $i = 1 \rightarrow N$ do
        $\mathbf{x}_i(t) =$ the next $K$ samples of $x_i$ in reverse order.
    end for

    for $i = 1 \rightarrow N$ do
        $y_i(t) = 0$
        for $j = 1 \rightarrow N, j \neq i$ do
            $y_i(t) += \mathbf{w}_{i,j,0}^T(t)\,\mathbf{x}_j(t)$
        end for
```

Program 5 The Hybrid SAD Pseudocode (Part 2)

```
    for $l = 0 \to \log_2 \frac{L}{2K}$ do
        if $\frac{k}{2^l} \in \mathbb{Z}$ then
            $\mathbf{tmp}^{2^{(l+1)}K} = \mathbf{0}^{2^{(l+1)}K}$
            for $j = 1 \to N, j \neq i$ do
                $\mathbf{tmp}^{2^{(l+1)}K} += \mathrm{FFT}\begin{bmatrix} \mathbf{w}_{i,j,2^l}(t) \\ \mathbf{w}_{i,j,2^l+1}(t) \\ \vdots \\ \mathbf{w}_{i,j,2^{l+1}-1}(t) \\ \mathbf{0}^{2^l K} \end{bmatrix} \circ \mathrm{FFT}\begin{bmatrix} \mathbf{x}_i(t) \\ \mathbf{x}_i(t-K) \\ \vdots \\ \mathbf{x}_i\left(t-2^{l+1}K\right) \end{bmatrix}$
            end for

            $\begin{bmatrix} \mathbf{y}_i\left(t+2^l K\right) \\ \mathbf{y}_i\left(t+2^l K+1\right) \\ \vdots \\ \mathbf{y}_i\left(t+2^{l+1}K-1\right) \\ \mathbf{discard}^{2^l K} \end{bmatrix} += \mathrm{FFT}^{-1}\left[\mathbf{tmp}^{2^{l+1}K}\right]$
        end if
    end for

    $y(t) += \mathrm{mod}(n,k)^{\mathrm{th}}$ element of $\mathbf{y}(t - \mathrm{mod}(n,k))$
end for
```

Program 6 The Hybrid SAD Pseudocode (Part 3)

```
  $k = \frac{t}{L}$
  **if** $k \in \mathbb{Z}$ **then**
    **for** $i = 1 \to N$ **do**
      $\mathbf{Y}_i^{2L}(t) = \text{FFT}\left[\mathbf{y}_i^{2L}(t)\right]$
      $\mathbf{Y}_i'^{2L}(t) = \text{FFT}\begin{bmatrix} \mathbf{y}_i^L(t) \\ \mathbf{0} \end{bmatrix}$
    **end for**
    **for** $i = 1 \to N$ **do**

      **for** $1 = 1 \to N, i \neq j$ **do**
        $\nabla \mathbf{W}_{i,j}^{2L}(t) = \overline{\mathbf{Y}_i'^{2L}(t)} \circ \mathbf{Y}_j^{2L}(t)$
        $\mathbf{W}_{i,j}^{2L}(t+1) = \mathbf{W}_{i,j}^{2L}(t) + \mu \nabla \mathbf{W}_{i,j}^{2L}(t)$
        $\begin{bmatrix} \mathbf{w}_{i,j,1}(t) \\ \mathbf{w}_{i,j,2}(t) \\ \vdots \\ \mathbf{w}_{i,j,\frac{L}{k}}(t) \\ \mathbf{v}^L \end{bmatrix} = \text{FFT}^{-1}\left[\mathbf{W}_{i,j}^{2L}(t)\right]$
        $\mathbf{W}_{i,j}^{2L}(t) = \text{FFT}\begin{bmatrix} \mathbf{w}_{i,j,1}(t) \\ \mathbf{w}_{i,j,2}(t) \\ \vdots \\ \mathbf{w}_{i,j,\frac{L}{k}}(t) \\ \mathbf{0}^L \end{bmatrix}$
      **end for**
    **end for**
  **end if**
**end for**
```

C. COMPUTATIONAL REQUIREMENTS

C.1 TD SAD

The update equations for the *multiple-input multiple-output* (MIMO) *time-domain symmetric adaptive decorrelation* (TD-SAD) algorithm are according to the following

$$\mathbf{w}_{i,j}\left(t+1\right)=\mathbf{w}_{i,j}\left(t\right)-\mu y_i\left(t\right)\mathbf{y}_j\left(t\right) \tag{C.1}$$

which requires $L+1$ multiplications, where L is the length of the demixing filter. For every *source*, $N-1$ filter updates are needed, which results in

$$\left(L+1\right)\left(N-1\right) \qquad \text{multiplies}$$

The filtering equations are as follows

$$y_i\left(t\right)=x_i\left(t\right)+\sum_{j=0 j\neq i}^{N-1}\mathbf{x}_j^T\left(t\right)\mathbf{w}_{i,j}\left(t\right) \tag{C.2}$$

which requires

$$\left(N-1\right)L \qquad \text{multiplies}$$

per source, bringing the total computational requirements of the entire TD-SAD algorithm (all sources) to

$$N\left(N-1\right)\left(2L+1\right) \qquad \text{multiplies}$$

However, because we are comparing this to algorithms that use block-updating, it is more convenient to specify the number of multiplies per L updates. This becomes

$$\begin{aligned} & NL\left(N-1\right)\left(2L+1\right) \\ =& \left(N^2L-NL\right)\left(2L+1\right) \\ =& 2N^2L^2+N^2L-NL^2-NL \qquad \text{multiplies} \end{aligned} \tag{C.3}$$

C.2 FD SAD

For implementation in the frequency domain, the FFT algorithm will be utilized. According to Bergland in [102], for real time-domain signals, a length $2L$ FFT requires

$$2L\log_2 L-5L+6 \quad \text{real multiplies}$$

Considering the algorithm specified in Program 3, there are 2 FFTs per source for the filtering, requiring

$$4L\log_2 L-10L+12 \quad \text{real multiplies.}$$

Because the length of all of the frequency-domain vectors is $2L$, aside from the FFTs, there are also

$$2L\left(N-1\right) \qquad \text{complex multiplies}$$

for the frequency-domain convolution.

For every complex multiplication, four real multiplies are needed. This brings the total number of multiplies for filtering the signals to

$$\begin{aligned}
& 4L\log_2 L - 10L + 12 + 8L\left(N-1\right) \\
=& 4L\log_2 L - 10L + 12 + 8LN - 8L \\
=& 8LN + 4L\log_2 L - 18L + 12 \qquad \text{real multiplies}
\end{aligned} \tag{C.4}$$

For the update, 2 FFTs are needed per source to find the output correlations. An additional $2\left(N-1\right)$ FFTs are needed to zero-pad the frequency-domain filters. This requires a total of

$$\begin{aligned}
& \left(2 + 2\left(N-1\right)\right)\left(2L\log_2 L - 5L + 6\right) \\
=& 2N\left(2L\log_2 L - 5L + 6\right) \\
=& 4NL\log_2 L - 10NL + 12N \qquad \text{real multiplies}
\end{aligned}$$

Aside from the multiplications required for the FFTs, there are also $2L\left(N-1\right)$ complex multiplications required for the frequency-domain correlation and L real multiplications for the adaptive step size, requiring

$$\begin{aligned}
& 8L\left(N-1\right) + L \\
=& 8NL - 7L \qquad \text{real multiplies}
\end{aligned}$$

bringing the total number of multiplies for the update to

$$\begin{aligned}&4NL\log_2 L - 10NL + 12N + 8NL - 7L\\ =&4NL\log_2 L - 2NL + 12N - 7L \qquad \text{real multiplies}\end{aligned} \tag{C.5}$$

(C.4) and (C.5) can then be summed and multiplied by the number of sources to find the total requirement for the *frequency-domain symmetric adaptive decorrelation* (FD-SAD) algorithm.

$$\begin{aligned}&N(8LN + 4L\log_2 L - 18L + 12 + 4NL\log_2 L - 2NL + 12N - 7L)\\ =&4LN^2\log_2 L + 6LN^2 + 4LN\log_2 L - 25LN + 12N^2 + 12N \qquad \text{real multiplies}\end{aligned} \tag{C.6}$$

C.3 Hybrid SAD

For the Hybrid SAD algorithm, the update is identical to the update of the FD-SAD algorithm, so requires

$$4NL\log_2 L - 2NL + 12N - 7L \qquad \text{real multiplies} \tag{C.7}$$

per source.

To find out the computational requirements of the filtering portion of the algorithm is a much more complex procedure. First we introduce a constant K, which is the sub-block size at which convolution becomes more efficient in the frequency domain. K changes depending on how many sources need to be separated, but tends to be 8, 16 or 32. We make the

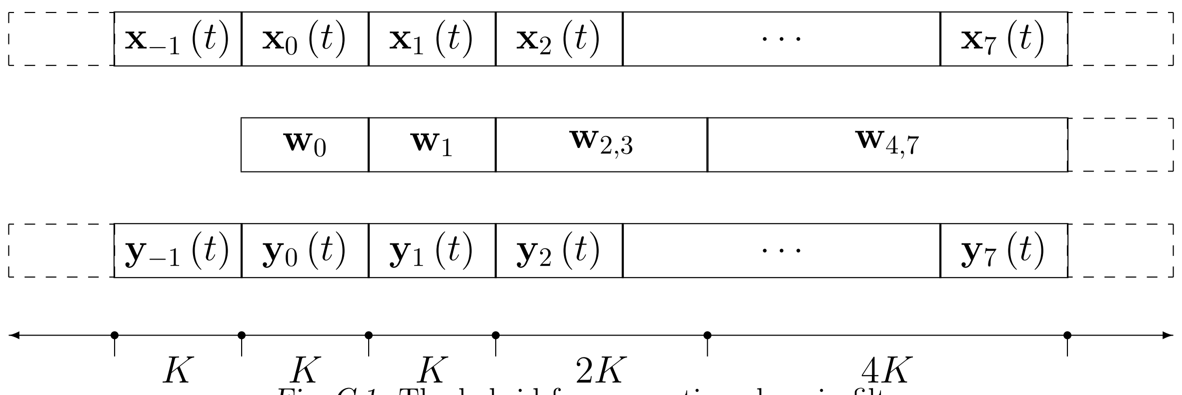


Fig. C.1: The hybrid frequency-time domain filter.

following assumptions about K and the filter size L

$$K{=}2^i, \qquad i = 0, 1, 2, \ldots$$
$$L{=}2^j, \qquad j = 1, 2, 3, \ldots$$
$$K{\leq}\frac{L}{2}$$

Considering Fig. C.1, the output $y(t)$ can be found by the following

$$\mathbf{y}_0{=}\mathbf{w}_0 * \mathbf{x}_{-1,0}(t) + \mathbf{w}_1 * \mathbf{x}_{-2,-1}(t) + \mathbf{w}_{2,3} * \mathbf{x}_{-4,-2} + \ldots + \mathbf{w}_{\frac{L}{2K},\frac{L-K}{K}} * \mathbf{x}_{\frac{-L}{K},-\frac{L}{2K}}$$
$$\mathbf{y}_1{=}\mathbf{w}_0 * \mathbf{x}_{0,1}(t) + \mathbf{w}_1 * \mathbf{x}_{-1,0}(t) + \mathbf{w}_{2,3} * \mathbf{x}_{-3,-1} + \ldots + \mathbf{w}_{\frac{L}{2K},\frac{L-K}{K}} * \mathbf{x}_{1-\frac{L}{K},1-\frac{L}{2K}}$$
$$\ldots$$

The first term is a length K convolution, which must be computed in the time-domain if zero lag is required. Because all input blocks prior to $\mathbf{x}_0(t)$ are known before $\mathbf{y}_0(t)$ is required, all of the other convolutions can be performed in the frequency domain. This means that the input signals must be filtered separately by unique block filters, of length

$$2^k K, \qquad k = 0, 1, 2, \ldots, \log_2\left(\frac{L}{2K}\right)$$

which means that each of the input signals need to be transformed to the frequency domain $\log_2\left(\frac{L}{K}\right)$ times, each time doubling the sub-block size.

Keeping in mind that to avoid circular convolution, the FFT size must be twice that of the sub-block size, to filter the entire block of $\mathbf{x}(t)$ using sub-block filters of size $2^k K$ requires $\frac{L}{2^k K}$ length-$2^{k+1}K$ FFTs. Each length-$2^{k+1}K$ FFT requires

$$\begin{aligned} & 2^{k+1}K\log_2\left(2^k K\right) - 5 \times 2^k K + 6 \\ =& 2^{k+1}K\left(\log_2\left(2^k\right) + \log_2 K\right) - 5 \times 2^k K + 6 \\ =& 2^{k+1}K\left(k + \log_2 K\right) - 5 \times 2^k K + 6 \qquad \text{real multiplies,} \end{aligned}$$

resulting in a total of

$$\begin{aligned} & \sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} \frac{L}{2^k K}\left(2^{k+1}K\left(k + \log_2 K\right) - 5 \times 2^k K + 6\right) \\ =& \sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k + \log_2 K\right) - 5L + \frac{6L}{2^k K} \qquad \text{real multiplies} \end{aligned}$$

for conversion of each signal into the frequency domain and the same to convert back to the time domain. Over the entire algorithm, this results in

$$2N \sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k + \log_2 K\right) - 5L + \frac{6L}{2^k K} \qquad \text{real multiplies} \tag{C.8}$$

Each of the filters have to be divided into sub-blocks and converted into the frequency domain also. However, every time-domain filter weight is only converted to the frequency domain once, unlike the time-domain samples of $x(t)$. For every filter, one FFT for each $2^k K$-length sub-block is needed, and there is only one sub-block for each k. This means that

$$\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right) - 5\times 2^k K + 6 \qquad \text{real multiplies,}$$

are needed for each of the $N(N-1)$ filters. This means that to convert all of the filters for hybrid filtering,

$$N(N-1)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right) - 5\times 2^k K + 6\right) \qquad \text{real multiplies} \tag{C.9}$$

are required.

For the actual filtering, elementwise multiplication in the frequency domain is needed. For each frequency-domain filter that passes over the sequence, L complex multiplications are needed. Because there are $\log_2\left(\frac{L}{K}\right)$ frequency-domain sub-filters, this results in a total of

$$4\log_2\left(\frac{L}{K}\right)L \qquad \text{real multiplies}$$

per separating filter, or

$$N(N-1)\,4\log_2\left(\frac{L}{K}\right)L \qquad \text{real multiplies}$$

for all of the filters. Note that this will become negative for $L \leq K$. We therefore use $\hat{K}$ to denote the lesser of K and L.

$$N\left(N-1\right)4\log_2\left(\frac{L}{\hat{K}}\right)L \qquad \text{real multiplies} \tag{C.10}$$

The last part of the calculation is finding the number of multiplications required by the time-domain filtering. Because the time-domain filter is $\hat{K}$ taps long, it requires

$$N\left(N-1\right)\hat{K}L \qquad \text{real multiplies} \tag{C.11}$$

for all of the filters.

Combining (C.8), (C.9), (C.10) and (C.11) gives the total number of multiplications required for the filtering portion of the algorithm.

$$\begin{aligned}
&2N\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k+\log_2 K\right)-5L+\frac{6L}{2^kK}\\
&+N\left(N-1\right)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right)-5\times 2^kK+6\right)\\
&+N\left(N-1\right)4\log_2\left(\frac{L}{\hat{K}}\right)L\\
&+N\left(N-1\right)\hat{K}L
\end{aligned}$$

$$
\begin{aligned}
=&2N\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k+\log_2 K\right)-5L+\frac{6L}{2^kK}\\
&+N\left(N-1\right)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right)-5\times 2^kK+6\right)\\
&+4\left(N^2-N\right)\left(\log_2 L-\log_2 \hat{K}\right)L+N^2\hat{K}L-N\hat{K}L
\end{aligned}
$$

$$
\begin{aligned}
=&2N\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k+\log_2 K\right)-5L+\frac{6L}{2^kK}\\
&+N\left(N-1\right)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right)-5\times 2^kK+6\right)\\
&+4N^2L\left(\log_2 L-\log_2 \hat{K}\right)-4NL\left(\log_2 L-\log_2 \hat{K}\right)+N^2\hat{K}L-N\hat{K}L
\end{aligned}
$$

$$
\begin{aligned}
=&2N\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k+\log_2 K\right)-5L+\frac{6L}{2^kK}\\
&+\left(N^2-N\right)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right)-5\times 2^kK+6\right)\\
&+4N^2L\log_2 L+\left(\hat{K}-4\log_2 \hat{K}\right)N^2L-4NL\log_2 L-\left(\hat{K}-4\log_2 \hat{K}\right)NL
\end{aligned}
$$

Combining this with (C.7) gives the total number of multiplies for the entire algorithm.

$$\begin{aligned}
&2N \sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k+\log_2 K\right) - 5L + \frac{6L}{2^k K} \\
&+\left(N^2-N\right)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right) - 5\times 2^k K + 6\right) \\
&+4N^2 L\log_2 L + \left(\hat{K}-4\log_2\hat{K}\right)N^2 L - 4NL\log_2 L - \left(\hat{K}-4\log_2\hat{K}\right)NL \\
&+4N^2 L\log_2 L - 2N^2 L + 12N^2 - 7LN
\end{aligned} \tag{C.12}$$

$$\begin{aligned}
=&2N \sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2L\left(k+\log_2 K\right) - 5L + \frac{6L}{2^k K} \\
&+\left(N^2-N\right)\left(\sum_{k=0}^{\log_2\left(\frac{L}{2K}\right)} 2^{k+1}K\left(k+\log_2 K\right) - 5\times 2^k K + 6\right) \\
&+8N^2 L\log_2 L + \left(\hat{K}-4\log_2\hat{K}-2\right)N^2 L \\
&-4NL\log_2 L - \left(\hat{K}-4\log_2\hat{K}+7\right)NL + 12N^2 \qquad \text{real multiplies}
\end{aligned} \tag{C.13}$$

C.4 Trinicon

For implementation in the frequency domain, the FFT algorithm will be utilized. According to Bergland in [102], for real time-domain signals, a length $2L$ FFT requires

$$2L\log_2 L - 5L + 6 \quad \text{real multiplies}$$

For the *Triple-N ICA for Convolutive mixtures* (TRINICON) algorithm, there are 2 FFTs per source for the filtering, requiring

$$4L\log_2 L - 10L + 12 \quad \text{real multiplies.}$$

In addition to this, there are also

$$6L\left(2N-1\right) \qquad \text{complex multiplies}$$

for the frequency-domain convolution, which is required for filtering.

For every complex multiplication, four real multiplies are needed. This brings the total number of multiplies for filtering the signals to

$$\begin{aligned}
&4L\log_2 L - 10L + 12 + 24L\left(2N-1\right)\\
=&4L\log_2 L - 10L + 12 + 48LN - 24L\\
=&48LN + 4L\log_2 L - 34L + 12 \qquad \text{real multiplies}
\end{aligned} \tag{C.14}$$

For the update, $3N-1$ FFTs are needed per source to find the gradient term. This requires a total of

$$\begin{aligned}
&3\left(N-1\right)\left(2L\log_2 L - 5L + 6\right)\\
=&6NL\log_2 L - 15NL + 18N - 6L\log_2 L + 15L - 18 \qquad \text{real multiplies}
\end{aligned}$$

Aside from the multiplications required for the FFTs, there are also $6L\left(2N-1\right)$ complex multiplications for the update term, requiring

$$\begin{aligned}
&24L\left(2N-1\right)\\
=&48NL - 24L \qquad \text{real multiplies}
\end{aligned}$$

bringing the total number of multiplies for the update to

$$\begin{aligned} &6LN\log_2 L - 15LN + 18N - 6L\log_2 L + 15L - 18 + 48NL - 24L \\ =&6LN\log_2 L + 33LN + 18N - 6L\log_2 L - 9L - 18 \qquad \text{real multiplies} \end{aligned} \tag{C.15}$$

(C.14) and (C.15) can then be summed and multiplied by the number of sources to find the total requirement for the FD-SAD algorithm.

$$\begin{aligned} &6LN^2\log_2 L + 33LN^2 + 18N^2 - 6LN\log_2 L - 9LN - 18N + 48LN^2 + 4LN\log_2 L - 34LN + 12N \\ =&6LN^2\log_2 L + 81LN^2 + 18N^2 - 2LN\log_2 L - 43LN - 6N \qquad \text{real multiplies} \end{aligned} \tag{C.16}$$

D.1 Decorrelation of Multiple Non-Stationary Sources Using a Divide and Conquer Crosstalk-Resistant Adaptive Noise-Canceler

T. J. Moir and J. I. Harris. Decorrelation of multiple non-stationary sources using a multivariable crosstalk-resistant adaptive noise canceller. *International Journal of Adaptive Control and Signal Processing*, 27(5):349367, 2013

Decorrelation of multiple non-stationary sources using a multivariable crosstalk-resistant adaptive noise-canceller.

T.J.Moir and J.I.Harris

Massey University at Albany

School of Engineering and Advanced Technology (SEAT)

Albany

Auckland

New-Zealand.

Abstract

A new structured approach to adaptive noise-cancellingof non-stationary stochastic signals is given. The divide and conquer method subdivides the problem into a number of sources with a power of 2 and subdivides the solution to smaller problems in a similar fashion to that of a fast-Fourier transform (FFT). Hence with number of sources 2^p where p is a positive integer, the problem is successively reduced to simpler solutions at each stage of order $2^p/2$. In its basic form the method uses only multivariable least-mean-squares (MLMS),ordinary LMS and only second-order statistics.

Keywords: Blind-source separation (BSS), least-mean squares (LMS), automatic noise canceller (ANC).

1.Introduction.

The problem of optimally separating a stochastic signal from random noise has its roots in modern times with the work of Wiener [1], Kolmogorov [2] for stationary signals and later Kalman [3] for the non-stationary case. Although these solutions can lead to the solutions of many realistic problems, the more difficult problem of separating non-stationary signals from each-other or non-stationary signals from non-stationary noise cannot be solved for a single sensor (or measurement) without further information of the signal and noise properties. In fact, if the spectra of signal and noise are overlapping, then removing the noise will always impair the signal. It was later shown that by using two or more sensors that extra information can be gathered about the signal and noise properties and the beginnings of a solution could be found under certain restrictive conditions[4]. For example, when using two sensors to cancel a single noise-source requires one of the sensors to pick up the mixture of signal and noise and the second to pick up just the noise alone. This is rarely encountered in many real-world problems since more than often the noise sensor also picks up the signal as well as the noise and the automatic noise-canceller (ANC) ends up cancelling the desired signal with the noise. Placing sensors long distances apart does not improve matters either, since very long filter lengths are needed and this leads to excess mean-square error. There are certain applications (eg where there is a natural acoustic barrier for sound signals) between sensors [5] that the basic adaptive filtering method will work, but these are not so commonly met in everyday office or outdoor situations.

Nevertheless the fundamental solution, which relies on the least-mean-squares (LMS) algorithm can be modified so that the sensors are close together. There have been many attempts at this for the case of acoustic signals and noise but the main drawback has been that the modified ANC requires a voice-activity detector (VAD) that can rapidly detect periods of noise and switch multiple LMS algorithms on and off depending on whether noise on its own is present.[6, 7]. Essentially the first LMS activates during noise-alone and the second during signal plus noise. Although speech has the advantage that many silence periods exist for normal sentences, it is nevertheless harder than expected for a VAD to determine what is speech and what is noise in some environments where for example there are two or more competing talkers. In the so-called cocktail party effect, it is difficult for such an algorithm to know which speech signal is desired and which is to be rejected. Geometric approaches do exist that simplify the VAD problem somewhat to a particular direction[8], but nevertheless it becomes necessary to find a method that does not rely on VADs at all.

Such a method has been explored in so-called blind-source separation (BSS) using the natural-gradient algorithm[9]. However such approaches can be computationally intensive and rely on probability density functions to model the desired speech. These approaches have been termed unsupervised adaptive filters as opposed to methods which rely on measured reference signals which are now known as supervised adaptive filters. This is with analogy to neural networks where the theory of many of the BSS methods originated independently of the LMS based approaches.

There is simpler method however which can separate two (later extended to more than two) non-stationary signals (or a signal from noise) without a VAD and by using LMS or the faster but more complex recursive-least squares (RLS) methods. The method relies on de-correlation between the two sources[10, 11] and separates the sources up to a polynomial ie the separated speech is convolved with an unknown filter. Fortunately the dynamics of this filter does little or no harm to the quality of the recovered speech. A similar approach is given by Weinstein et al [12] and for multiple sources using LMS and RLS by Mei and Yin [13]. Other more sophisticated methods also exist e.g. [14] which take into account of the inherent weight-vector bias that the method involves. The technique has become known as crosstalk-resistant adaptive noise-cancelling. (CRANC), or even symmetric adaptive decorrelation (SAD).

The CRANC method employed here is for multiple blind-source-separation and is an extension of the earlier work to the multivariable case. Although the multivariable case has been considered elsewhere [13], their method is essentially a scalar technique applied to the multivariable case, whereas this approach is entirely different. Note that technically the method should be known as an adaptive noise-cancelling approach but since the sources can be separated from one another the term BSS is also of some relevance.

Mathematical preliminaries:

If a polynomial-matrix defined as $\boldsymbol{X}(z^{-1}) = \boldsymbol{X}^1 + \boldsymbol{X}^2 z^{-1} + \boldsymbol{X}^3 z^{-2} + ... + \boldsymbol{X}^n z^{-n}$ of degree n with real matrix coefficients has *all* roots of $\det[\boldsymbol{X}(z^{-1})] = 0$ *inside* the unit circle in the z plane, then it is termed strict sense minimum phase. Superscripts are used here to denote the individual polynomial matrix and should not be confused with the usual usage as the power of an expression. For example the term z^{-1} denotes the inverse of the scalar z-transform operator in the usual fashion. Where the superscript is used as a power of a matrix, we use square brackets $[\boldsymbol{\Phi}]^{k-i-1}$ thus to denote a matrix raised to the power k-i-1 and to avoid confusion with its other meaning discussed above.

No zeros are assumed to be on the unit circle for any polynomial matrix. For a polynomial-matrix with *any* roots of $\det[\boldsymbol{X}(z^{-1})]=0$ lying *outside* of the unit circle in the z-plane, it is termed non-minimum phase. For simplicity $\boldsymbol{X}(z^{-1})$ is often written as $\boldsymbol{X}$, omitting the complex argument. To avoid mixing time-domain and frequency-domain properties, define the backwards shift operator q^{-1} when dealing with time-domain signals. Hence for some signal or vector $q^{-1}\boldsymbol{x}_k=\boldsymbol{x}_{k-1}$ or $\boldsymbol{y}_k=\boldsymbol{X}(q^{-1})\boldsymbol{u}_k$ .The identity matrix is defined as $\boldsymbol{I}_n$ for an identity matrix of order n. For example, in the text $\boldsymbol{I}_4$ and $\boldsymbol{I}_2$ represent the identity matrices of order 4 and 2 respectively. For matrix polynomials we use the notation of subscripts to denote sub-matrix polynomials. For example

$$\boldsymbol{G}(z^{-1})=\begin{bmatrix}\boldsymbol{G}_{11}(z^{-1}) & \boldsymbol{G}_{12}(z^{-1})\\ \boldsymbol{G}_{21}(z^{-1}) & \boldsymbol{G}_{22}(z^{-1})\end{bmatrix}$$

2.Mixture of 4 random uncorrelated signals.

For simplicity first consider a convolution mixing (or correlating) transfer-function matrix for 4 uncorrelated sources and time-index k=0,1,2...

$$\boldsymbol{x}_k=\boldsymbol{H}(q^{-1})\boldsymbol{t}_k \tag{1}$$

Where $\boldsymbol{H}(z^{-1})$ is a 4X4 matrix-polynomial which can be written as a polynomial-matrix of degree n with $det(\boldsymbol{H}(z^{-1}))\neq 0$.

$$\boldsymbol{H}(z^{-1})=\boldsymbol{H}^0+\boldsymbol{H}^1z^{-1}+...+\boldsymbol{H}^nz^{-n}$$

The vector $\boldsymbol{t}_k=[t_k^1,t_k^2,t_k^3,t_k^4]^T$ is composed of the 4 clean desired zero-mean and uncorrelated random signals which are to be estimated from observations of $\boldsymbol{x}_k$. Should the inverse of $\boldsymbol{H}(z^{-1})$ not exist then it is not possible to separate the sources. No restriction is made on whether the desired signals are stationary or non-stationary, but they must be “persistently exciting”[15] in order for the adaptive estimation algorithms to converge properly. However, the random signals must be purely nondeterministic i.e. the signals cannot be predicted from their own past[16]. Now suppose that $\boldsymbol{H}(z^{-1})$ is split into 4, 2X2 sub-polynomial-matrices thus

$$\boldsymbol{H}(z^{-1})=\begin{bmatrix} \boldsymbol{H}_{11}(z^{-1}) & \boldsymbol{H}_{12}(z^{-1}) \\ \boldsymbol{H}_{21}(z^{-1}) & \boldsymbol{H}_{22}(z^{-1}) \end{bmatrix}$$

as shown in Figure 1.

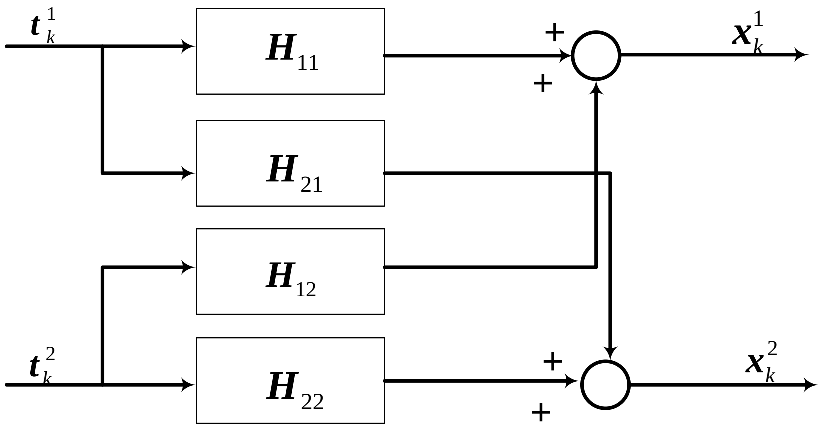


Figure 1. Four input mixture process.

The vectors $\boldsymbol{t}_k^1=[t_k^1,t_k^2]^T$ and $\boldsymbol{t}_k^2=[t_k^3,t_k^4]^T$ shown in bold should not be confused with their scalar counterparts.

Then by defining two vectors of length 2 $\boldsymbol{s}_k^1=\boldsymbol{H}_{11}(q^{-1})\boldsymbol{t}_k^1$ and $\boldsymbol{s}_k^2=\boldsymbol{H}_{22}(q^{-1})\boldsymbol{t}_k^2$ we can write a new simplified 4X4 coupling polynomial-matrix $\boldsymbol{G}$ where

$$\boldsymbol{x}_k=\boldsymbol{G}(q^{-1})\boldsymbol{s}_k \tag{2}$$

with the vectors $\boldsymbol{s}_k=[(\boldsymbol{s}_k^1)^T,(\boldsymbol{s}_k^2)^T]^T$ and $\boldsymbol{x}_k=[(\boldsymbol{x}_k^1)^T,(\boldsymbol{x}_k^2)^T]^T$ also suitably partitioned and

$$\boldsymbol{G}(z^{-1})=\begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{G}_{12}(z^{-1}) \\ \boldsymbol{G}_{21}(z^{-1}) & \boldsymbol{I}_2 \end{bmatrix}$$

where $\boldsymbol{G}_{12}(z^{-1})=\boldsymbol{H}_{12}(z^{-1})\boldsymbol{H}_{22}(z^{-1})^{-1}$ and $\boldsymbol{G}_{21}(z^{-1})=\boldsymbol{H}_{21}(z^{-1})\boldsymbol{H}_{11}(z^{-1})^{-1}$ are 2X2 polynomial matrices. This is illustrated in Figure 2.

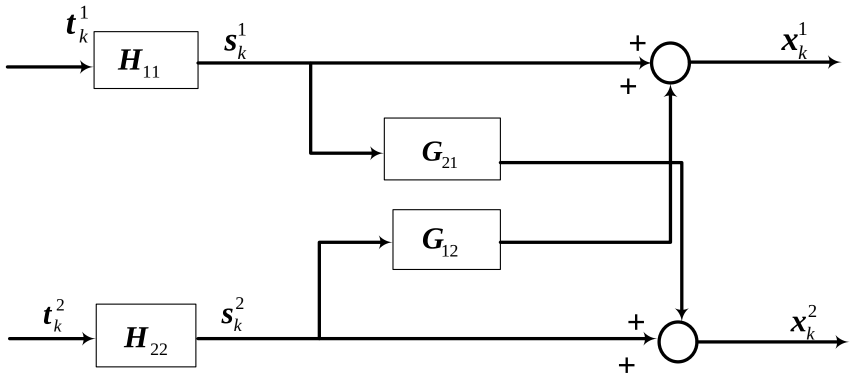


Figure 2. Simplified four input mixture process.

We must assume that the 2x2 sub-polynomial-matrices $\boldsymbol{H}_{11}(z^{-1})$ and $\boldsymbol{H}_{22}(z^{-1})$ also of degree n, are non-singular and for causality must also be strict sense minimum-phase polynomial matrices. For the non-minimum phase case it is not possible to write the polynomial- matrix-fractions $\boldsymbol{G}_{12}(z^{-1})=\boldsymbol{G}_{12}^1 z^{-1}+\boldsymbol{G}_{12}^2 z^{-2}+...$ and $\boldsymbol{G}_{21}(z^{-1})=\boldsymbol{G}_{21}^1 z^{-1}+\boldsymbol{G}_{21}^2 z^{-2}+...$ as a convergent power-series. We can write from the above

$$\boldsymbol{x}_k^1=\boldsymbol{s}_k^1+\boldsymbol{G}_{12}(q^{-1})\boldsymbol{s}_k^2 \tag{3a}$$

$$\boldsymbol{x}_k^2=\boldsymbol{s}_k^2+\boldsymbol{G}_{21}(q^{-1})\boldsymbol{s}_k^1 \tag{3b}$$

and attempt to estimate $\boldsymbol{s}_k^1$ and $\boldsymbol{s}_k^2$ from measurements of $\boldsymbol{x}_k^1$ and $\boldsymbol{x}_k^2$. We then re-define the integer n to be the degree of two polynomial-matrices found from $\boldsymbol{G}_{12}(z^{-1})$ and $\boldsymbol{G}_{21}(z^{-1})$ suitably truncated at n, viz $\boldsymbol{G}_{12}(z^{-1})=\boldsymbol{G}_{12}^1 z^{-1}+\boldsymbol{G}_{12}^2 z^{-2}+...+\boldsymbol{G}_{12}^n z^{-n}$ and $\boldsymbol{G}_{21}(z^{-1})=\boldsymbol{G}_{21}^1 z^{-1}+\boldsymbol{G}_{21}^2 z^{-2}+...+\boldsymbol{G}_{21}^n z^{-n}$. We note that it is necessary when defining these polynomial matrices that the zeroth coefficient matrices are both zero. This is necessary when estimating these matrices in the next section.

The backward vector separation method

If we assume initially that $\boldsymbol{G}_{12}$ and $\boldsymbol{G}_{21}$ are known, then we can define a backward separation method

$$\boldsymbol{y}_k^1=\boldsymbol{x}_k^1-\boldsymbol{G}_{12}(q^{-1})\boldsymbol{y}_k^2 \tag{4a}$$

$$\boldsymbol{y}_k^2 = \boldsymbol{x}_k^2 - \boldsymbol{G}_{21}(q^{-1})\boldsymbol{y}_k^1 \tag{4b}$$

As illustrated in Figure 3.

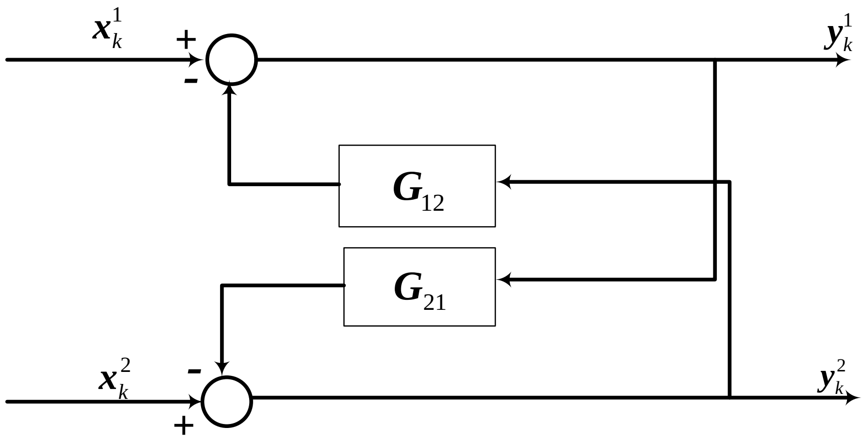


Figure 3. Backwards vector separation method.

By substituting (3a) and 3(b) into (4a) and (4b) and provided

$$\boldsymbol{I}_2 - \boldsymbol{G}_{12}\boldsymbol{G}_{21} \neq \boldsymbol{0}$$

we arrive after some algebra with the two vectors

$$\boldsymbol{y}_k^1 = \boldsymbol{s}_k^1$$

$$\boldsymbol{y}_k^2 = \boldsymbol{s}_k^2$$

3.Multivariable LMS (MLMS)

There are two cross-coupled equations which describe the multivariable LMS (MLMS) backwards separation method for the case with 4 inputs. (two 2 dimensional vectors)

$$\boldsymbol{C}_{k+1}^1 = \boldsymbol{C}_k^1 + \mu_1 \boldsymbol{U}_k^1 (\boldsymbol{e}_k^1)^T \tag{5a}$$

$$\boldsymbol{C}_{k+1}^2 = \boldsymbol{C}_k^2 + \mu_2 \boldsymbol{U}_k^2 (\boldsymbol{e}_k^2)^T \tag{5b}$$

and we define two error vectors of length 2 as part of (5a) and (5b) given by

$$\boldsymbol{e}_k^1 = \boldsymbol{x}_k^1 \text{-} (\boldsymbol{C}_k^1)^T \boldsymbol{U}_k^1 \tag{6a}$$

$$\boldsymbol{e}_k^2 = \boldsymbol{x}_k^2 \text{ - } (\boldsymbol{C}_k^2)^T \boldsymbol{U}_k^2 \tag{6b}$$

The two coupled MLMS equations (5a,b) minimise the mean-square error vectors $E[(\boldsymbol{e}_k^1)^T \boldsymbol{e}_k^1]$ and $E[(\boldsymbol{e}_k^2)^T \boldsymbol{e}_k^2]$, (appendix A), The coupled MLMS algorithm also implements (4a,b) so that the separated vectors appear as the error vectors of each MLMS and $\boldsymbol{C}_k^1, \boldsymbol{C}_k^2$ must also contain estimates of the coefficient matrices of polynomial matrices $\boldsymbol{G}_{12}$ and $\boldsymbol{G}_{21}$ respectively. e.g. for the first of these $\boldsymbol{C}_k^1 = [\hat{\boldsymbol{G}}_{12}^1, \hat{\boldsymbol{G}}_{12}^2 ... \hat{\boldsymbol{G}}_{12}^n]^T$ and $\boldsymbol{C}_k^2 = [\hat{\boldsymbol{G}}_{21}^1, \hat{\boldsymbol{G}}_{21}^2 ... \hat{\boldsymbol{G}}_{21}^n]^T$ for the second. Compare for instance equation (6a,b) with (4,a,b). The two error vectors (6a,b) now provide the separated output vectors after convergence.

The $\boldsymbol{C}_k^1$ and $\boldsymbol{C}_k^2$ matrices are of order 2nx2, μ_1 and μ_2 are step sizes. (see appendix A) Also $\boldsymbol{U}_k^1, \boldsymbol{U}_k^2$ are regressor vectors, each with length 2n whose elements are composed of regressor error terms accordingly

$$\boldsymbol{U}_k^1 = [(\boldsymbol{e}_{k-1}^2)^T, (\boldsymbol{e}_{k-2}^2)^T ... (\boldsymbol{e}_{k-n}^2)^T]^T \tag{7a}$$

and

$$\boldsymbol{U}_k^2 = [(\boldsymbol{e}_{k-1}^1)^T, (\boldsymbol{e}_{k-2}^1)^T ... (\boldsymbol{e}_{k-n}^1)^T]^T \tag{7b}$$

Note that there are no error terms in $\boldsymbol{e}_k$ for physical realiseability. This also implies that $\boldsymbol{G}_{12}(0) = \boldsymbol{G}_{21}(0) = 0$ giving a strict causal set of equations[13].

Substituting (6a) and (6b) into (5a) and (5b) and after some algebra

$$\boldsymbol{C}_{k+1}^1 = [\boldsymbol{I}_{2n} - \mu_1 \boldsymbol{U}_k^1 (\boldsymbol{U}_k^1)^T] \boldsymbol{C}_k^1 + \mu_1 \boldsymbol{U}_k^1 (\boldsymbol{x}_k^1)^T \tag{8a}$$

$$\boldsymbol{C}_{k+1}^2 = [\boldsymbol{I}_{2n} - \mu_2 \boldsymbol{U}_k^2 (\boldsymbol{U}_k^2)^T] \boldsymbol{C}_k^2 + \mu_2 \boldsymbol{U}_k^2 (\boldsymbol{x}_k^2)^T \tag{8b}$$

There are other forms of multichannel LMS in existence e.g. [17, 18] which could also be used but they differ in that the weight matrix herein is a vector in other work. A normalised MLMS version has also been studied[19].

Steady-state performance of ideal solution.

Although the LMS algorithm convergence can easily be proven (appendix A), global convergence of *cross-coupled* multivariable LMS algorithms as in (5a,b) cannot. Even the simple scalar case has not been solved, though a simple example with two weights has been illustrated in[10]. However, what we can prove is that if the algorithm converges at all, then it could converge to the following solution. We do not claim however that this solution will be the one that it converges to. It is worth going down this track initially rather than more complex approaches to see if the method can be made to work though the convergence is not guaranteed. A more rigorous approach would be to include cross-coupling terms in the derivation as was carried out in the scalar case in [20]. It has been shown for the scalar case that the following solution is just one of many possible equilibrium points[21].

Write both equations (8a) and (8b) in augmented state-variable format, and after taking expectations we arrive at:

$$\bar{\boldsymbol{C}}_{k+1} = \boldsymbol{\Phi}\bar{\boldsymbol{C}}_k + \bar{\boldsymbol{B}} \qquad (9)$$

where the 4n symmetric square matrix $\boldsymbol{\Phi}$ is

$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{I}_{2n} - \mu_1 \boldsymbol{R}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{2n} - \mu_2 \boldsymbol{R}_{22} \end{bmatrix}$ and the vector $\bar{\boldsymbol{B}} = [\ \mu_1 \bar{\boldsymbol{B}}_1 \quad \mu_2 \bar{\boldsymbol{B}}_2\ ]^T$ has length 4n and

$\bar{\boldsymbol{C}}_k = \begin{bmatrix} \bar{\boldsymbol{C}}_k^1 & \bar{\boldsymbol{C}}_k^2 \end{bmatrix}^T$.

The bar above the matrices in (9) refers to statistical steady-state.

The two correlations matrices $\boldsymbol{R}_{11} = E[\boldsymbol{U}_k^1 (\boldsymbol{U}_k^1)^T]$, $\boldsymbol{R}_{22} = E[\boldsymbol{U}_k^2 (\boldsymbol{U}_k^2)^T]$ where E[.] is the expectation operator and $\bar{\boldsymbol{B}}_1 = E[\boldsymbol{U}_k^1 (\boldsymbol{x}_k^1)^T]$, $\bar{\boldsymbol{B}}_2 = E[\boldsymbol{U}_k^2 (\boldsymbol{x}_k^2)^T]$. Now in closed-form the state-vector responds to a statistical steady-state vector input $\bar{\boldsymbol{B}}$ according to the standard discrete matrix convolution

$$\bar{\boldsymbol{C}}_k = \sum_{i=0}^{k-1} \boldsymbol{\Phi}^{k-i-1} \bar{\boldsymbol{B}} \qquad (10)$$

which can be written as

$$\bar{\boldsymbol{C}}_k = \sum_{i=0}^{k-1} \boldsymbol{\Phi}^{i} \bar{\boldsymbol{B}} \qquad (11)$$

Now $\sum_{i=0}^{k-1} \boldsymbol{\Phi}^{i}$ is a matrix geometric sequence which has a closed form (provided that the eigenvalues of $\boldsymbol{\Phi}$ all lie *within* the unit circle in the z-plane) $\sum_{i=0}^{k-1} \boldsymbol{\Phi}^{i} = \left[\boldsymbol{I}_{4n} \text{-}\ \boldsymbol{\Phi}^{k}\right] \boldsymbol{I}_{4n}\text{-}\boldsymbol{\Phi}^{-1}$.

Now since $\lim_{k\to\infty} \boldsymbol{\Phi}^{k} \to \boldsymbol{0}$ (the null matrix), the sum to infinity $\sum_{i=0}^{\infty} \boldsymbol{\Phi}^{i} = \boldsymbol{I}_{4n}\text{-}\boldsymbol{\Phi}^{-1}$,and the resulting steady-state converged augmented weight-matrix is

$$\bar{\boldsymbol{C}}_{opt} \to [\boldsymbol{I}_{4n} - \boldsymbol{\Phi}]^{-1}\bar{\boldsymbol{B}} \tag{12}$$

$$\to [\boldsymbol{R}_{11}^{-1}\bar{\boldsymbol{B}}_1 \quad \boldsymbol{R}_{22}^{-1}\bar{\boldsymbol{B}}_2]^T \tag{13}$$

These are the following solutions of the two Wiener-Hopf equations

$$\boldsymbol{R}_{\boldsymbol{U}_1\boldsymbol{U}_1}\bar{\boldsymbol{C}}_k^1 = E[\boldsymbol{U}_k^1(\boldsymbol{x}_k^1)^T] \tag{14a}$$

$$\boldsymbol{R}_{\boldsymbol{U}_2\boldsymbol{U}_2}\bar{\boldsymbol{C}}_k^2 = E[\boldsymbol{U}_k^2(\boldsymbol{x}_k^2)^T] \tag{14b}$$

Where for each case in steady-state, the error vector is orthogonal to the observations of its opposite number.(since $\boldsymbol{U}_k^1$ contains errors in $\boldsymbol{e}_{k-i}^2$ and $\boldsymbol{U}_k^2$ contains errors in $\boldsymbol{e}_{k-i}^1$) It should be noted however in the above proof ,that the two multivariable LMS equations (8a) and (8b) are linked such that the convergence of the second implies the convergence of the first and vice-versa. Therefore as pointed out for the scalar case by[10], global convergence to the optimal weights is by no means assured theoretically due to the fact that the two are bootstrapped together and (14a,b) is just one possible solution.

We can proceed further by considering the first error vector on its own from (6a) and writing

$$J^1 = trE[\boldsymbol{e}_k^1(\boldsymbol{e}_k^1)^T] \tag{15}$$

and minimise the above by finding the derivative of

$$J^1 = trE[\boldsymbol{x}_k^1 - (\boldsymbol{C}_k^1)^T\boldsymbol{U}_k^1][(\boldsymbol{x}_k^1)^T - (\boldsymbol{U}_k^1)^T\boldsymbol{C}_k^1] \tag{16}$$

or expanded

$$J^1 = trE\{\boldsymbol{x}_k(\boldsymbol{x}_k^1)^T - (\boldsymbol{C}_k^1)^T\boldsymbol{U}_k^1(\boldsymbol{x}_k^1)^T - \boldsymbol{x}_k^1(\boldsymbol{U}_k^1)^T\boldsymbol{C}_k + (\boldsymbol{C}_k^1)^T\boldsymbol{U}_k^1(\boldsymbol{U}_k^1)^T\boldsymbol{C}_k^1\} \tag{17}$$

Taking the gradient with respect to the matrix $\boldsymbol{C}_k^1$

$$\nabla_{\boldsymbol{C}_k^1} J = \frac{\partial J^1}{\partial \boldsymbol{C}_k^1} = E[-2\boldsymbol{U}_k^1(\boldsymbol{x}_k^1)^T + 2\boldsymbol{U}_k^1(\boldsymbol{U}_k^1)^T\boldsymbol{C}_k^1 + crossgradientterms] = 0 \tag{18}$$

If the cross gradient terms are assumed to be zero then we quickly arrive at (14a). However, it has been pointed out for the scalar case in [14] and [20], that terms such as $\nabla_{\boldsymbol{C}_k^1} trE[(\boldsymbol{C}_k^1)^T\boldsymbol{U}_k^1(\boldsymbol{x}_k^1)^T] \neq \boldsymbol{U}_k^1(\boldsymbol{x}_k^1)^T$, since this ignores the fact that the weight matrix $\boldsymbol{C}_k^1$ is correlated with the matrix $\boldsymbol{U}_k^1$. However, this paper will make such an assumption since the alternative is a much higher overhead in computation, and the case above still has much merit. It should also be pointed out that although cross-coupled multivariable (and scalar) LMS algorithms have a bias, ordinary multivariable LMS (appendix A) or ordinary LMS does not.

The two individual recovered vectors $\hat{\boldsymbol{s}}_k^1$ and $\hat{\boldsymbol{s}}_k^2$ (the error vectors) are in themselves the output of two further 2X2 polynomial matrix mixing matrices $\boldsymbol{H}_{11}(z^{-1})$ and $\boldsymbol{H}_{22}(z^{-1})$ i.e. $\hat{\boldsymbol{s}}_k^1 = \boldsymbol{H}_{11}(z^{-1})\boldsymbol{t}_k^1$ and $\hat{\boldsymbol{s}}_k^2 = \boldsymbol{H}_{22}(z^{-1})\boldsymbol{t}_k^2$. Both of these signal vectors can be further separated to 4 individual signals by applying two, two-input-output cross-coupled (scalar) LMS algorithms to $\hat{\boldsymbol{s}}_k^1$ and $\hat{\boldsymbol{s}}_k^2$. This will recover the source signals up to an unknown shaping transfer-function for each signal. These scalar transfer functions are unlikely to cause any form of audible loss of quality to the recovered signals since the main source of noise is cross-coupling from the other sources.

4. Separation of non-stationary mixtures.

By using MLMS it has been shown that a vector of four mixed sources can be separated into two vectors, each with two mixed sources. The separation of two mixed sources has been explored in some detail elsewhere[10] and requires only two cross-coupled LMS algorithms. For example, consider the two vectors recovered from the coupled 2 X 2-input MLMS algorithms, $\hat{\boldsymbol{s}}_k^1 = \boldsymbol{H}_{11}(q^{-1})\boldsymbol{t}_k^1$ and $\hat{\boldsymbol{s}}_k^2 = \boldsymbol{H}_{22}(q^{-1})\boldsymbol{t}_k^2$. They require 4 LMS algorithms in total to recover the 4 individual signals. Overall we require 2 MLMS and 4 LMS algorithms. The separation of two coupled sources is given below[10, 13]:

The 2-input (dual input) separator.

Assume the 2x2 coupling matrix of FIR transfer functions $\boldsymbol{H}_{11}$ is non-singular with

$$\hat{\boldsymbol{s}}_k^1 = \boldsymbol{H}_{11}(q^{-1})\boldsymbol{t}_k^1 = \begin{bmatrix} h_{11}(q^{-1}) & h_{12}(q^{-1}) \\ h_{21}(q^{-1}) & h_{22}(q^{-1}) \end{bmatrix} \begin{bmatrix} t_k^1 \\ t_k^2 \end{bmatrix}$$

To proceed, we must further assume that h_{22} and h_{11} are both minimum-phase

and write

$$\begin{bmatrix} s_k^1 \\ s_k^2 \end{bmatrix} = \begin{bmatrix} 1 & g_{12}(q^{-1}) \\ g_{21}(q^{-1}) & 1 \end{bmatrix} \begin{bmatrix} r_k^1 \\ r_k^2 \end{bmatrix}$$

where $r_k^1 = h_{11}t_k^1$ and $r_k^2 = h_{22}t_k^2$, $g_{12} = h_{12}/h_{22}$, $g_{21} = h_{21}/h_{11}$ and we estimate the r terms instead of the t terms. The r signals will only differ by FIR transfer functions from the t signals. Separation is then found via the weight-vector updates of the two LMS algorithms:

$$\boldsymbol{w}_{k+1}^1 = \boldsymbol{w}_k^1 + \mu_1 e_k^1 (\boldsymbol{X}_k^1)^T \qquad (19a)$$

$$\boldsymbol{w}_{k+1}^2 = \boldsymbol{w}_k^2 + \mu_2 e_k^2 (\boldsymbol{X}_k^2)^T \qquad (19b)$$

Where $e_k^i = s_k^i - \boldsymbol{X}_k^T \boldsymbol{w}_k^i, i = 1,2$, $\boldsymbol{X}_k^1 = [e_{k-1}^2, e_{k-2}^2 ... e_{k-n}^2]^T$ and $\boldsymbol{X}_k^2 = [e_{k-1}^1, e_{k-2}^1 ... e_{k-n}^1]^T$

and the two de-correlated random signals are $\hat{r}_k^i = e_k^i, i = 1,2$. We make the assumption for strict causality that the two polynomials $g_{12}(0) = g_{21}(0) = 0$. The same technique can then be applied to a non-singular $\boldsymbol{H}_{22}$. The convergence of this algorithm is studied in [10]. A further refinement of a delay in one of the paths is discussed in [22].

The 4-input separator.

The 4 source blind separation solution is illustrated below.

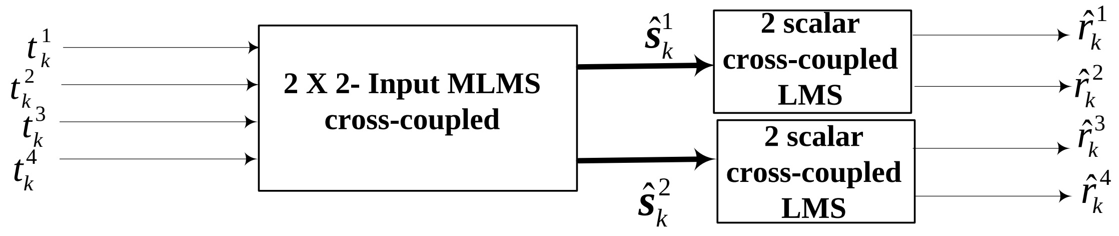


Figure 4. The 4-input separator.

Computational Cost

For L=4 sources it is easy to find the computational cost of this algorithm. Figure 4 shows that we require 2 MLMS algorithms and 4 LMS algorithms. Define N=n+1 where n is the order of the system to be identified using either LMS or NLMS.

Then for 1 MLMS requires

Multiplies: $2NL^2 + L$

Adds/subtracts: $2NL^2 + L - 1$

For 1 LMS requires

Multiplies: $2N + 1$

Adds/subtracts: $2N$

We note that although 2 MLMS algorithms are required for L=4 that each error vector is of length L/2. Therefore the total computation can be found as

Total computation cost for L=4 (2 MLMS + 4 LMS)

Multiplies: $24N + 8$

Adds/subtracts: $24N + 2$

At this stage it is convenient to compare with previous work[13] which uses solely LMS algorithms. The Multi-LMS algorithm is as follows

$$e_k^i = x_k^i - \sum_{j=1;\, j\neq i}^{L} (\hat{\boldsymbol{w}}_k^{ij})^T \boldsymbol{Y}_k^j \tag{20a}$$

$$\boldsymbol{w}_k^{ij} = \boldsymbol{w}_{k-1}^{ij} + \mu_{ij} e_k^i \boldsymbol{Y}_k^j, i, j = 1,2,...L, i \neq j \tag{20b}$$

Where $\boldsymbol{Y}_k^j = [e_{k-1}^j, e_{k-2}^j ... e_{k-n}^j]^T$.

These equations require L(L-1) LMS sets of equations. For L=4 this is 12 sets of LMS requiring

Multiplies: $24N + 12$

Adds/subtracts: $12N$

So we find for L=4 that the Multi-LMS method requires 4 less multiplies and 2 more adds/subtracts. This will potentially provide a marginal improvement in computation time.

Separation of 8 sources or above.

The separation of $L = 2^p$ sources for some integer p can be performed by using the results for 1,2,4 etc in higher-order solutions. For example for p=3, 8 sources, we require two NLMS algorithms each estimating error vectors of size 4. The algorithm already discussed for separating 4 coupled sources can then be used twice to separate the two error vectors each with 4 mixed outputs. The algorithm for 4 inputs in turn relies on two- 2- input algorithms.

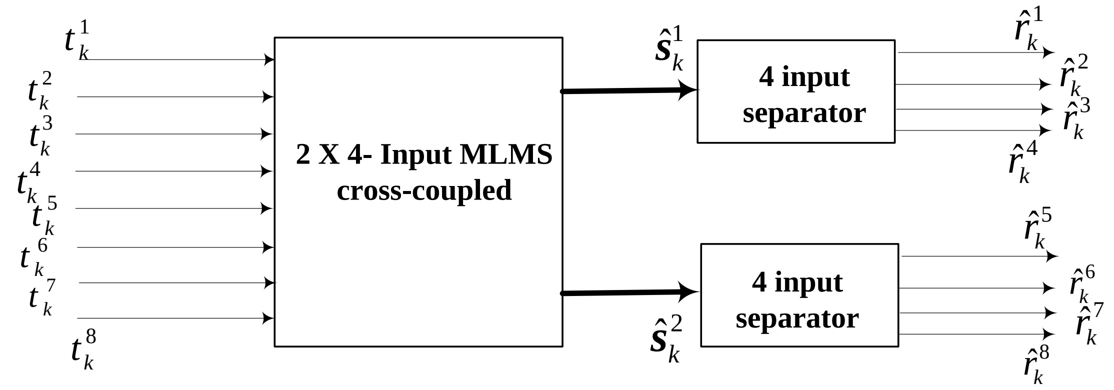


Figure 5. Illustrates the separation of 8 mixed non-stationary random signals.

For the new algorithm when L=8,we need 112N+24 multiplies and 112N+10 adds/subtracts. For the Multi-LMS algorithm we require L(L-1)=56 LMS algorithms which results in 112N+56 multiplies and 112N adds/subtracts. The proposed method therefore requires 32 fewer multiplies and 10 more adds/subtracts.

Likewise we can show that for L=16 that the new approach needs 176 fewer multiplies 34 more adds/subtracts. The Multi-LMS method also requires no fewer than 240 LMS equations.

For an L-square convolution mixing matrix $\boldsymbol{H}(z^{-1})$ we must have for separation to exist that the mixing matrix is invertible and that its z-transform elements $h_{ij}(0)=0, i \neq j, i, j = 1, 2...L$ for causal backward separation to exist. *We also require that the diagonal elements of the mixing matrix* $\boldsymbol{H}(z^{-1})$ *must be all strict sense minimum phase. i.e. the roots of each of* $h_{ii}(z^{-1})=0, i=1,2...L$ *must have their roots within (and not on) the unit circle of the z-plane.*

Algorithm 4.1 Multivariable CRANC for L source separation.

For L=2 sources we use the previously references cross-coupled dual LMS algorithms.

For L>2 sources:

Step 1. Ensure that L>2 is an integer power of 2 and employ a hierarchical multivariable LMS approach beginning with two cross-coupled MLMS algorithms each with error-vector size L/2.

Step 2. The two L/2 error vectors are then feed two 2 further cross-coupled MLMS algorithms each with error size L/4 giving 4 lots of MLMS algorithms of size L/4.

Step3. This procedure is repeated until finally there are only two dimensional error-vectors remaining. There will be L/2 lots of cross-coupled dual-LMS algorithms left (each with two error outputs) and hence L separated signals in total.

For any L there will be L-2 MLMS cross-coupled algorithms used in total plus L/2 cross-coupled scalar LMS algorithms.

4. Results

Simulation

Four signals were mixed using the following second order mixing system

$$\mathbf{H}_0 = \mathbf{I}_4$$

$$\mathbf{H}_1 = \begin{bmatrix} 0 & -0.1 & 0.4 & -0.6 \\ 0.6 & 0 & 0.7 & 0.6 \\ 0.5 & 0.3 & 0 & 0.6 \\ 0.3 & 0.9 & -0.4 & 0 \end{bmatrix}$$

$$\mathbf{H}_2 = \begin{bmatrix} 0 & 0 & -0.2 & 0.2 \\ 0.3 & 0 & 0.4 & -0.3 \\ -0.3 & 0.1 & 0 & 0.5 \\ -0.2 & 0.7 & 0.1 & 0 \end{bmatrix}$$

There was only one source of interest; the other three sources were taken to be background noise disturbances. The source of interest, shown as the first source in Figure 6 was the speech sample kdt_002.wav taken from the KED_TIMIT database. The other three signals were samples of a car assembly line taken from the NOISEX database, labelled "Factory floor noise 2" at the Signal Processing Information Base(SPIB) at Rice University USA. The separating system used three tap-weights for each of the stages in the divide-and-conquer algorithm.

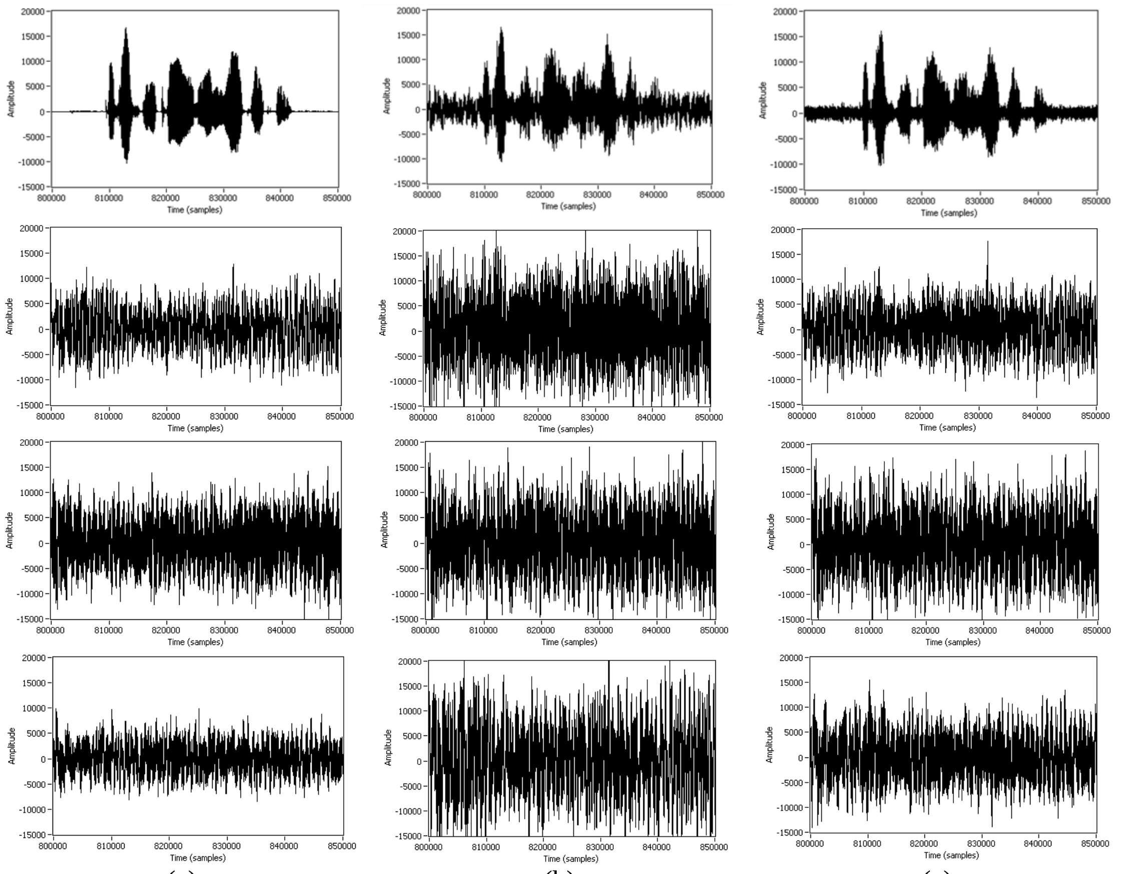
Figure 6 shows a definitive improvement in the signal-to-noise ratio (SNR). Because speech is temporally dependent, there are some periods of noise alone, during which the power of the noise can be estimated. During speech, the combined power of the speech and noise can also be estimated. Assuming that the speech signal is orthogonal to the background noise, the SNR can be calculated using

$$\mathrm{SNR} = 10\log_{10}\ P_{SN} - P_N\ \ -10\log_{10}\ P_N$$

Where P_{SN} is the combined power of the speech and noise, and P_N is the power of the noise by itself. Using this, we calculated an increase in the SNR of 9dB between (b) and (c) of Figure 6.

A real room recording

To test the algorithm in a real environment, we set up the equipment as shown in Figure 7. The room was 4m wide, 7m long, 2.3m high with a carpeted floor, plasterboard walls and ceiling, and typically furnished with a lounge suite, a piano, a table and four chairs. Four loudspeakers were set up in the position of the sources. The noise sources, symbolized by the speakers in Figure 7, were the same noise sources used in the simulation, now only played through the loudspeakers. The speech, symbolized by the head in Figure 7, was the speech sample contained in the package 'Lunatick-20080326-cc.tgz' from the VoxForge speech corpus, also played through a loudspeaker. The microphones were placed in a square, with 20cm between adjacent microphones, and recorded using a sampling rate of 44.1 kHz.



(a) (b) (c)

Figure 6. The separation results of four convolved sources. (a) are the clean input signals before mixing, (b) are the mixed signals as would be blindly received, and (c) are the unmixed signals.

Using 500-taps in both the cross-coupled MLMS and the ordinary cross-coupled LMS, we obtained an increase in SNR of 5dB. Figure 8 shows the results using the both the cross-coupled LMS method described in this paper, and also the results of the method proposed by Mei et al in[13]. Table 1 shows the input SNR and output SNR of both methods with different tap-lengths. The tap-weights for the proposed method defines the number of taps for each stage in the divide-and-conquer algorithm.

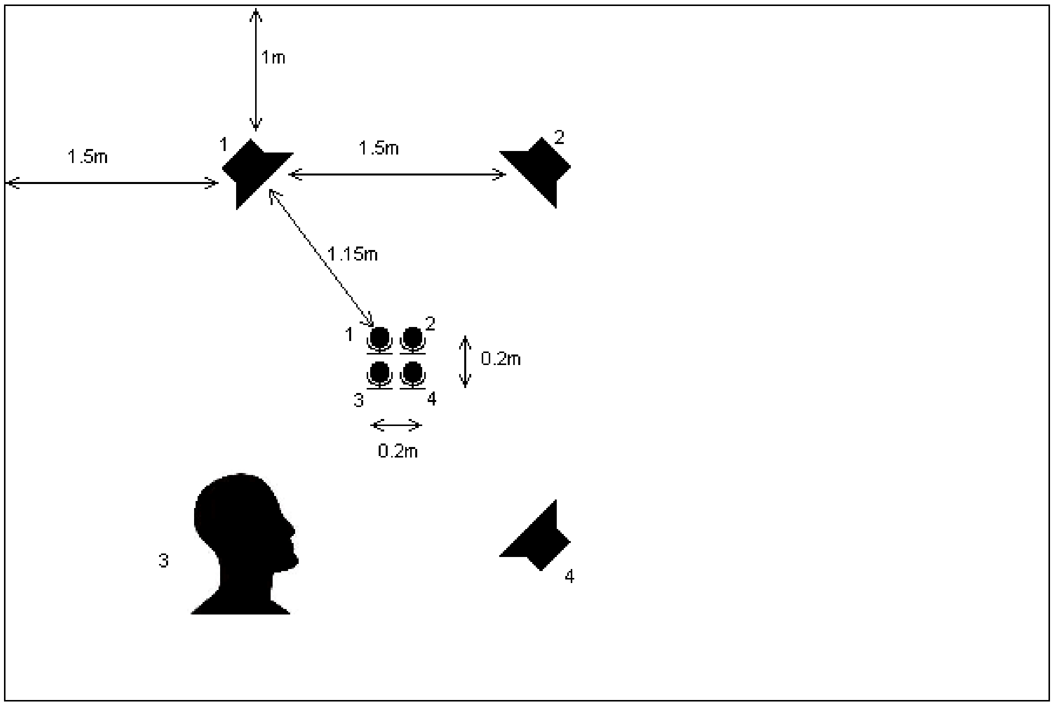


Figure 7. The experimental set-up

	Input SNR	Output SNR	Increase in SNR
Proposed Method			
500 tap-weights	**7.7 dB**	**12.6 dB**	**4.9 dB**
1000 tap-weights	**7.7 dB**	**14.2 dB**	**6.5 dB**
1500 tap-weights	**7.7 dB**	**14.3 dB**	**6.6 dB**
Mei et. al. Method			
500 tap-weights	**7.7 dB**	**12.4 dB**	**4.7 dB**
1000 tap-weights	**7.7 dB**	**13.9 dB**	**6.2 dB**
1500 tap-weights	**7.7 dB**	**13.8 dB**	**6.1 dB**

Table 1. SNR Improvement for the two methods.

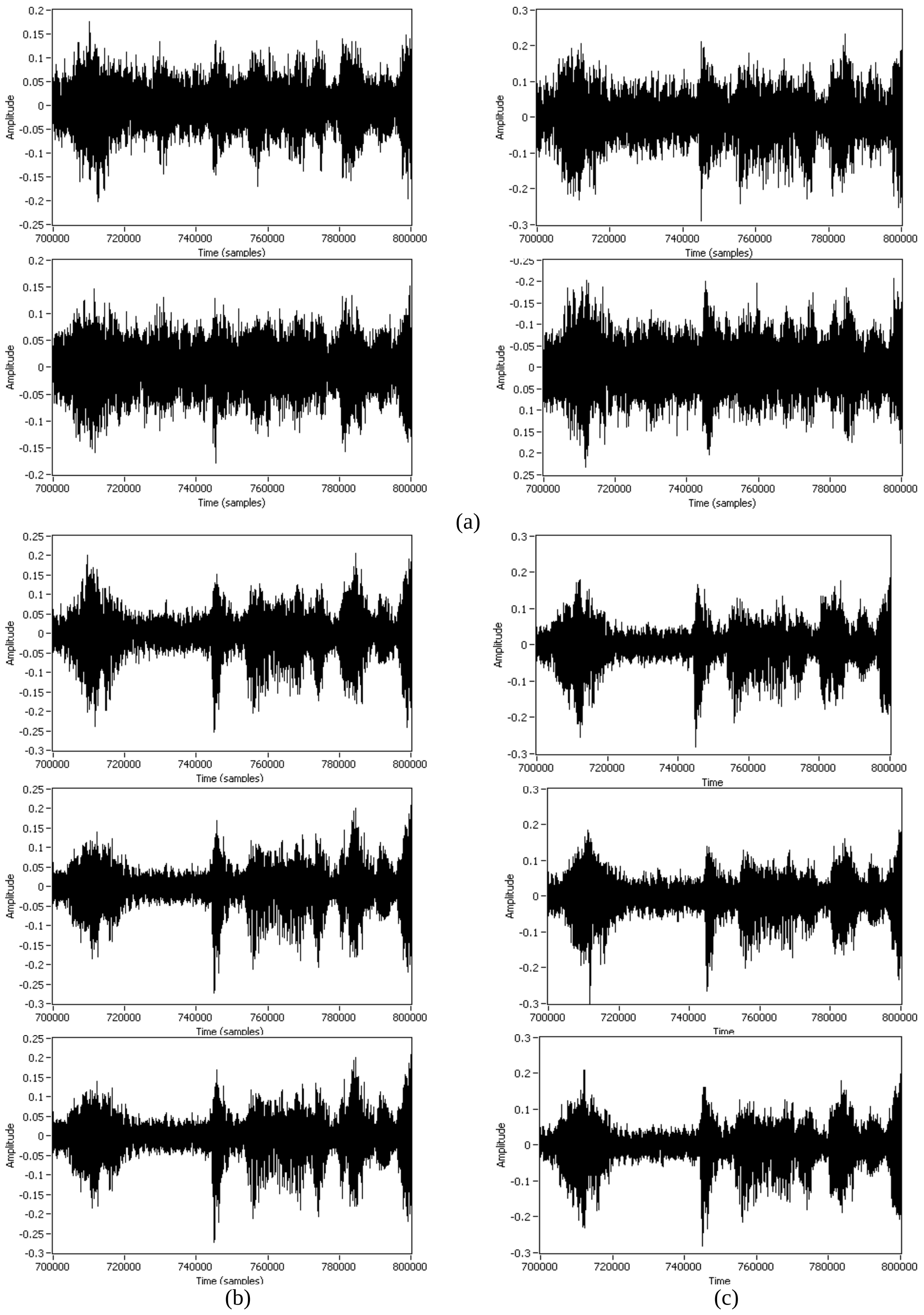


Figure 8. The separation of speech using real recordings. (a) are the recorded signals; (b) is the separated speech using the proposed method and 500, 1000, and 1500 weights

respectively; and (c) is the separated speech using the method proposed by Mei et. al. with 500, 1000, and 1500 weights respectively.

We also note that for such an example we can dispense with one of the two dual-LMS algorithms in the solution (see Figure 4) since the recovery of one signal only is necessary and the three others are “don’t care” noise signals. This gives a computational saving of 4N+1 multiplies and 2N+2 adds/subtracts. These types of realistic noise signals are particularly difficult to remove, especially since they are coming from three different directions. It was found with informal listening tests that the quality of the received speech was quite clear and easy to understand with much roughness from the noise removed including the distant sound of a circular saw. As an alternative to LMS and MLMS an approach using recursive-least squares (RLS) and its multivariable counterpart (Appendix B) can be used instead. However, with large numbers of weights required for realistic environments the time taken to run such simulations can be prohibitively large and tracking performance of RLS is known to be inferior to that of LMS, relying on ad-hoc methods using forgetting factors.

Just as with BSS methods, it is not obvious which of the separated outputs is the one of interest. However, with analogy to the (two LMS) scalar case of two coupled LMS algorithms, if a 90 degree line bisecting the line of microphones is drawn, then the most likely output of the desired speech will occur at the microphone who’s corresponding decorrelated output is nearest. It should also be pointed out here that the same data was run through the algorithm with eight (i.e. an overdetermined system) rather than four microphone inputs with identical results. In this sense the algorithm behaves much like an analogy to the FFT algorithm. That is, the number of data points in an FFT must be a power of 2, but this causes little real-world problems since zero-padding can be used if this is not the case.

Conclusions

A new approach to blind-source separation of non-stationary signals has been shown. The hierarchical method is based on multivariable coupled LMS of decreasing vector size and has been shown to work well in a realistic acoustic environment. Although convergence is not mathematically guaranteed it was found that coupled LMS or MLMS algorithms always converge to a solution that gives overall noise-reduction. A difficult real-world example of separation of a continuous sentence of speech from three non-stationary noise sources has been shown to give improvements in SNR in the range 5-6dB.

Appendix A. Multivariable least-mean square (MLMS) algorithm.

We examine the LMS algorithm where the error is a vector of length L rather than a scalar. Many other authors have also investigated this problem[23] and it is sometimes known as vector LMS or multiple-error LMS. As pointed out in [23] , algorithms which operate on multiple parallel scalar LMS algorithms are not equivalent to the multivariable case since the vector LMS algorithm exploits the crosstalk between the different signals, whereas a bank of scalar LMS algorithms effectively treats the interference as noise.

Preliminaries: We need the following standard matrix results[24]. For matrices **X**,**R**,**A**

$\frac{\partial}{\partial \boldsymbol{X}} tr(\boldsymbol{X}^T \boldsymbol{RX}) = \boldsymbol{RX} + \boldsymbol{R}^T \boldsymbol{X}$ and if **R** is symmetric this result becomes $\frac{\partial}{\partial \boldsymbol{X}} tr(\boldsymbol{X}^T \boldsymbol{RX}) = 2\boldsymbol{RX}$

$\frac{\partial}{\partial \boldsymbol{X}} tr(\boldsymbol{X}^T \boldsymbol{A}) = \boldsymbol{A}$, $\frac{\partial}{\partial \boldsymbol{X}} tr(\boldsymbol{AX}) = \boldsymbol{A}^T$

Cost function:

$$J{=}E[\boldsymbol{e}_k^T \boldsymbol{e}_k]$$

Where E[.] is expected value. We can write the above as

$$J{=}trE[\boldsymbol{e}_k \boldsymbol{e}_k^T]$$

Where $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{y}_k$ is the vector error signal, $\boldsymbol{x}_k$ is the desired signal vector and $\boldsymbol{y}_k = \boldsymbol{C}_k^T \boldsymbol{U}_k$ is the filter vector output all of dimension L. The vector $\boldsymbol{U}_k$ is a vector of regressor vector inputs of dimension (n+1)L and the matrix $\boldsymbol{C}_k$ has dimension (n+1)LXL. We assume for this application that $\boldsymbol{C}_k^T = [\boldsymbol{C}_k^0, \boldsymbol{C}_k^1, \boldsymbol{C}_k^2 ... \boldsymbol{C}_k^n]$ n+1 weight matrices to estimate in total for an nth order multivariable filter. The regressor vector

$$\boldsymbol{U}_k = [(\boldsymbol{u}_k)^T, (\boldsymbol{u}_{k-1})^T ... (\boldsymbol{u}_{k-n})^T]^T$$

Where $\boldsymbol{u}_k$ is the filter input vector of dimension m ie $\boldsymbol{u}_k = [u_k^1, u_k^2 ... u_k^m]^T$. This is illustrated in Figure A1.

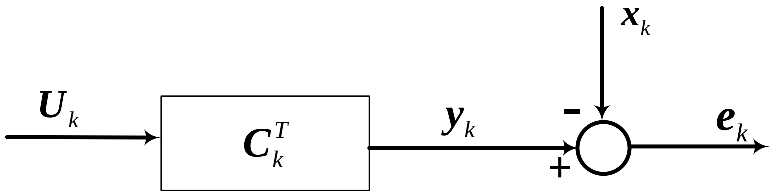


Figure A1. Generation of the error vector for MLMS

Now

$$\boldsymbol{e}_k\boldsymbol{e}_k^T = \boldsymbol{x}_k\boldsymbol{x}_k^T - \boldsymbol{C}_k^T\boldsymbol{U}_k\boldsymbol{x}_k^T - \boldsymbol{x}_k\boldsymbol{U}_k^T\boldsymbol{C}_k + \boldsymbol{C}_k^T\boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{C}_k$$

and

$$J = trE[\boldsymbol{e}_k\boldsymbol{e}_k^T] = trE\{\boldsymbol{x}_k\boldsymbol{x}_k^T - \boldsymbol{C}_k^T\boldsymbol{U}_k\boldsymbol{x}_k^T - \boldsymbol{x}_k\boldsymbol{U}_k^T\boldsymbol{C}_k + \boldsymbol{C}_k^T\boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{C}_k\}$$

Using the previous matrix derivative results gives the gradient vector as:

$$\nabla J = \frac{\partial J}{\partial \boldsymbol{C}_k} = E\{-2\boldsymbol{U}_k\boldsymbol{x}_k^T + 2\boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{C}_k\} = 0$$

Define $\boldsymbol{B} = E[\boldsymbol{U}_k x_k^T]$ and $\boldsymbol{R} = E[\boldsymbol{U}_k\boldsymbol{U}_k^T]$ and we have

$$\boldsymbol{C}_{opt} = \boldsymbol{R}^{-1}\boldsymbol{B}$$

as the optimal weight-matrix. Substituting the value back into J gives the minimum of the cost function after some algebra as

$$J_{\min} = tr[\boldsymbol{B}^T\boldsymbol{C}_{opt}] - E\|\boldsymbol{x}_k\|^2$$

Method of Steepest Descent

The equation for steepest descent has the form

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k - \frac{1}{2}\mu\nabla J$$

$$= \boldsymbol{C}_k + \mu[\boldsymbol{B} - \boldsymbol{R}\boldsymbol{C}_{opt}]$$

We require for convergence a further energy constraint $\sup\frac{1}{2}\boldsymbol{U}_k^T\boldsymbol{U}_k\mu < \kappa < 1$

where κ is a constant and $\boldsymbol{U}_k$ is purely nondeterministic[16].

But **B** and **R** cannot be known *apriori*, then replace them with their instantaneous estimates.

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k + \mu\boldsymbol{U}_k[\boldsymbol{x}_k^T - \boldsymbol{U}_k^T\boldsymbol{C}_k]$$

or alternatively in more familiar form:

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k + \mu \boldsymbol{U}_k \boldsymbol{e}_k^T$$

Where

$$\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{C}_k^T \boldsymbol{U}_k$$

Convergence in the mean of multivariable LMS.

From the earlier result

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k + \mu \boldsymbol{U}_k [\boldsymbol{x}_k^T - \boldsymbol{U}_k^T \boldsymbol{C}_k]$$

Taking expectations and assuming that the regressor vector is uncorrelated with the weight matrix, we have

$$\overline{\boldsymbol{C}}_{k+1} = \overline{\boldsymbol{C}}_k - \mu \overline{\boldsymbol{U}}_k \overline{\boldsymbol{U}}_k^T \overline{\boldsymbol{C}}_k + \mu \overline{\boldsymbol{U}}_k \overline{\boldsymbol{x}}_k^T$$

Defining the correlation $\boldsymbol{R} = E\boldsymbol{[U}_k \boldsymbol{(U}_k\boldsymbol{)}^T]$ and we $\boldsymbol{B} = E\boldsymbol{[}\overline{\boldsymbol{U}}_k \overline{\boldsymbol{x}}_k^T]$ as previously, we have

$$E[\boldsymbol{C}_{k+1}] = [\boldsymbol{I} - \mu \boldsymbol{R}] E[\boldsymbol{C}_k] + \boldsymbol{B}$$

Where the identity and correlation matrices above have dimension (n+1)L. The above expression is a state-variable expression with constant input $\boldsymbol{B}$ and the solution can be found in a similar manner to equation (10)

$$E[\boldsymbol{C}_k] = \sum_{i=0}^{k-1} \boldsymbol{I} - \mu \boldsymbol{R}^{\;k-i-1} \overline{\boldsymbol{B}}$$

or equivalently

$$E[\boldsymbol{C}_k] = \sum_{i=0}^{k-1} \boldsymbol{\Gamma}^{\,i}\, \overline{\boldsymbol{B}}$$

Where the matrix $\boldsymbol{\Gamma} = \boldsymbol{I} - \mu \boldsymbol{R}$ is defined in the above.

Now $\sum_{i=0}^{k-1} \boldsymbol{\Gamma}^{\,i}$ is a matrix geometric sequence which has a closed form (*provided that the eigenvalues of* $\boldsymbol{\Gamma}$ *all lie within the unit circle in the z-plane*) $\sum_{i=0}^{k-1} \boldsymbol{\Gamma}^{\,i} = \boldsymbol{I}\text{-}\boldsymbol{\Gamma}^{\;-1}\left[\boldsymbol{I}\text{- } \boldsymbol{\Gamma}^{\;k}\right]$.

Now since $\lim_{k\to\infty} \boldsymbol{\Gamma}^{\;k} \to \boldsymbol{0}$ (the null matrix), the sum to infinity $\sum_{i=0}^{\infty} \boldsymbol{\Gamma}^{\,i} = \boldsymbol{I}\text{-}\boldsymbol{\Gamma}^{\;-1}$,and the resulting steady-state converged augmented weight-matrix is

$$\overline{\boldsymbol{C}}_{opt} \to [\boldsymbol{I} - \boldsymbol{\Gamma}]^{-1} \overline{\boldsymbol{B}}$$

Which is of course the statistical steady-state optimal (Wiener) weight-matrix.

$$\bar{\boldsymbol{C}}_{opt} \to \boldsymbol{R}^{-1}\boldsymbol{B}$$

We should note therefore that the necessary condition for convergence is that the eigen-values of $\boldsymbol{\Gamma} = \boldsymbol{I} - \mu\boldsymbol{R}$ must all have magnitude less than unity leading to the well-known formula for the upper limit on the step-size:

$$0 < \mu < 1/\lambda_{\max}$$

Where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix. This last step follows simply by a spectral decomposition of $\boldsymbol{\Gamma}$ into

$$\boldsymbol{\Gamma} = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^{-1}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues and $\boldsymbol{T}$ is the modal matrix of eigenvectors. Clearly $\lim_{k\to\infty} \boldsymbol{\Gamma}^{k} \to \boldsymbol{0}$ iff the eigenvalues have magnitude less than unity since

$$\lim_{k\to\infty}\left[\boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^{-1}\right]^{k} \to \boldsymbol{0}$$

Appendix B. Multivariable recursive-least squares (MRLS) algorithm.

Although the scalar RLS algorithm is commonly found in the literature, it is much harder to find a proof of the MRLS algorithm. Traditionally in multivariable control problems the input signal vector to a system is known and hence the MRLS problem can be split into several scalar RLS algorithms instead. However, when, as in the cases here we do not have measurements of the input then we require the MRLS algorithm. The RLS family of algorithms has significantly faster convergence rate than LMS but suffers from lack of good tracking ability and stability issues when used to estimate AR or ARMA type systems. We avoid the stability issues here by using an FIR(MA) model for the multivariate time-series. Considering that the FIR multivariable output vector of order m is $\boldsymbol{y}_k = \boldsymbol{C}_k^T\boldsymbol{U}_k$ and can be written for N=(n+1) consecutive sample vectors (since there are n+1 unknown matrix weights)

$$\begin{bmatrix} \boldsymbol{y}_k^T \\ \boldsymbol{y}_{k+1}^T \\ . \\ . \\ \boldsymbol{y}_{k+N}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_k^T \\ \boldsymbol{U}_{k+1}^T \\ . \\ . \\ \boldsymbol{U}_{k+N}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{C}_k^0 \\ \boldsymbol{C}_k^1 \\ . \\ . \\ \boldsymbol{C}_k^n \end{bmatrix}$$

Or alternatively

$$\boldsymbol{Y}_k = \boldsymbol{X}_k\boldsymbol{W}_k$$

Where $\boldsymbol{Y}_k$ is NXm, $\boldsymbol{X}_k$ is NXNm and $\boldsymbol{W}_k$ is an NmXm matrix respectively. It is then required to find the matrix $\boldsymbol{W}_k$ which contains the impulse response of the multivariable FIR system. ie $\boldsymbol{W}_k = [(\boldsymbol{C}_k^0)^T, (\boldsymbol{C}_k^1)^T ... (\boldsymbol{C}_k^n)^T]^T$.

Define an error matrix

$$\boldsymbol{E}_k^T = \boldsymbol{Y}_k - \boldsymbol{X}_k\boldsymbol{W}_k$$

and minimise the cost function

$$J = min\ trace(\boldsymbol{\Phi}_{EE})$$

Where $\boldsymbol{\Phi}_{EE} = E[\boldsymbol{E}_k\boldsymbol{E}_k^T]$ is error spectral density matrix and E[.] represents statistical expectation.

Using results similar to Appendix A we differentiate J to give the optimal least-squares estimate:

$$\boldsymbol{W}_0 = E(\boldsymbol{X}_k^T\boldsymbol{X}_k)^{-1}E(\boldsymbol{X}_k^T\boldsymbol{Y}_k)$$

Now define an NmXNm symmetric positive-definite error-covariance matrix $\boldsymbol{P}_k = [\boldsymbol{X}_k^T\boldsymbol{X}_k]^{-1} > 0$ and define $\boldsymbol{B}_k = \boldsymbol{X}_k^T\boldsymbol{Y}_k$ and therefore

$\boldsymbol{P}_k^{-1} = \boldsymbol{P}_{k-1}^{-1} + \boldsymbol{U}_k\boldsymbol{U}_k^T$ and $\boldsymbol{B}_k = \boldsymbol{B}_{k-1} + \boldsymbol{U}_k\boldsymbol{y}_k^T$. We can invert $\boldsymbol{P}_k^{-1}$ by using the *matrix inversion lemma* (Sherman-Morrison-Woodbury) $(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{U}^T)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(1 + \boldsymbol{U}^T\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{U}^T\boldsymbol{A}^{-1}$

and this gives an update

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - \frac{\boldsymbol{P}_{k-1}\boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{P}_{k-1}}{1 + \boldsymbol{U}_k^T\boldsymbol{P}_{k-1}\boldsymbol{U}_k}$$

We also define the column vector $\boldsymbol{K}_k$ of length Nm

$$\boldsymbol{K}_k = \frac{\boldsymbol{P}_{k-1}\boldsymbol{U}_k}{1 + \boldsymbol{U}_k^T\boldsymbol{P}_{k-1}\boldsymbol{U}_k}$$

and note that

$$\boldsymbol{K}_k(1 + \boldsymbol{U}_k^T\boldsymbol{P}_{k-1}\boldsymbol{U}_k) = \boldsymbol{P}_{k-1}\boldsymbol{U}_k$$

From which

$$\boldsymbol{K}_k = [\boldsymbol{P}_{k-1} - \boldsymbol{K}_k\boldsymbol{U}_k^T\boldsymbol{P}_{k-1}]\boldsymbol{U}_k$$

$$= \boldsymbol{P}_k\boldsymbol{U}_k$$

Substitute $\boldsymbol{B}_k = \boldsymbol{B}_{k-1} + \boldsymbol{U}_k\boldsymbol{y}_k^T$ and $\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - \boldsymbol{K}_k\boldsymbol{U}_k^T\boldsymbol{P}_{k-1}$ into the weight matrix estimate vector $\hat{\boldsymbol{W}}_k = \boldsymbol{P}_k\boldsymbol{B}_k$

$$\hat{\boldsymbol{W}}_k = [\boldsymbol{P}_{k-1} - \boldsymbol{K}_k\boldsymbol{U}_k^T\boldsymbol{P}_{k-1}][\boldsymbol{B}_{k-1} + \boldsymbol{U}_k\boldsymbol{y}_k^T]$$

We note that $\hat{\boldsymbol{W}}_{k-1} = \boldsymbol{P}_{k-1}\boldsymbol{B}_{k-1}$ so that after simplification we get the weight-matrix update as

$$\hat{\boldsymbol{W}}_k = \hat{\boldsymbol{W}}_{k-1} + \boldsymbol{K}_k[\boldsymbol{y}_k^T - \boldsymbol{U}_k^T\hat{\boldsymbol{W}}_{k-1}]$$

MRLS Summary

To identify an nth order multivariable FIR system with m inputs and outputs. As with the MLMS case (see Fig A1) the regressor vector has the form $\boldsymbol{U}_k = [(\boldsymbol{u}_k)^T, (\boldsymbol{u}_{k-1})^T ... (\boldsymbol{u}_{k-n})^T]^T$. A forgetting factor $0 < \lambda \leq 1$ has been added to improve the tracking performance.

1.Initialise $\boldsymbol{P}_0 = p_0\boldsymbol{I}_{(n+1)m}$ where p_0 can be either small or large depending on the speed of convergence required and initialise the (n+1)mXm weight matrix $\hat{\boldsymbol{W}}_0$ to some initial values (possibly zero).

2. Update the m length column error vector $\boldsymbol{e}_k = \boldsymbol{y}_k - \hat{\boldsymbol{W}}_{k-1}^T\boldsymbol{U}_k]$

3. Update the (n+1)m length column gain vector $\boldsymbol{K}_k = \dfrac{\boldsymbol{P}_{k-1}\boldsymbol{U}_k}{\lambda + \boldsymbol{U}_k^T\boldsymbol{P}_{k-1}\boldsymbol{U}_k}$

4. Update the (n+1)m square covariance matrix $\boldsymbol{P}_k = \dfrac{1}{\lambda}[\boldsymbol{P}_{k-1} - \dfrac{\boldsymbol{P}_{k-1}\boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{P}_{k-1}}{\lambda + \boldsymbol{U}_k^T\boldsymbol{P}_{k-1}\boldsymbol{U}_k}]$

4. Update the weight matrix $\hat{\boldsymbol{W}}_k = \hat{\boldsymbol{W}}_{k-1} + \boldsymbol{K}_k\boldsymbol{e}_k^T$ where $\hat{\boldsymbol{W}}_k = [(\hat{\boldsymbol{C}}_k^0)^T, (\hat{\boldsymbol{C}}_k^1)^T ... (\hat{\boldsymbol{C}}_k^n)^T]^T$

References.

[1] N. Wiener, *Extrapolation, Interpolation and Smoothing of Stationary Time Series*: New York,Wiley, 1949.

[2] A. N. Kolmogorov, "Stationary sequences in Hilbert Space
(English Trans In Kailath,T (ed) Linear Least Squares Estimation, pp.66-89, Dowden, Hutchinson & Ross, Pennsylvania 1977)," *Bolletin Moskovskogo Gosudarstvenogo Universiteta.Matematika,* vol. 2, pp. 1-40, 1941.

[3] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans ASME,Journal of Basic Engineering,* vol. 82 Series D, pp. 35-45, 1960.

[4] B. Widrow, Glover, Jr., J. R., McCool, J. M., Kaunitz, J., Williams, C. S., Hearn, R. H., Zeidler, J. R., Dong, Jr., E., and Goodlin, R. C., "Adaptive noise cancelling: principles and applications," in *Proceedings of the IEEE*, 1975, pp. 1692-1716.

[5] W. A. Harrison, Lim J. S. and Singer, E., " A new application of adaptive noise cancellation," *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on,* vol. 34, pp. 21 - 27, Feb. 1986 1986.

[6] D. Van Compernolle and K. U. Leuven, "Switching adaptive filters for enhancing noisy and reverberant speech from microphone array recordings," in *Proceedings of the IEEE International conference on acoustics,speech and signal processing*, Albuquerque, 1990, pp. 833-836.

[7] J. Vanden Berghe and J. Wouters, "An adaptive noise canceller for hearing aids using two nearby microphones," *Journal of the Acoustical Society of America,* vol. 103, pp. 3621-3626, Jun 1998.

[8] H. Agaiby and T. J. Moir, "A robust word boundary detection algorithm with application to speech recognition," in *Digital Signal Processing Proceedings, 1997. DSP 97., 1997 13th International Conference on*, 1997, pp. 753-755 vol.2.

[9] A. Cichocki and S. Amari, *Adaptive Blind Signal and Image Processing*: John Wiley,England, 2002.

[10] S. Van Gerven and D. Van Compernolle, "Signal separation by symmetric adaptive decorrelation: stability, convergence, and uniqueness," *Signal Processing, IEEE Transactions on,* vol. 43, pp. 1602-1612, 1995.

[11] R. Zinser, Jr., G. Mirchandani, and J. Evans, "Some experimental and theoretical results using a new adaptive filter structure for noise cancellation in the presence of crosstalk," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, 1985, pp. 1253-1256.

[12] E. Weinstein, M. Feder, and A. V. Oppenheim, "Multi-channel signal separation by decorrelation," *Speech and Audio Processing, IEEE Transactions on,* vol. 1, pp. 405-413, 1993.

[13] T. Mei and F. Yin, "Blind separation of convolutive mixtures by decorrelation," *Signal Processing,* vol. 84, pp. 2297-2313, 2004.

[14] G. Mirchandani, R. L. Zinser, Jr., and J. B. Evans, "A new adaptive noise cancellation scheme in the presence of crosstalk [speech signals]," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on,* vol. 39, pp. 681-694, 1992.

[15] A. H. Jazwinski, *Stochastic processes for filtering theory*: Adademic Press, 1970.

[16] V. Solo, "The limiting behavior of LMS," *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 37, pp. 1909-1922, 1989.

[17] S. Elliott, I. Stothers, and P. Nelson, "A multiple error LMS algorithm and its application to the active control of sound and vibration," *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 35, pp. 1423-1434, 1987.

[18] S. J. Elliott and P. A. Nelson, "Algorithm for multichannel LMS adaptive filtering," *Electronics Letters,* vol. 21, pp. 979-981, 1985.

[19] J. Benesty, F. Amand, A. Gilloire, and Y. Grenier, "Adaptive filtering algorithms for stereophonic acoustic echo cancellation," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, 1995, pp. 3099-3102 vol.5.

[20] L. Lepauloux, P. Scalart, and C. Marro, "A Efficient Low-Complexity Algorithm for Crosstalk-Resistant Adaptive Noise-Canceller," in *17th European Signal Processing Conference (EUSIPCO 2009)*, Glasgow,Scotland, August 24-28, 2009, pp. 204-208.

[21] D. Van Compernolle and S. Van Gerven, "Feedforward and feedback in a symmetric adaptive noise canceler.," in *Eusipco-92,Sixth European Signal Processing Conference Aug 24-27 1992.*, Brussels Belgium, 1992, pp. 1081-1084.

[22] S. M. Kou and W. M. Peng, "Principle and applications of asymmetric

crosstalk-resistant adaptive noise canceler," *Journal of the Franklin Institute,* vol. 337, pp. 57-71, 2000.

[23] A. Batra and J. R. Barry, "Blind cancellation of co-channel interference," in *Global Telecommunications Conference, 1995. GLOBECOM '95., IEEE*, 1995, pp. 157-162 vol.1.

[24] K. B. Peterson and M. S. Pedersen, *The Matrix Cookbook*, 20081110 ed.: Technical University of Denmark, 2008.

D.2 Blind Source Separation for Adaptive Speech Control

T. Moir, F. Alam, and J. Harris. Blind source separation for adaptive speech control. In *The eighteenth annual Conference on Mechatronics and Machine Vision in Practice, Brisbane, Australia, 2011-12-06 - 2011-12-08*. AUT University, 2011

Blind Source Separation for Adaptive Speech Control

Jonathan Harris[1], Tom Moir[2], and Fakhrul Alam[1]

[1]School of Engineering and Advanced Technology, Massey University, Auckland, New Zealand
[2]School of Engineering, Auckland University of Technology, Auckland, New Zealand

Abstract

Crosstalk resistant adaptive noise cancellation (CTRANC) is a method of separating convolutively mixed sources where little *a priori* information is known about the system. Possible areas of application for such an algorithm include speech signal processing, in telecommunications, and in the biomedical industry. In this paper we propose a novel adaptation to the traditional CTRANC which increases computational efficiency when the number of sources fits the requirement of $L = 2^n$ where $n \in \mathbb{Z}$ and $n > 1$. Preliminary results also show a modest improvement in separation performance when comparing it to the multiple-input multiple-output method proposed by Mei and Yin (2004).

1 Introduction

The blind source separation problem is the problem of trying to identify the individual sources with no *a priori* knowledge of the sources or the mixing system. Normally, all that can actually be acquired is different mixtures of the sources using multiple sensors. If only interested in the unmixed signals, this crosstalk has a detrimental effect on the usefulness of the acquired signal, and if significant enough, may render the raw signal totally unusable.

A real-life example of blind source separation is what is known as the “cocktail party problem”. Consider the case where there is a room of people, all of whom are talking simultaneously; the human brain is able to adequately extract one person’s speech from the rest. If an algorithm can be found to replicate these results, it would provide a very useful tool in the area of automatic speech recognition, which in turn could be used for speech control.

The application of such an algorithm need not be restricted solely to audio applications. In the medical world it could be used to isolate signals for electrocardiograms (ECGs) or electromyograms (EMGs) (Zhang and Cichocki 2000). It could also be used in telecommunications to reduce the crosstalk created by multiple transmitters (Pedersen et al. 2007). Another less obvious application for blind source separation is to separate images that have been mixed (Amari and Cichocki 1998), though this does not apply to CTRANCs.

One trivial way of solving this problem is to use a Widrow-Hoff least mean-squares (LMS) filter and an approximation of the noise signal to remove the noise from the mixture. However, this has the fundamental flaw that the noise signal has to be relatively signal-free. While there may be situations in which acquiring such a noise approximation is the quite plausible (for example, in a jet cockpit, where the engine noise can be obtained with negligible speech crosstalk), in the majority of everyday situations this assumption cannot be justified.

To overcome this problem, Zinser *et al.* (1985) proposed a cross-talk resistant adaptive noise canceller. The basic premise was that cross-coupling two LMS filters could result in an adaptive noise canceller that was not susceptible to crosstalk from the desired signal in the noise estimate.

In this paper, we propose a novel adaptation to the cross-talk resistant noise canceller that utilizes vector-LMS to increase the computational efficiency of the algorithm when dealing with 2^k input signals, where $k > 1$. We then show that the proposed algorithm actually slightly outperforms the multiple-input multiple-output (MIMO) cross-talk resistant adaptive noise canceller proposed by Mei and Yin (2004) in terms of input-output signal-to-noise ratios with while reducing computational complexity.

This paper is organized as follows. In section 2, the background information of all of the components required for the development of a CTRANC based of vector-LMS are discussed. Section 3 shows the derivation of the novel algorithm, and compares its computational complexity to the CTRANC proposed by Mei and Yin (2004). The experimental set-up and results are discussed in section 4, and the paper is then concluded in section 5.

2 Background Information

The Mixing System

We will first consider the case of a two-input, two-output (TITO) system. In matrix form this is

$$\mathbf{x}(t) = \mathbf{G}^T(t)\tilde{\mathbf{S}}(t) \tag{1}$$

where $\mathbf{x}(t) = [x_1(t), \quad x_2(t)]^T$, the superscript ^{T} denotes the transpose operator,

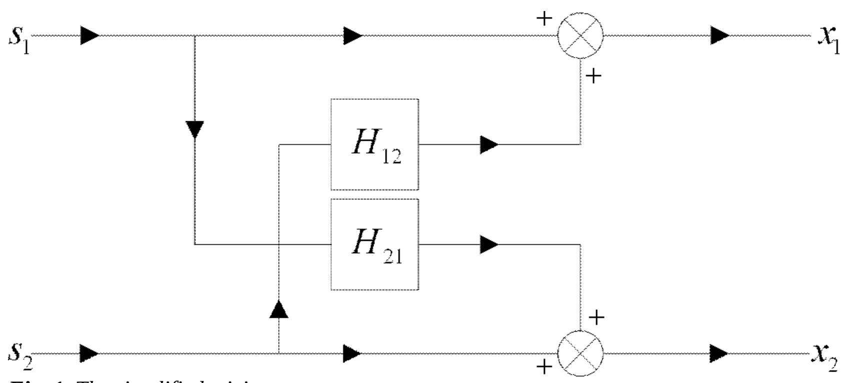


Fig. 1. *The simplified mixing system*

t denotes the time index,

$$\mathbf{G}(t) = \left[G^0(t) \quad G^1(t) \quad \ldots \quad G^{n-1}(t) \right]^T$$

is the mixing matrix with n taps (the superscript number indicates the tap index), and

$$\begin{aligned}\tilde{\mathbf{S}}(t) \quad &= \quad \left[\tilde{s}_1(t) \quad \tilde{s}_2(t) \quad \tilde{s}_1(t-1) \quad \tilde{s}_2(t-1) \right. \\ &\qquad \left. \ldots \quad \tilde{s}_1(t-n+1) \quad \tilde{s}_2(t-n+1) \right]\end{aligned}$$

However, because we are more interested in the separation of the signals rather than the deconvolution of them, we take the assumption that the channels between each source and the closest microphone are simply the Kronecker delta function. This simplifies the problem because it means that we only have to account for two unknown filters rather than four. Fig. 1 shows the simplified mixing system.

On the other hand, this means that at best, we will separate the signals only up to filtered versions of the original. In order to find the original unfiltered versions of the sources, blind dereverberation is needed. This is a very difficult problem when only given one instance of the filtered speech; temporal whitening is not recommended since pure speech is naturally temporally correlated (Douglas 2003), and temporal whitening would make the speech sound unnatural. On the other hand, temporal decorrelation may have its uses in applications where listening to the signal is not needed - e.g. automatic speech recognition.

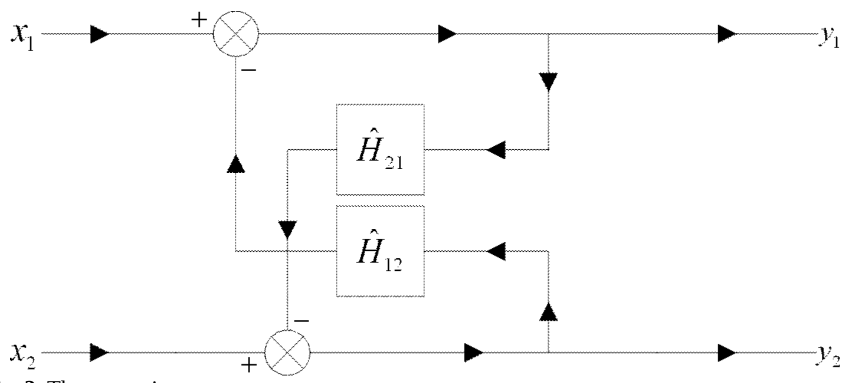


Fig. 2. The separating system

The Cross-talk Resistant Adaptive Noise Canceller

Zinser et al. proposes an adaptation to the LMS filter in order to make it more resilient to cross-talk. Rather than feeding the noise estimate directly into the LMS filter as a reference, he describes how a second LMS filter can be used to remove any of the crosstalk from the noise estimate, resulting in a better noise approximation (Zinser et al. 1985). Fig. 2 shows a block diagram of the backward-separation system. It can be seen that as $\hat{H}_{12}$ and $\hat{H}_{21}$ converge to H_{12} and H_{21} respectively, y_1 and y_2 will converge to s_1 and s_2 respectively. Note that permutation of the order of inputs to outputs cannot occur. This permutation occurs where there is no guarantee that any specific source will be mapped to a specific output. The inability to permute differentiates the CTRANC from other methods of blind separation (such as independent component analysis (Comon 1995)), which is based purely on the independence of the outputs. However, this is based on the assumption that each microphone is the closest microphone to a unique source.

Mei and Yin expand on this idea to derive the following simplified equation updates for the filters $\hat{H}_{12}$ and $\hat{H}_{21}$ (Mei and Yin 2004).

$$\hat{H}_{12}(t+1) = \hat{H}_{12}(t) - \mu_1 y_1(t)\mathbf{Y}_2(t)$$
$$\hat{H}_{21}(t+1) = \hat{H}_{21}(t) - \mu_2 y_2(t)\mathbf{Y}_1(t)$$

where μ_1 and μ_2 are the positive learning rates, $y_1(t)$ and $y_2(t)$ are the estimates of the separated signals at time t, and

$$\mathbf{Y}_1(t) = \left[y_1(t-1), \quad y_1(t-2), \quad \ldots, \quad y_1(t-n)\right]^T$$
$$\mathbf{Y}_2(t) = \left[y_2(t-1), \quad y_2(t-2), \quad \ldots, \quad y_2(t-n)\right]^T$$

Vector-LMS

While ordinary LMS will find the transversal filter weights when given both the input and output of a filter, vector-LMS will find the mixing system given the inputs and outputs of the mixing system. For example, if we applied vector-LMS to two-input two-output system shown in equation (1), the matrix-polynomial of the filter would converge to $\mathbf{G}$. Batra and Barry show the derivation of the vector LMS algorithm

$$\widehat{\mathbf{G}}(t+1) = \widehat{\mathbf{G}}(t) + \mu \mathbf{S}(t)\mathbf{e}(t)^T$$

where $\widehat{\mathbf{G}}(t)$ is the estimate at time t of the mixing polynomial matrix $\mathbf{G}$, μ is the step size, $\mathbf{S}(t) = \left[\mathbf{s}^T(t),\quad \mathbf{s}^T(t-1),\quad \ldots,\quad \mathbf{s}^T(t-n)\right]^T$ is a vector of length $2n$ of the inputs where n is the filter order, and $\mathbf{e}(t)$ is a length-2 vector of the errors between the desired filter output $\mathbf{x}$ and its actual output $\hat{\mathbf{x}}$ where $\hat{\mathbf{x}} = \mathbf{G}^T\mathbf{S}$ (Batra and Barry 1995).

In this paper, we develop a crosstalk resistant adaptive noise canceller that utilizes vector-LMS to obtain a multibranched-recursive structure, creating a more modular algorithm with increased computational efficiency.

3 The Cross-coupled Vector-LMS

In order to show the working of the CTRANC based on vector-LMS, we will consider the situation of four inputs and four outputs. In Fig. 3 we have a matrix polynomial representation of the mixing system, where $\tilde{\mathbf{s}}_1 = \left[\tilde{s}_1,\quad \tilde{s}_2\right]^T$ and $\tilde{\mathbf{s}}_2 = \left[\tilde{s}_3,\quad \tilde{s}_4\right]^T$ are the four inputs multiplexed into two vectors, $\mathbf{x}_1 = \left[x_1,\quad x_2\right]^T$ and $\mathbf{x}_2 = \left[x_3,\quad x_4\right]^T$ are the four outputs multiplexed into two vectors, and $\mathbf{G}_{11}$, $\mathbf{G}_{12}$, $\mathbf{G}_{21}$, and $\mathbf{G}_{22}$ are all mixing polynomial matrices representing the entire mixing system. Note that these should not be confused with their scalar counterparts. Using the same reasoning as with the ordinary CTRANC, we derive the following update equations for the separating polynomial matrices $\widehat{\mathbf{H}}_{12}$ and $\widehat{\mathbf{H}}_{21}$.

$$\widehat{\mathbf{H}}_{12}(t+1) = \widehat{\mathbf{H}}_{12}(t+1) + \mu_1 \mathbf{Y}_2(t)\mathbf{y}_1^T(t)$$

$$\widehat{\mathbf{H}}_{21}(t+1) = \widehat{\mathbf{H}}_{21}(t+1) + \mu_2 \mathbf{Y}_1(t)\mathbf{y}_2^T(t)$$

where μ_1 and μ_2 are convergence weights, $\mathbf{y}_1$ and $\mathbf{y}_2$ are the length-2 output

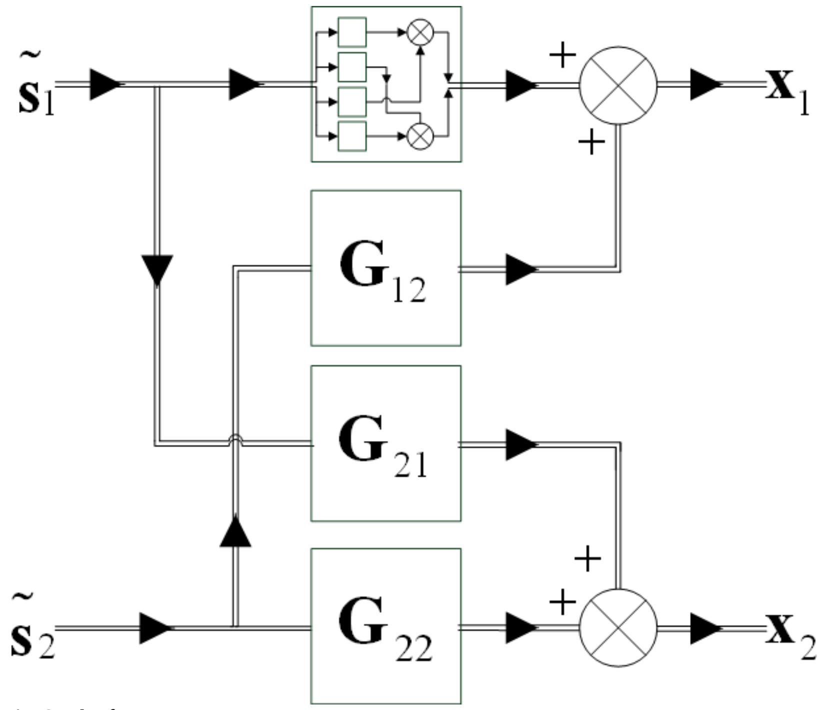


Fig. 3. The four input mixing system

vectors $[y_1, \quad y_2]$ and $[y_3, \quad y_4]$ respectively, and the length-$2n$ vectors $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are defined by

$$\mathbf{Y}_1 = \left[\mathbf{y}_1^T(t-1), \quad \mathbf{y}_1^T(t-2), \quad \ldots, \quad \mathbf{y}_1^T(t-n)\right]^T$$

$$\mathbf{Y}_2 = \left[\mathbf{y}_2^T(t-1), \quad \mathbf{y}_2^T(t-2), \quad \ldots, \quad \mathbf{y}_2^T(t-n)\right]^T$$

Essentially what this algorithm will do is separate a system of four mixed sources into two systems of two mixed sources. One can then apply the algorithm from an ordinary CTRANC to separate each of the sources into approximations of the original individual signals.

Number of Inputs	Proposed Method		Mei and Yin Method
	Separate all	Extract one	
4	$24n+32$	$20n+26$	$24n+36$
8	$112n+136$	$84n+98$	$112n+168$
16	$480n+544$	$340n+370$	$480n+720$

Table 1. Multiplication operations required.

Number of Inputs	Proposed Method		Mei and Yin Method
	To separate all	To extract one	
4	$12n+20$	$10n+16$	$12n+24$
8	$56n+80$	$42n+56$	$56n+112$
16	$240n+304$	$170n+200$	$240n+480$

Table 2. Addition/subtraction operations required.

Computational Efficiency

Mei and Yin (2004) proposed an adaptation to the TITO CTRANC that extended it for use with more than two input signals. This was simply an extension of the two-channel case. For example, with three sources each input needed two LMS filters removing the crosstalk from the other two channels. Thus the computational complexity of their algorithm was equivalent to $L(L-1)$ LMS algorithms. The multibranched recursive approach that we propose is more efficient under certain conditions as will now be shown.

We will now consider the computational requirements for the proposed algorithm. With $L=2^k$ inputs, it requires two 2^{k-1}-vector LMS algorithms, four 2^{k-2}-vector LMS algorithms, etc. The number of multiplication and addition/subtraction operations for each vector LMS algorithm is given by the following equations.

$$2(n+1)M^2+M \qquad \text{multiplications}$$
$$(n+1)M^2+M \qquad \text{additions/subtractions}$$

where n is the filter size and M is the size of the input/output vectors. These equations also work for scalar LMS, when $M=1$.

Another advantage in the proposed method is that its modular structure allows the removal of portions that may be unnecessary. For example, in an eight-input system, if only one source needs to be extracted, and it is known which output channel that source maps to, then six scalar LMS and two 2-vector LMS algorithms can be discarded. This allows for further computational savings.

Tables 1 and 2 show the multiplication and addition/subtraction requirements for 4, 8, and 16 input systems for the cases where all sources need to be extracted, and when only one needs to be extracted. These results show that the current method is more computationally efficient than that proposed by Mei and Yin for all given cases.

4 Separation Performance

We conducted a simple experiment to discover the relative separation of the proposed method to the method in (Mei and Yin 2004).

Experimental Procedure

The experiment was set up as follows: four microphones were placed as four corners of a 0.2m $\times$ 0.2m square near the middle of a 4m $\times$ 7m room furnished with a lounge suite, a piano and a dining room suite. There were three noise sources, all samples of a car assembly line from the file labeled 'factory floor noise 2' from the NOISEX database. The speech was created by using a loudspeaker playing the speech sample in the package 'Lunatick-20080326–cc.tgz' from the VoxForge speech corpus. The algorithm was implemented using NI LabVIEW.

Using the described set-up, we used the proposed algorithm to reduce the noise level. Each filter had 1000 tap-weights. We chose this number because increasing the number of tap weights beyond 1000 increased computational complexity with a negligible increase in SNR, while decreasing the number of tap-weights adversely affected the results. Because we do not have the power of the desired signal by itself, to calculate the SNR, we net to use the following formula

$$SNR = 10\log_{10}\left(\frac{P_{SN} - P_N}{P_N}\right) \tag{2}$$

where P_{SN} is the combined power of the speech with the noise and P_N is the power of the noise. This is based on the assumption that the noise and the speech are statistically independent.

Results

Using the formula for calculating signal-to-noise ratios given in equation (2), we obtained the results as shown in Table 3. In an informal listening test, we also found that the speech was more comprehensible in the separated signals than in the mixed signals.

	Input SNR	Output SNR
Proposed Method	7.7 dB	14.2 dB
Mei and Yin Method	7.7 dB	13.9 dB

Table 3. Increases in SNR

There is a modest gain in the SNR for the proposed method when comparing it to the method described by Mei and Yin. This indicates that the proposed method can perform separation at least as well as the method proposed by Mei and Yin, while saving in computational complexity.

5 Conclusion

One solution to the blind source problem is to use a cross-talk resistant noise canceller to separate the signals. This paper describes an adaptation to the CTRANC algorithm to increase its computational efficiency. Experimental data shows that there is a modest increase in performance due to these adaptations. It also has the advantage that it is potentially even more computationally efficient if there is only one desired source, and it is known which channel it will be separated to. In future studies we propose to incorporate this method with an automatic speech recognition system, and evaluate its performance in that capacity.

6 References

L. Zhang and A. Cichocki, (2000) Blind deconvolution of dynamical systems: A state space approach," Journal of Signal Processing, vol. 4, no. 2, pp. 111-130.

M. S. Pedersen, J. Larsen, U. Kjems, and L. C. Parra, (2007) "A survey of convolutive blind source separation methods," Multichannel Speech Processing Handbook.

S. Amari and A. Cichocki, (1998) "Adaptive blind signal processing-neural network approaches," Proceedings of the IEEE, vol. 86, no. 10, pp. 2026– 2048.

S. V. Gerven and D. V. Compernolle, (1995) "Signal separation by symmetric adaptive decorrelation: stability, convergence, and uniqueness," IEEE Transactions on Signal Processing, vol. 43, no. 7, pp. 1602–1612.

S. Douglas, (2003) "Convolutive blind separation of speech mixtures using the natural gradient," Speech Communication, vol. 39, no. 1-2, pp. 65–78.

R. Zinser, G. Mirchandani, and J. Evans (1985) "Some experimental and theoretical results using a new adaptive filter structure for noise cancellation in the presence of crosstalk," in Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85., vol. 10, 1985, pp. 1253– 1256.

P. Comon, (1994)"Independent component analysis, a new concept?" Signal processing, vol. 36, no. 3, p. 287-314.

T. Mei and F. Yin, (2004) "Blind separation of convolutive mixtures by decorrelation," Signal Processing, vol. 84, no. 12, pp. 2297–2313.

A. Batra and J. Barry, (1995) "Blind cancellation of co-channel interference," in Proceedings of GLOBECOM '95, Singapore, 1995, pp. 157–162

REFERENCES

[1] H. Buchner, R. Aichner, and W. Kellermann. TRINICON: a versatile framework for multichannel blind signal processing. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 3, page iii889, 2004.

[2] B. Widrow, J.R. Glover, J.M. McCool, J. Kaunitz, C.S. Williams, R.H. Hearn, J.R. Zeidler, Jr. Eugene Dong, and R.C. Goodlin. Adaptive noise cancelling: Principles and applications. *Proceedings of the IEEE*, 63(12):1692–1716, 1975.

[3] J.M. Cioffi and T. Kailath. Fast, recursive-least-squares transversal filters for adaptive filtering. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 32(2):304–337, 1984.

[4] William G Gardner. Efficient convolution without input/output delay. In *Audio Engineering Society Convention 97*. Audio Engineering Society, 1994.

[5] M. Miyoshi and Y. Kaneda. Inverse filtering of room acoustics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(2):145–152, February 1988.

[6] J. F Cardoso and B. H Laheld. Equivariant adaptive source separation. *IEEE Transactions on signal processing*, 44(12):30173030, 1996.

[7] S. Amari, S.C. Douglas, A. Cichocki, and H.H. Yang. Multichannel blind deconvolution and equalization using the natural gradient. In *Signal Processing Advances in Wireless*

Communications, 1997 First IEEE Signal Processing Workshop on, pages 101–104, 1997.

[8] S. Amari and A. Cichocki. Adaptive blind signal processing-neural network approaches. *Proceedings of the IEEE*, 86(10):2026–2048, October 1998.

[9] M. S Pedersen, J. Larsen, U. Kjems, and L. C Parra. A survey of convolutive blind source separation methods. *Multichannel Speech Processing Handbook*, 2007.

[10] Alexander Ypma, Amir Leshem, and Robert P.W. Duin. Blind separation of rotating machine sources: bilinear forms and convolutive mixtures. *Neurocomputing*, 49(1-4):349–368, December 2002.

[11] Christian Jutten and Jeanny Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, July 1991.

[12] TJ Moir. A z-domain transfer function solution to the non-minimum phase acoustic beamformer. *International journal of systems science*, 38(7):563–575, 2007.

[13] S. Van Gerven and D. Van Compernolle. Signal separation by symmetric adaptive decorrelation: stability, convergence, and uniqueness. *IEEE Transactions on Signal Processing*, 43(7):1602–1612, July 1995.

[14] M Girolami. A nonlinear model of the binaural cocktail party effect. *Neurocomputing*, 22(1-3):201–215, November 1998.

[15] V.J. Mathews and Zhenhua Xie. A stochastic gradient adaptive filter with gradient adaptive step size. *Signal Processing, IEEE Transactions on*, 41(6):2075–2087, 1993.

[16] S. Makino, Y. Kaneda, and N. Koizumi. Exponentially weighted stepsize nlms adaptive filter based on the statistics of a room impulse response. *Speech and Audio Processing, IEEE Transactions on*, 1(1):101–108, 1993.

[17] S. Kalluri and G.R. Arce. A general class of nonlinear normalized adaptive filtering algorithms. *IEEE Transactions on Signal Processing*, 47(8):2262–2272, August 1999.

[18] A. Batra and J.R. Barry. Blind cancellation of co-channel interference. In *Proceedings of GLOBECOM '95*, pages 157–162, Singapore, 1995.

[19] J. Benesty, F. Amand, A. Gilloire, and Y. Grenier. Adaptive filtering algorithms for stereophonic acoustic echo cancellation. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, pages 3099–3102, Detroit, MI, USA, 1995.

[20] S.C. Douglas. Analysis of the multiple-error and block least-mean-square adaptive algorithms. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 42(2):92–101, 1995.

[21] M. Dentino, J. McCool, and B. Widrow. Adaptive filtering in the frequency domain. *Proceedings of the IEEE*, 66(12):1658– 1659, December 1978.

[22] E. Ferrara. Fast implementations of LMS adaptive filters. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):474–475, 1980.

[23] J.J. Shynk. Frequency-domain and multirate adaptive filtering. *Signal Processing Magazine, IEEE*, 9(1):14 –37, January 1992.

[24] G.A. Clark, S.K. Mitra, and S. Parker. Block implementation of adaptive digital filters. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 29(3):744–752, 1981.

[25] A. Feuer. Performance analysis of the block least mean square algorithm. *Circuits and Systems, IEEE Transactions on*, 32(9):960–963, 1985.

[26] R. Zinser, G. Mirchandani, and J. Evans. Some experimental and theoretical results using a new adaptive filter structure for noise cancellation in the presence of crosstalk. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, volume 10, pages 1253–1256, 1985.

[27] E. Vincent, R. Gribonval, and C. Fvotte. Performance measurement in blind audio source separation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(4):14621469, 2006.

[28] Emmanuel Vincent, Rémi Gribonval, and Mark D Plumbley. Oracle estimators for the benchmarking of source separation algorithms. *Signal Processing*, 87(8):1933–1950, 2007.

[29] Valentin Emiya, Emmanuel Vincent, Niklas Harlander, and Volker Hohmann. Subjective and objective quality assessment of audio source separation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(7):2046–2057, 2011.

[30] Jedrzej Kociński, Szymon Drgas, and Edward Ozimek. Evaluation of blind source separation for different algorithms based on second order statistics and different spatial configurations of directional microphones. *Applied Acoustics*, 73(2):109–116, 2012.

[31] Josef Kornycky, Banu Gunel, and Ahmet Kondoz. Comparison of subjective and objective evaluation methods for audio source separation. In *Proceedings of Meetings on Acoustics*, volume 4, page 050001. Acoustical Society of America, 2008.

[32] Jon Barker, Emmanuel Vincent, Ning Ma, Heidi Christensen, and Phil Green. The pascal chime speech separation and recognition challenge. *Computer Speech & Language*, 27(3):621–633, 2013.

[33] Emmanuel Vincent, Shoko Araki, and Pau Bofill. The 2008 signal separation evaluation

campaign: A community-based approach to large-scale evaluation. In *Independent Component Analysis and Signal Separation*, pages 734–741. Springer, 2009.

[34] S. Gorlow and S. Marchand. Informed audio source separation using linearly constrained spatial filters. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(1):3–13, January 2013.

[35] Marc Delcroix, Takafumi Hikichi, and Masato Miyoshi. Dereverberation and denoising using multichannel linear prediction. *IEEE Transactions on Audio, Speech and Language Processing*, 15(6):1791–1801, August 2007.

[36] M. Joho and P. Schniter. Frequency domain realization of a multichannel blind deconvolution algorithm based on the natural gradient. In *Proc. ICA2003*, page 543548, 2003.

[37] Hefa Zhang, Liping Li, and Wanchun Li. Independent vector analysis for convolutive blind noncircular source separation. *Signal Processing*, 92(9):2275–2283, 2012.

[38] Syed A Jafar. Blind interference alignment. *Selected Topics in Signal Processing, IEEE Journal of*, 6(3):216–227, 2012.

[39] Feng Youqian, Zhang Shanwen, and Li Zhengchao. A method of radar signal separation. In *2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007)*, pages 846–849, Harbin, Heilongjiang, China, December 2007.

[40] L. Zhang and A. Cichocki. Blind deconvolution of dynamical systems: A state space approach. *Journal of Signal Processing*, 4(2):111130, 2000.

[41] Tiemin Mei, Alfred Mertins, Fuliang Yin, Jiangtao Xi, and Joe F. Chicharo. Blind

source separation for convolutive mixtures based on the joint diagonalization of power spectral density matrices. *Signal Processing*, 88(8):1990–2007, August 2008.

[42] K. Torkkola. Blind separation of convolved sources based on information maximization. In *IEEE Workshop on Neural Networks for Signal Processing*, page 423432, September 1996.

[43] J.-F. Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, October 1998.

[44] T Mei and F Yin. Blind separation of convolutive mixtures by decorrelation. *Signal Processing*, 84(12):2297–2313, December 2004.

[45] T.-W. Lee, A.J. Bell, and R. Orglmeister. Blind source separation of real world signals. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, pages 2129–2134, Houston, TX, USA, 1997.

[46] L. Parra and C. Spence. Convolutive blind separation of non-stationary sources. *IEEE Transactions on Speech and Audio Processing*, 8(3):320–327, May 2000.

[47] Matthew Anderson, Xi-Lin Li, and Tlay Adal. Complex-valued independent vector analysis: Application to multivariate gaussian model. *Signal Processing*, 92(8):1821–1831, August 2012.

[48] Matthew Anderson, Tuelay Adali, and Xi-Lin Li. Joint blind source separation with multivariate gaussian model: algorithms and performance analysis. *Signal Processing, IEEE Transactions on*, 60(4):16721683, 2012.

[49] Nicolle Correa, Tülay Adalı, and Vince D Calhoun. Performance of blind source separation algorithms for fmri analysis using a group ica method. *Magnetic resonance imaging*, 25(5):684–694, 2007.

[50] Xi-Lin Li, Tülay Adalı, and Matthew Anderson. Joint blind source separation by generalized joint diagonalization of cumulant matrices. *Signal Processing*, 91(10):2314–2322, 2011.

[51] Seungjin Choi, Andrzej Cichocki, Hyung-Min Park, and Soo-Young Lee. Blind source separation and independent component analysis: A review. *Neural Information Processing-Letters and Reviews*, 6(1):1–57, 2005.

[52] Zhenwei Shi, Hongjuan Zhang, and Zhiguo Jiang. Hybrid linear and nonlinear complexity pursuit for blind source separation. *Journal of Computational and Applied Mathematics*, 236(14):3434–3444, 2012.

[53] Yuanqing Li, S-I Amari, Andrzej Cichocki, Daniel WC Ho, and Shengli Xie. Underdetermined blind source separation based on sparse representation. *Signal Processing, IEEE Transactions on*, 54(2):423–437, 2006.

[54] P. Comon, C. Jutten, and J. Herault. Blind separation of sources, part II: problems statement. *Signal processing*, 24(1):1120, 1991.

[55] Jerome Antoni and S Chauhan. A study and extension of second-order blind source separation to operational modal analysis. *Journal of Sound and Vibration*, 332(4):1079–1106, 2013.

[56] Ingo P Waldmann, Giovanna Tinetti, Pieter Deroo, Morgan DJ Hollis, Sergey N Yurchenko, and Jonathan Tennyson. Blind extraction of an exoplanetary spectrum through independent component analysis. *The Astrophysical Journal*, 766(1):7, 2013.

[57] Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics Express*, 18(10):10762–10774, 2010.

[58] T. Yousefi Rezaii and M. Geravanchizadeh. Robust multi-channel crosstalk resistant noise cancellation based on SAD structure and LMMN algorithm. In *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, pages 1047–1052, 2008.

[59] Sen M. Kuo and Wei M. Peng. Principle and applications of asymmetric crosstalk-resistant adaptive noise canceler. *Journal of the Franklin Institute*, 337(1):57–71, January 2000.

[60] N. Mitianoudis and M.E. Davies. Audio source separation of convolutive mixtures. *IEEE Transactions on Speech and Audio Processing*, 11(5):489–497, September 2003.

[61] K. Rahbar and J.P. Reilly. Blind source separation of convolved sources by joint approximate diagonalization of cross-spectral density matrices. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, pages 2745–2748, Salt Lake City, UT, USA, 2001.

[62] S Douglas. Convolutive blind separation of speech mixtures using the natural gradient. *Speech Communication*, 39(1-2):65–78, January 2003.

[63] Y. Liang, F. Cong, and I. Hagiwara. Multistage blind source separation and deconvolution for convolutive mixture of speech signals. *International Journal of Computational Intelligence Research*, 3(1):5559, 2007.

[64] R. Aichner, H. Buchner, S. Araki, and S. Makino. On-line time-domain blind source separation of nonstationary convolved signals. In *Proc. Int. Symposium on Independent Component Analysis and Blind Signal Separation*, 2003.

[65] H. Sahlin and H. Broman. Separation of real-world signals. *Signal processing*, 64(1):103113, 1998.

[66] S. Gannot, D. Burshtein, and E. Weinstein. Signal enhancement using beamforming and nonstationarity with applications to speech. *IEEE Transactions on Signal Processing*, 49(8):16141626, 2001.

[67] Yiteng Huang, J. Benesty, and Jingdong Chen. A blind channel identification-based two-stage approach to separation and dereverberation of speech signals in a reverberant environment. *IEEE Transactions on Speech and Audio Processing*, 13(5):882–895, September 2005.

[68] S. Gudvangen and S.J. Flockton. Comparison of pole-zero and all-zero modelling of acoustic transfer functions. *Electronics Letters*, 28(21):1976, 1992.

[69] M. Wolfel. Enhanced speech features by single-channel joint compensation of noise and reverberation. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(2):312–323, February 2009.

[70] Tomohiro Nakatani, Keisuke Kinoshita, and Masato Miyoshi. Harmonicity-based blind dereverberation for single-channel speech signals. *IEEE Transactions on Audio, Speech and Language Processing*, 15(1):80–95, January 2007.

[71] M.Z. Ikram and D.R. Morgan. Exploring permutation inconsistency in blind separation of speech signals in a reverberant environment. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, pages II1041–II1044, Istanbul, Turkey, 2000.

[72] H. Buchner, R. Aichner, and W. Kellermann. A generalization of blind source separation algorithms for convolutive mixtures based on second-order statistics. *IEEE Transactions on Speech and Audio Processing*, 13(1):120–134, January 2005.

[73] L. Lepauloux, P. Scalart, and C. Marro. An efficient low-complexity algorithm for

crosstalk-resistant adaptive noise canceller. In *European Signal Processing Conference*, page 204208, 2009.

[74] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1066–1074, 2007.

[75] Ravish Mehra, Nikunj Raghuvanshi, Lauri Savioja, Ming C Lin, and Dinesh Manocha. An efficient gpu-based time domain solver for the acoustic wave equation. *Applied Acoustics*, 73(2):83–94, 2012.

[76] T. Kim, T. Eltoft, and T. W Lee. Independent vector analysis: An extension of ICA to multivariate components. *Independent Component Analysis and Blind Signal Separation*, page 165172, 2006.

[77] Hiroshi Sawada, Shoko Araki, and Shoji Makino. Underdetermined convolutive blind source separation via frequency bin-wise clustering and permutation alignment. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(3):516–527, 2011.

[78] Zbynek Koldovsky and Petr Tichavsky. Time-domain blind separation of audio sources on the basis of a complete ica decomposition of an observation space. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(2):406–416, 2011.

[79] H. Sawada, R. Mukai, S. Araki, and S. Makino. A robust and precise method for solving the permutation problem of frequency-domain blind source separation. *IEEE Transactions on Speech and Audio Processing*, 12(5):530–538, September 2004.

[80] T. Kim, H. T Attias, S. Y Lee, and T. W Lee. Blind source separation exploiting higher-order frequency dependencies. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(1):7079, 2007.

[81] S.C. Douglas, H. Sawada, and S. Makino. Natural gradient multichannel blind deconvolution and speech separation using causal FIR filters. *Speech and Audio Processing, IEEE Transactions on*, 13(1):92–104, 2005.

[82] B.D. Van Veen and K.M. Buckley. Beamforming: a versatile approach to spatial filtering. *IEEE ASSP Magazine*, 5(2):4–24, April 1988.

[83] Yu Takahashi, Tomoya Takatani, Keiichi Osako, Hiroshi Saruwatari, and Kiyohiro Shikano. Blind spatial subtraction array for speech enhancement in noisy environment. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):650–664, May 2009.

[84] P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287314, 1994.

[85] Jayaraman J Thiagarajan, Karthikeyan Natesan Ramamurthy, and Andreas Spanias. Mixing matrix estimation using discriminative clustering for blind source separation. *Digital Signal Processing*, 23(1):9–18, 2013.

[86] Hosein Mohimani, Massoud Babaie-Zadeh, and Christian Jutten. A fast approach for overcomplete sparse decomposition based on smoothed norm. *Signal Processing, IEEE Transactions on*, 57(1):289–301, 2009.

[87] E. Weinstein, M. Feder, and A.V. Oppenheim. Multi-channel signal separation by decorrelation. *IEEE Transactions on Speech and Audio Processing*, 1(4):405–413, October 1993.

[88] Lang Tong, R-W Liu, Victor C Soon, and Y-F Huang. Indeterminacy and identifiability of blind identification. *Circuits and Systems, IEEE Transactions on*, 38(5):499–509, 1991.

[89] S. Choi and A. Cichocki. Blind separation of nonstationary sources in noisy mixtures. *Electronics Letters*, 36(9):848 –849, April 2000.

[90] G. Mirchandani, R.L. Zinser, and J.B. Evans. A new adaptive noise cancellation scheme in the presence of crosstalk [speech signals]. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 39(10):681–694, 1992.

[91] Shun-ichi Amari, Tian-ping Chen, and Andrzej Cichocki. Stability analysis of learning algorithms for blind source separation. *Neural Networks*, 10(8):1345–1351, November 1997.

[92] Dinh Tuan Pham and P. Garat. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Transactions on Signal Processing*, 45(7):1712–1725, July 1997.

[93] Shun-ichi Amari. Natural gradient works efficently in learning. *Neural Computation*, 10(2):251, February 1998.

[94] S. Amari and S.C. Douglas. Why natural gradient? In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1213–1216 vol.2, 1998.

[95] S. Sundaralingam and K. C. Sharman. Genetic evolution of adaptive filters. In *Proceedings of DSP, London UK*, page 4753, 1997.

[96] Ying Tan and Jun Wang. Nonlinear blind source separation using higher order statistics and a genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 5(6):600–612, December 2001.

[97] M.Z. Ikram and D.R. Morgan. Permutation inconsistency in blind speech separa-

tion: investigation and solutions. *IEEE Transactions on Speech and Audio Processing*, 13(1):1–13, January 2005.

[98] Takashi Itahashi and Kiyotoshi Matsuoka. Stability of independent vector analysis. *Signal Processing*, 92(8):1809–1820, 2012.

[99] U.A. Lindgren and H. Broman. Source separation using a criterion based on second-order statistics. *Signal Processing, IEEE Transactions on*, 46(7):1837 –1850, July 1998.

[100] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. pages 123–134, 1960.

[101] B. Widrow, J. M McCool, M. G Larimore, and C. R Johnson Jr. Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings of the IEEE*, 64(8):11511162, 1976.

[102] G. D Bergland. Numerical analysis: A fast fourier transform algorithm for real-valued series. *Communications of the ACM*, 11(10):703710, 1968.

[103] J.C. Cox. The maximum tolerable delay of speech and music. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, volume 10, pages 612–615, 1985.

[104] S Araki S Makino H Sawada, R Mukai. Bss demo. `http://www.kecl.ntt.co.jp/icl/signal/sawada/demo/bss2to4/index.html`, April 2003.

[105] T. J. Moir and J. I. Harris. Decorrelation of multiple non-stationary sources using a multivariable crosstalk-resistant adaptive noise canceller. *International Journal of Adaptive Control and Signal Processing*, 27(5):349367, 2013.

[106] T. Moir, F. Alam, and J. Harris. Blind source separation for adaptive speech control. In *The eighteenth annual Conference on Mechatronics and Machine Vision in Practice, Brisbane, Australia, 2011-12-06 - 2011-12-08*. AUT University, 2011.