

TIME-DOMAIN BLIND SOURCE SEPARATION FOR CONVOLUTIVE
MIXTURES USING SECOND-ORDER STATISTICS

by

CEMİL DEMİR

BSc, Electrical Electronics Engineering, Bilkent University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2007

ACKNOWLEDGEMENTS

I am grateful to Asst. Prof. F. Kerem Harmancı for not only his help but also his friendship. Whenever I faced with problems during the preparation of this thesis, he helped me without hesitation.

I am also thankful to the members of the committee...

ABSTRACT

TIME-DOMAIN BLIND SOURCE SEPARATION FOR CONVOLUTIVE MIXTURES USING SECOND-ORDER STATISTICS

Blind Source Separation (BSS) methods are generally used to separate the speech of people in the same room by using audio data recorded by multi-microphone system. This thesis focuses on a time-domain technique in BSS. In particular, the method presented uses the statistical independence of sources, the non-stationarity and the non-whiteness of speech signals. The statistical independence of sources implies that cross-correlations of demixed signals over all time-lags are null. Here, non-whiteness property is exploited by simultaneous minimization of output cross-correlations over multiple time-lags and non-stationarity property is exploited by simultaneous minimization of output cross-correlations at different time-instants. In the method, the demixing filter coefficients are obtained using an iterative optimization method and cost function. The coefficients are updated via the Natural Gradient adaptation method and the method is implemented in a block-online manner. Using Sylvester structure of the demixing filter matrix and with some approximations, the method is implemented efficiently.

The method is found to be successful in simulations that are done using synthetically mixed signals in the computer. Moreover, the method is used to separate the noise from the speech signals that are also mixed synthetically and this suggests that the method can be used as a front-end processor for speech recognition applications.

ÖZET

EVİRİŞİMSEL KARIŞIMLAR İÇİN ZAMAN ALANINDA İKİNCİ DERECEDEDEN İSTATİSTİKLERLE GÖZÜ KAPALI KAYNAK AYRIŞTIRMASI

Gözü Kapalı Kaynak Ayrıştırma (GKKA) yöntemleri genellikle aynı anda birden fazla kişinin konuştuğu durumlarda konuşmaların birden fazla mikrofondan alınarak, konuşmacıların konuşmalarının birbirinden ayrıştırılması için kullanılmaktadır. Bu çalışma zaman alanında GKKA yöntemi üzerine odaklanmıştır. Uygulanan yöntem, kaynak sinyallerinin istatistiksel olarak bağımsız olma, durağan olmama ve beyaz olmama özelliklerini kullanmaktadır. Kaynakların istatistiksel bağımsız olması ayrıştırılmış sinyallerin çapraz ilintilerinin tüm zaman ötelemelerinde sıfır olmasını gerektirir. Beyaz olmama özelliği çıktı sinyallerinin çapraz ilintilerinin birden fazla zaman ötelemeleğinde eş zamanlı olarak en küçüklenmesiyle, durağan olmama özelliği ise çıktı sinyallerinin çapraz ilintilerinin eş zamanlı olarak birden fazla zaman aralığında en küçüklenmesiyle kullanılmaktadır. Kullanılan yöntemde, ayrıştırıcı süzgeç katsayıları tanımlanan maliyet fonksiyonunun döngülü eniyileme yöntemi kullanılarak güncellenmesiyle elde edilmiştir. Katsayıların güncellenmesinde Doğal Gradyan yönteminden yararlanılmıştır ve katsayıların güncellenmesi parçalı gerçek zamanlı olarak gerçekleşmiştir. Ayrıştırıcı matrislerin Sylvester yapısında olma özelliği ve bazı yaklaşıklamalar kullanılarak yöntem verimli bir şekilde gerçekleşmiştir.

Sentetik olarak karıştırılmış sinyallerde anlatılan yöntemin başarılı olduğu saptanmıştır. Aynı zamanda bu yöntem sentetik olarak gürültülendirilmiş sinyallerdeki gürültünün azaltılmasında kullanılmıştır. Bu başarımlar kullanılan yöntemin konuşma tanıma motorlarında bir ön-uç işlemcisi gibi kullanılabileceğini önermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	viii
LIST OF TABLES	x
LIST OF SYMBOLS/ABBREVIATIONS	xi
1. INTRODUCTION	1
1.1. Background and Motivation	1
1.2. Literature Review	2
1.3. Thesis Outline	4
2. METHODOLOGY	6
2.1. Model Description	6
2.1.1. Instantaneous Mixing System	6
2.1.2. Convolutional Mixing System	7
2.2. Block Time-Domain BSS Algorithm	9
2.2.1. Matrix Formulation For Convolutional Mixtures	9
2.2.2. Cost Function and Algorithm Derivation	15
2.2.3. Independence, Non-stationarity and Non-whiteness Properties	18
2.3. Gradient Approaches to the Update of Demixing Filter Coefficient	19
2.3.1. Standard Gradient Approach	20
2.3.2. Natural Gradient Approach	22
3. IMPLEMENTATION OF THE METHOD	23
3.1. Implementation of the Algorithm Using Different Weighting Functions	23
3.1.1. Off-line Implementation	23
3.1.2. On-line Implementation	24
3.1.3. Block-Online Implementation	25
3.2. Approximated Version and Efficient Implementation of the Algorithm	27
3.2.1. Estimation of Correlation Matrices using Correlation Method	27
3.2.2. Approximation of the Normalization and Regularization Strategy	29

3.2.3. Efficient Implementation of the Matrix-Matrix Multiplication	30
3.2.4. Appropriate Initialization	31
3.2.5. Stepsize Control	33
4. SEPARATION PERFORMANCE RESULTS	34
4.1. Introduction	34
4.2. Results with Synthetically Mixed Signals	35
4.2.1. Fixed Stepsize Results	35
4.2.2. Adaptive Stepsize Results	37
4.3. Results with signals mixed using Image Method	38
5. APPLICATION: IMPROVEMENT IN SPEECH RECOGNITION PERFORMANCE USING TIME-DOMAIN BSS	47
5.1. Speech Recognition Overview	47
5.2. Speech Recognition In Noisy Environments	48
5.3. BSS and Noise Reduction	49
5.4. Simulation Results	49
6. CONCLUSION AND FUTURE WORK	51
APPENDIX A: Natural Gradient Adaptation Algorithm	52
A.1. Euclidean and Riemannian Geometry	52
A.2. Natural-Gradient Descent	54
A.3. Application of Natural Gradient to Instantaneous Blind Source Separation	56
REFERENCES	57

LIST OF FIGURES

Figure 2.1.	Mixing and Demixing System	8
Figure 2.2.	Correlation Matrices of mixed and demixed signals	17
Figure 2.3.	Correlation Matrix with whiteness	19
Figure 2.4.	Correlation Matrix with nonwhiteness	20
Figure 3.1.	Weighting Function for Off-line Implementation	24
Figure 3.2.	Weighting Function for On-line Implementation	25
Figure 3.3.	Weighting Function for Block On-line Implementation	26
Figure 3.4.	Block-Online Update Method Figure	28
Figure 3.5.	Setups for BSS requiring: (a) only causal delays and (b) causal and noncausal delays for the demixing system W	32
Figure 4.1.	Room Impulse Responses created randomly	35
Figure 4.2.	SIR values for different iteration numbers	36
Figure 4.3.	Source Signals	37
Figure 4.4.	Mixture Signals	38
Figure 4.5.	Separated Signals	39

Figure 4.6.	SIR Values for different forgetting factors	40
Figure 4.7.	SIR Values for different subblock numbers	40
Figure 4.8.	SIR Values for different stepsizes	41
Figure 4.9.	SIR values for different iteration numbers with adaptive stepsize .	41
Figure 4.10.	SIR Values for different forgetting factors with adaptive stepsize .	42
Figure 4.11.	SIR values for different subblock numbers with adaptive stepsize .	42
Figure 4.12.	Speakers and microphones in the room in 3D	43
Figure 4.13.	Speakers and microphones in the room in 2D	43
Figure 4.14.	SIR values for different l values with fixed stepsize	44
Figure 4.15.	SIR values for different l values with adaptive stepsize	44
Figure 4.16.	SIR values for different d values with fixed stepsize	45
Figure 4.17.	SIR values for different d values with adaptive stepsize	45
Figure 4.18.	SIR values for different s values	46
Figure 4.19.	SIR values for different s values for adaptive stepsize	46

LIST OF TABLES

Table 5.1.	Training Databases	48
Table 5.2.	Airplane Noise Error Rates	50
Table 5.3.	Babble Noise Error Rates	50
Table 5.4.	Office Noise Error Rates	50

LIST OF SYMBOLS/ABBREVIATIONS

C	Overall System Matrix
d	Distance between microphones
D	Number of time-lags
h	Mixing filter
H	Mixing Matrix
i	Subblock Index
j	Iteration Number
J	Cost Function
K	Subblock Number
l	Distance between midpoints of the speakers and midpoints of microphones
L	Demixing Filter Length
m	Block Index
M	Mixing Filter Length
N	Length of output block
p	Microphone Index
P	Source Number
q	Source Index
Q	Microphone Number
R	Correlation Matrix
s	Source signal
s	Source signal vector
S	Source Signal Matrix
w	Demixing filter
W	Demixing Matrix
x	Mixed signal
x	Microphone signal vector
X	Mixed Signal matrix
y	Demixed signal

Y	Output Signal Matrix
β	Weighting Function
μ	Stepsize Parameter
γ	Forgetting Factor
Γ	Update Term
ϵ	Rectangular Window
NG	Natural Gradient
SER	Sentence Error Rate
SIR	Signal-to-Interference Ratio
WER	Word Error Rate

1. INTRODUCTION

1.1. Background and Motivation

Blind Source Separation (BSS) consists of recovering unobserved signals or "sources" from several observed mixtures that are obtained at the output of a set of sensors which receives a different combination of the source signals. The adjective "blind" stresses the fact that

1. We do not have observed original signals
2. We do not have any information about the mixing system

The weakness of the priori information is precisely the strength of the BSS model and hence BSS has received considerable attention in the last few years and many algorithms have been proposed [21, 16, 5, 38]. However, the separation of broadband signals in reverberant environments remains a challenging problem. Many researchers have proposed frequency domain algorithms which in general transforms the convolutive mixture problem in the time domain to multiple instantaneous mixtures in frequency domain. However, in [33] it was shown that for long reverberation the separation performance of frequency domain BSS degrades for long FFT frame sizes. The term 'Blind' stress the fact that we do not use any information about the mixing system and source signals that are employed in mixing process. But, of course, we have some assumptions about them. The main assumption about the source signals is that they are statistically independent from each other. The second assumption is about mixing system. Although some initial results are also achieved for the nonlinear mixing models, two linear mixing models are generally used in the community. When each observed mixture is a superposition of scaled independent sources, the mixture is called instantaneous mixture and when each observed mixture is a superposition of filtered independent sources, the mixture is called convolutive mixture. Since instantaneous mixtures can be considered as a subset of convolutive mixtures, we mainly focus on convolutive case in this study. The convolutive BSS has several practical applications

in acoustics, telecommunication and seismology. Actually, it can be applied in any area that has linear mixtures of time-series as observation and aims to find the original independent time-series. However, we deal with acoustic mixtures that are resulted in convolutions of audio sources and impulse response of acoustic reverberative environment. This case is called "Cocktail-Party Problem" in the literature. The main application of convolutive BSS problem in acoustics is to help people with hearing deficiencies for whom the task of speech discrimination is very difficult in the presence of multiple conversations. Another application that we implemented in this study is to separate noise from the noisy speech and thus improve speech recognition performance using Convolutive BSS method. In this case, BSS can be regarded as a front-end processing mechanism for speech recognition engine.

1.2. Literature Review

BSS problem is a recent and remarkable chapter in the development of signal processing. The field began at a neural network conference in Utah in 1986 with Jutten and Heraut paper [12]. Source Separation has long been a topic of interest in electrical engineering and many algorithms had been developed before BSS was introduced. However, before BSS, major assumptions were always required on the nature of the sources. BSS problem can be categorized via different properties such as, mixing model types, separation criterion types, algorithm implementation domain types. Let us consider mixing model type at first. According to mixing models, BSS problem can be examined in two types, instantaneous mixing models and convolutive mixing models. In Instantaneous mixing models which are described in section 2.1.1 in detail, observed signals are the sum of scaled versions of source signals and many algorithms have been developed to solve this problem [5, 36]. Algorithms developed to perform instantaneous blind separation of sources were given the name Independent Component Analysis (ICA) algorithms and the term ICA is often used interchangeably with BSS. Bell and Sejnowski's [2] approach to ICA became the most popular choice to BSS problem due to its simplicity and its favorable convergence properties. However, the algorithm involved matrix inversion which significantly hindered efficiency. Amari discovered an important improvement using Natural Gradient Descent [30] to

the algorithm of Bell and Sejnowski which eliminated the matrix inversion. This gave a significant performance improvement and made ICA more practical for real world problems. Another important ICA algorithm, called FastICA [6], was developed in 1997 by Oja and Hyvarinen of the Helsinki University of Technology. It was shown to be a very good alternative to Bell and Sejnowski's algorithm, and is probably currently the most widely used ICA algorithm.

There are multiple potential applications of convolutive blind source separation. In acoustics, different sound sources are recorded simultaneously with possibly multiple microphones. These sound sources may be speech or music or underwater signals recorded in passive sonar [7]. The *cocktail-party problem* has come to characterize the task of recovering speech in a room of simultaneous and independent speakers [11, 34]. Convolutive BSS proposed as a solution to this problem. We know from [5] under some conditions, we can find an inverse system of a multiple-input and multiple-output system with appropriately chosen multi-dimensional filters. The first solution to the convolutive BSS problem was developed by Bell and Sejnowski [2], who proposed a feedforward neural network structure using FIR filters. This approach was improved by Torkkola who employed a similar network, but with feedback structure [23]. Both of these approaches are limited in their practicality, primarily because they involve computationally expensive convolutions of long filters. To overcome the shortcomings of these time domain algorithms, for convolutive mixtures, frequency-domain BSS was proposed and is very popular since all techniques originally developed for instantaneous BSS can be applied independently in each frequency bin. This bin-wise processing, implying a narrowband signal model is denoted as the narrowband approach. Such narrowband approaches for convolutive BSS and methods for instantaneous BSS can be found, e.g., in [5], [22], [15]. Unfortunately, this traditional narrowband approach exhibits several limitations as identified in [31]. In particular, the permutation problem, which is inherent in BSS, may then also appear independently in each frequency bin so that extra repair measures have to be taken to address this internal permutation. Problems caused by circular convolution effects due to the narrowband approximation are reported in, e.g., [16]. In order solve these problems, beamforming, clustering and independent vector analysis (IVA) are mainly used. Although adaptive beamforming

(ABF) and BSS both attempt to extract the selected source signals from observed sensor mixtures by a filter array, time and frequency information are utilized in convolutive blind source separation while spatial information of source signals or sensor array is used in adaptive beamforming. In [24], the geometric information used in adaptive beamforming is incorporated in frequency domain convolutive BSS algorithm as linear constraints or as the initial adaptation condition. A convolutive BSS system can be viewed as multiple sets of adaptive beamforming, which means the separation filter array for every output can be viewed as a beamformer. In [35], this idea is used to deal with frequency permutation and arbitrary scaling problem in frequency domain convolutive BSS. Permutation ambiguity problem can be solved using via clustering methods. Recently Sawada proposed a method based on clustering basis vectors of estimated mixing filter, where only prior information the maximal distance between sensor is required and a sophisticated normalization method is introduced [19]. A new ICA formulation with multivariate components, which is called independent vector analysis (IVA), has been proposed in order to avoid permutation correction in BSS, especially with speech [37, 20, 4]. All frequency components of a source are considered together as a multivariate signal and hence, instead of employing a contrast function that measures the source independence in each frequency bin, one that measures the whole independence among multivariate sources is employed. In this thesis, we study a rigorous derivation of a more general class of broadband algorithms for convolutive mixtures based on second-order statistics, i.e., the frequency bins are no longer considered to be independent for unrestricted time-domain signals. This general approach inherently avoids the above-mentioned problems.

1.3. Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, the mixing models are introduced and their mathematical properties are given and different gradient approaches are also investigated. Chapter 3 gives the details of the method implemented and also introduces an efficient version of the method. Moreover, in this chapter three different versions of implementation; offline, online and block-online are presented. In chapter 4, the validation of the studied solution based on the simulation results is

presented. The performance of the method is evaluated based on several parameters, it is shown that the method performs well for speech separation. In chapter 5, the studied method is applied to a noisy speech problem to separate noise from the speech and it is found that using the method as a denoiser will improve speech recognition performance. Finally, Chapter 6 gives the conclusion remarks and the road-map of the future work.

2. METHODOLOGY

We studied a time-domain algorithm to the BSS problem in this thesis. The algorithm is mainly based on the study in [29, 28, 18]. Firstly, The structure of the problem scheme will be explained and then studied solution will be described in detail.

2.1. Model Description

In BSS, we are trying to recover the source signals from their linear mixtures so the way how source signals reach to the sensors must be investigated to study a solution to the recovering problem.

There can be two types of mixing systems as stated before.

1. Instantaneous Mixing System
2. Convolutional Mixing System

2.1.1. Instantaneous Mixing System

In Instantaneous Mixing System, sensor signals are the sum of scaled versions of the sources. That is, if we denote source signals with s , sensor signals with x and mixing coefficients with h , n -th sample of the sensor signal is

$$x_p(n) = \sum_{q=1}^P s_q(n) h_{qp} \quad (2.1)$$

, If we write this for all sensors, we have

$$\begin{pmatrix} x_1(n) \\ \vdots \\ x_P(n) \end{pmatrix} = \begin{pmatrix} h_{11} & h_{21} & \cdots & h_{P1} \\ \vdots & \vdots & \vdots & \vdots \\ h_{1P} & h_{2P} & \cdots & h_{PP} \end{pmatrix} \begin{pmatrix} s_1(n) \\ \vdots \\ s_P(n) \end{pmatrix} \quad (2.2)$$

and we can represent this equation in a compact form as,

$$\mathbf{x}(n) = \mathbf{H}\mathbf{s}(n) \quad (2.3)$$

where

$$\mathbf{x}(n) = [x_1(n), x_2(n), \dots, x_P(n)], \quad (2.4)$$

is sensor signals vector,

$$\mathbf{s}(n) = [s_1(n), s_2(n), \dots, s_P(n)] \quad (2.5)$$

source signals vector, and $\mathbf{H}$ is the "mixing matrix" that contains the mixing coefficients.

2.1.2. Convolutional Mixing System

In Convolutional Mixing System, since in real acoustic environments, sound travels slowly compared to the distances, source signals arrive at the different sensors at different times and due to the reverberation instead of only one sample of sources, more than one samples reaches to the sensor from the same source at the same time. Therefore, sensor signals are not just the sum of the scaled versions of sources. For example, at time n , we can have $s_q(n - n_{d1})$ and $s_q(n - n_{d2})$ at the same sensor. As a result, sensor signals are convolved with the room impulse response and hence we have convolutional mixtures at the sensors and for each sample of sensor signals we have:

$$x_p(n) = \sum_{q=1}^P \sum_{k=0}^{M-1} h_{qp}(k) s_q(n - k) \quad (2.6)$$

where $h_{qp}(k), k = 0, 1, \dots, M - 1$ denote the coefficients of the filter from q th source to p th sensor. M denotes the mixing filter length and its value depends on the reverberative property of the acoustic environment, e.g for a reverberation time of 50

ms and a sampling frequency of 8 kHz gives $M = 400$ taps. For convolutive mixtures, equation 2.3 becomes:

$$\mathbf{x}(n) = \mathbf{H} * \mathbf{s}(n) \quad (2.7)$$

In this equation, every element of $\mathbf{H}$ is a filter. When we examine these two systems, it can be concluded that, Instantaneous Mixtures are subsets of Convolutive Mixtures which are more complicated to separate. However, for both mixing systems, the aim of BSS is to find a demixing system such that we can separate the source signals from each other and note that for convolutive mixtures, we are not aiming the deconvolution of individual sensor signals. By using BSS method, we can reconstruct the source signals up to an unknown permutation and an unknown filtering of the individual sources because of the fact that if two signals are independent, their scaled, permuted and delayed versions are also independent so independence assumption is not enough to recover the source signals in their original scale, original order and original time dependency. However, delayed, permuted and scaled versions are good enough for practical applications. Under these constraints, our demixing system will be as: For

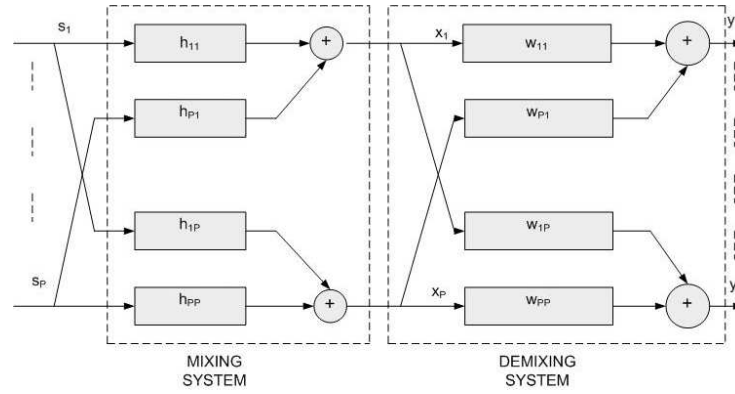


Figure 2.1. Mixing and Demixing System

each sample of demixing system's output signal is:

$$y_q(n) = \sum_{p=1}^P \sum_{k=0}^{L-1} w_{pq}(k) x_p(n-k) \quad (2.8)$$

$w_{pq}(k), k = 0, 1, \dots, L - 1$ denote the current weights of the MIMO filter taps from the p th sensor to the q th output. Actually, BSS aims to find out such w_{pq} values that the separation of sources is achieved. A complete BSS system that contains mixing and demixing system can be shown as in Figure 2.1. Afterwards, we will deal with convolutive mixtures problem because if we can solve it, we can easily adapt this solution to the Instantaneous Mixtures. In order to simplify the problem without losing of generality, we made two assumptions about the overall system, they are:

1. Number of sources Q = Number of sensors P
2. Mixing filter tap number M = Demixing filter tap number L

The reason why we have number of demixing filter taps is equal to the number of mixing filter taps numbers is to allow the modeling of all reflections due to reverberation. By assuming number of sources and number of sensors are equal, we do not lose anything from generality as more general scenario with different number of sensors and sources can always be reduced to the assumed BSS model [5, 25, 32].

2.2. Block Time-Domain BSS Algorithm

2.2.1. Matrix Formulation For Convolutive Mixtures

We use the nonstationarity and nonwhiteness properties of signals to find the demixing filter coefficients, therefore, we need to process output signals block by block to make use of these properties. For that reason, we have to write the relation between a block of the output signals and the sensor signals. Moreover, in order to design a demixing system we should understand the rationale behind the mixing system so firstly sensor and source signal relations will be introduced and by using the analogy output signal, sensor signal relations will be derived. To make more precise matrix formulation, we need to introduce two more parameters that are used to exploit the nonstationarity and nonwhiteness properties. These parameters are:

- number of time-lags D taken into account for the correlations in the cost function

- length N of output signal block as basis for the estimates of short-time correlations.

Let's take a sensor signal block with time index m as:

$$\mathbf{x}_p(m) = [x_p(mL), \dots, x_p(mL + N - 1)]^T \quad (2.9)$$

We can express sensor signal samples in terms of source signal samples and mixing system parameters as:

$$x_p(n) = \sum_{q=1}^P \mathbf{s}_q^T(n) \mathbf{h}_{qp} \quad (2.10)$$

where

$$\mathbf{s}_q(n) = [s_q(n), s_q(n-1), \dots, s_q(n-L+1)]^T \quad (2.11)$$

is a vector containing the latest L samples of the source signal s_q of the q -th channel and

$$\mathbf{h}_{qp} = [h_{qp}(0), h_{qp}(1), \dots, h_{qp}(L-1)]^T \quad (2.12)$$

contains the current weights of the MIMO filter taps from the q -th source to the p -th sensor. We can express source signal and sensor signal relation in vector form using equations (2.9) and (2.10). In that case, we can write the relation as;

$$\begin{pmatrix} x_p(mL) \\ \vdots \\ \vdots \\ x_p(mL + N - 1) \end{pmatrix} = \begin{pmatrix} s_q(mL) & \cdots & s_q(mL - L + 1) \\ s_q(mL + 1) & \cdots & s_q(mL - L + 2) \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ s_q(mL + N - 1) & \cdots & s_q(mL - L + N) \end{pmatrix} \begin{pmatrix} h_{qp}(0) \\ h_{qp}(1) \\ \vdots \\ h_{qp}(L - 1) \end{pmatrix} \quad (2.13)$$

Since we use sensor signal block matrix in demixing process, we have to write a sensor signal block matrix in terms of source signal matrix and mixing filter matrix. We can do it as

$$\mathbf{X}_p(m) = \begin{pmatrix} x_p(mL) & \cdots & x_p(mL - D + 1) \\ x_p(mL + 1) & \cdots & x_p(mL - D + 2) \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ x_p(mL + N - 1) & \cdots & x_p(mL - D + N) \end{pmatrix} \quad (2.14)$$

Let's write $\mathbf{X}_p(m)$ in terms of source signal samples and mixing filter coefficients:

$$\mathbf{S}_q(m) = \begin{pmatrix} s_q(mL) & \cdots & \cdots & s_q(mL - D - L + 2) \\ s_q(mL + 1) & \cdots & \cdots & s_q(mL - D - L + 3) \\ \vdots & \vdots & \vdots & \vdots \\ s_q(mL + N - 1) & \cdots & \cdots & s_q(mL - D - L + N + 1) \end{pmatrix} \quad (2.15)$$

and

$$\mathbf{H}_{qp}(m) = \begin{pmatrix} h_{qp}(0) & 0 & \cdots & 0 \\ h_{qp}(1) & h_{qp}(0) & \cdots & \cdots \\ \vdots & h_{qp}(1) & \cdots & 0 \\ h_{qp}(L-1) & \vdots & \vdots & h_{qp}(0) \\ 0 & h_{qp}(L-1) & \cdots & h_{qp}(1) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & h_{qp}(L-1) \\ 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} \quad (2.16)$$

It should be emphasized that $\mathbf{H}_{qp}$ matrix has a Sylvester Structure that all columns are shifted versions of each other and note that for the general case $1 \leq D \leq L$, the last $L-D+1$ rows are padded with zeros to ensure compatibility with the size of $\mathbf{S}_q(m)$.

Using these definitions, we can express sensor signal block matrix in terms of source signal block matrices and mixing filter matrix as:

$$\mathbf{X}_p(m) = \sum_{q=1}^P \mathbf{S}_q(m) \mathbf{H}_{qp} \quad (2.17)$$

When we combine all channels together to write overall input-output relation of mixing system, we will have:

$$\mathbf{X}(m) = \mathbf{S}(m) \mathbf{H} \quad (2.18)$$

with the matrices

$$\mathbf{X}(m) = [\mathbf{X}_1(m) \cdots \mathbf{X}_P(m)] \quad (2.19)$$

$$\mathbf{S}(m) = [\mathbf{S}_1(m) \cdots \mathbf{S}_P(m)] \quad (2.20)$$

and

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{11} & \cdots & \mathbf{H}_{1P} \\ \vdots & \vdots & \vdots \\ \mathbf{H}_{P1} & \cdots & \mathbf{H}_{PP} \end{pmatrix} \quad (2.21)$$

Let's investigate the same relations between sensor signals and output signals. Let's take an output block with time index m as:

$$\mathbf{y}_q(m) = [y_q(mL), \dots, y_q(mL + N - 1)]^T \quad (2.22)$$

We can write equation 2.8 as:

$$y_q(n) = \sum_{p=1}^P \mathbf{x}_p^T(n) \mathbf{w}_{pq} \quad (2.23)$$

where

$$\mathbf{x}_p(n) = [x_p(n), x_p(n-1), \dots, x_p(n-L+1)]^T \quad (2.24)$$

is a vector containing the latest L samples of the sensor signal x_p of the p -th channel where

$$\mathbf{w}_{pq} = [w_{pq}(0), w_{pq}(1), \dots, w_{pq}(L-1)]^T \quad (2.25)$$

contains the current weights of the MIMO filter taps from the p -th sensor to the q -th output. In order to use nonstationarity property, we have to make estimations for short-time correlation so we will write the equation (2.22) for the output vector in (2.23) to incorporate N -length block output vector, in that case we will have

$$\begin{pmatrix} y_q(mL) \\ \vdots \\ y_q(mL+N-1) \end{pmatrix} = \begin{pmatrix} x_p(mL) & \cdots & x_p(mL-L+1) \\ x_p(mL+1) & \cdots & x_p(mL-L+2) \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ x_p(mL+N-1) & \cdots & x_p(mL-L+N) \end{pmatrix} \begin{pmatrix} w_{pq}(0) \\ w_{pq}(1) \\ \vdots \\ w_{pq}(L-1) \end{pmatrix} \quad (2.26)$$

Now, by incorporating D time-lags, we will use nonwhiteness property of the source signals. We will define a $N \times D$ matrix to use nonstationarity and nonwhiteness together.

$$\mathbf{Y}_q(m) = \begin{pmatrix} y_q(mL) & \cdots & y_q(mL-D+1) \\ y_q(mL+1) & \cdots & y_q(mL-D+2) \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ y_q(mL+N-1) & \cdots & y_q(mL-D+N) \end{pmatrix} \quad (2.27)$$

Let's write $\mathbf{Y}_q(m)$ in terms of sensor signal samples and demixing filter coefficients:

$$\mathbf{X}_p(m) = \begin{pmatrix} x_p(mL) & \cdots & \cdots & x_p(mL - D - L + 2) \\ x_p(mL + 1) & \cdots & \cdots & x_p(mL - D - L + 3) \\ \vdots & \vdots & \vdots & \vdots \\ x_p(mL + N - 1) & \cdots & \cdots & x_p(mL - D - L + N + 1) \end{pmatrix} \quad (2.28)$$

and

$$\mathbf{W}_{pq}(m) = \begin{pmatrix} w_{pq}(0) & 0 & \cdots & 0 \\ w_{pq}(1) & w_{pq}(0) & \cdots & \cdots \\ \vdots & w_{pq}(1) & \cdots & 0 \\ w_{pq}(L-1) & \vdots & \vdots & w_{pq}(0) \\ 0 & w_{pq}(L-1) & \cdots & w_{pq}(1) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & w_{pq}(L-1) \\ 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} \quad (2.29)$$

It should be emphasized that $\mathbf{W}_{pq}$ matrix also has a Sylvester Structure and note that for the general case $1 \leq D \leq L$, the last $L-D+1$ rows are padded with zeros to ensure compatibility with the size of $\mathbf{X}_p(m)$. Using these definitions, we can express output signal block matrix in terms of sensor signal block matrices and demixing filter matrix as:

$$\mathbf{Y}_q(m) = \sum_{p=1}^P \mathbf{X}_p(m) \mathbf{W}_{pq} \quad (2.30)$$

When we combine all channels together to write overall input-output relation of demixing system, we will have:

$$\mathbf{Y}(m) = \mathbf{X}(m) \mathbf{W} \quad (2.31)$$

with the matrices

$$\mathbf{Y}(m) = [\mathbf{Y}_1(m) \cdots \mathbf{Y}_P(m)] \quad (2.32)$$

$$\mathbf{X}(m) = [\mathbf{X}_1(m) \cdots \mathbf{X}_P(m)] \quad (2.33)$$

and

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_{11} & \cdots & \mathbf{W}_{1P} \\ \vdots & \vdots & \vdots \\ \mathbf{W}_{P1} & \cdots & \mathbf{W}_{PP} \end{pmatrix} \quad (2.34)$$

2.2.2. Cost Function and Algorithm Derivation

In BSS applications for convolutive mixtures, we want to separate output channels while putting no constraints on the temporal structure of the individual signals so we should express the relationship between source signals and sensor signals as we did for sensor and output signals. We have

$$\mathbf{X} = \mathbf{S}\mathbf{H} \quad (2.35)$$

so we can express overall system as

$$\mathbf{Y} = \mathbf{S}\mathbf{H}\mathbf{W} \quad (2.36)$$

We know for convolutive mixtures, filtering ambiguity is inevitable, therefore, it is at best possible to obtain an arbitrary block diagonal matrix $\mathbf{C} = \mathbf{H}\mathbf{W}$ for overall system. Actually, by trying to obtain a block diagonal matrix, we are aiming only separation of the signals, not aiming the deconvolution of individual signals so this approach will not cause any noticeably effect on the spectral content of the individual

signals. As a result, in our BSS problem, we want to form a demixing filter coefficient vector so that we could have a block diagonal resultant matrix $\mathbf{C}$. In this thesis, we will focus on a wideband solution based on second-order statistics (SOS) to solve this block diagonalization problem. Since we will use SOS in the solution, the stochastic signals are described with their short-time correlation matrices with lags based on the matrix formulation introduced above. Our short-time correlation matrices are

$$\mathbf{R}_{\mathbf{xx}} = \mathbf{X}^H(m)\mathbf{X}(m) \quad (2.37)$$

$$\mathbf{R}_{\mathbf{yy}} = \mathbf{Y}^H(m)\mathbf{Y}(m) \quad (2.38)$$

of size $2\text{PL} \times 2\text{PL}$ and $\text{PD} \times \text{PD}$ respectively. Now the problem is that how can we estimate our short-time correlation matrices. Actually there are two basic methods used in the correlation matrix estimation for nonstationary signals and they are:

1. Correlation Method
2. Covariance Method

In Covariance Method, all samples of the signal are used to estimate the correlation matrices whereas in Correlation Method, samples of the time interval whose correlation matrix will be estimated are used [13]. When these two methods are compared, it is known that while the correlation method leads to a slightly lower computational complexity, covariance method results more accurate results. In our solution, we will use the correlation method in the estimation problem. Based on the correlation matrices defined above, we will use a cost function first introduced in [17] which inherently includes all L time-lags of all auto-correlations and cross-correlations of the BSS output signals and we have the cost function as:

$$J(m, \mathbf{W}) = \sum_{i=0}^{\infty} \beta(i, m) \{ \log \det \text{bdiag} \mathbf{Y}^H(i)\mathbf{Y}(i) - \log \det \mathbf{Y}^H(i)\mathbf{Y}(i) \} \quad (2.39)$$

In this equation, H denotes complex conjugate transposition, the bdiag operation on a partitioned block matrix sets all submatrices on the off-diagonals to zero and β is a weighting function with finite support that is normalized according to the equation $\sum_{i=0}^m \beta(i, m) = 1$. The weighting function gives us the opportunity of implementing the separation algorithm in three different ways, these three versions of the algorithm are:

1. Online
2. Offline
3. Block-online

These versions will be described in section (2.3) in detail. For BSS, these submatrices refer to the correlation and cross-correlation matrices of different channels of size $L \times L$. The cost function becomes zero if and only if the cross-correlation matrices will be zero, that means output cross-correlation over all time-lags become zero. This can be shown as in Figure (2.2). By Oppenheim's Inequality [8], we know $\sum_q \log \det \mathbf{Y}_q^H \mathbf{Y}_q \geq$

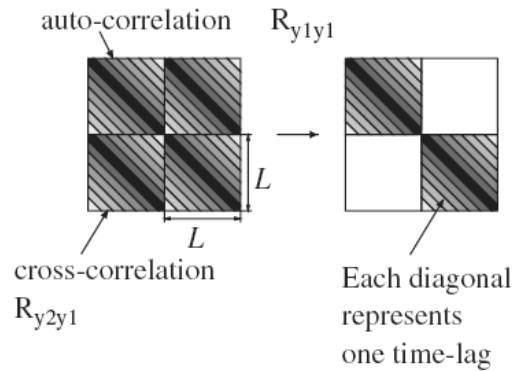


Figure 2.2. Correlation Matrices of mixed and demixed signals

$\log \det \mathbf{Y}^H \mathbf{Y}$ so it is guaranteed that the first term in the cost function is bigger than the second one, that is, the cost function is always non-negative. Let's use the geometry so as to understand what the cost function tries to do. If we think the determinant function as the volume of a parallelepiped and block matrices as column vectors that are spanning the parallelepiped, the cost function tries to make these column vectors orthogonal to each other simultaneously.

2.2.3. Independence, Non-stationarity and Non-whiteness Properties

In this section, the properties that are used to separate the mixed signals from each other will be explained. As we know, independence requires the uncorrelatedness and so the correlation matrix of the independent signals are diagonal. Therefore, when the speech signals are not mixed their correlation matrices are diagonal but when they are mixed we do not have diagonal correlation matrices any more. Using this property, we can try to find such a demixing filter matrix that makes the non-diagonal correlation matrix of the mixed signals diagonal.

However, since the mixtures are convolutive, the diagonalization is not sufficient to find demixing filter coefficients. For that reason, non-whiteness and non-stationarity of the speech signals are used together. We defined the correlation matrices for multiple time-lags and using this correlation matrices we exploited the nonwhiteness property. If the speech signals are white, signal samples will be uncorrelated and so the correlation matrix at multiple time-lags will be diagonal and the correlation matrix at multiple time lags will not create new conditions that help us to find the demixing filters. The new conditions are created via nonwhiteness because we create a new constraint that auto-correlations at multiple time-lags will not be zero. Without nonwhiteness, correlation matrix will be as Figure (2.3), but with nonstationarity the matrix will be as in Figure (2.4) so it gives us the more equations to solve to find demixing filters.

Stationarity requires that signal statistics does not change in time. However, it is known that speech signals are nonstationary signals and their statistical properties change in time. For that reason, with block by block processing of the speech signals we can impose the correlation matrices of every block to be block-diagonal at the output and for each time interval we will create new conditions to solve demixing filter coefficients. Therefore, nonstationarity helps us to find demixing filters.

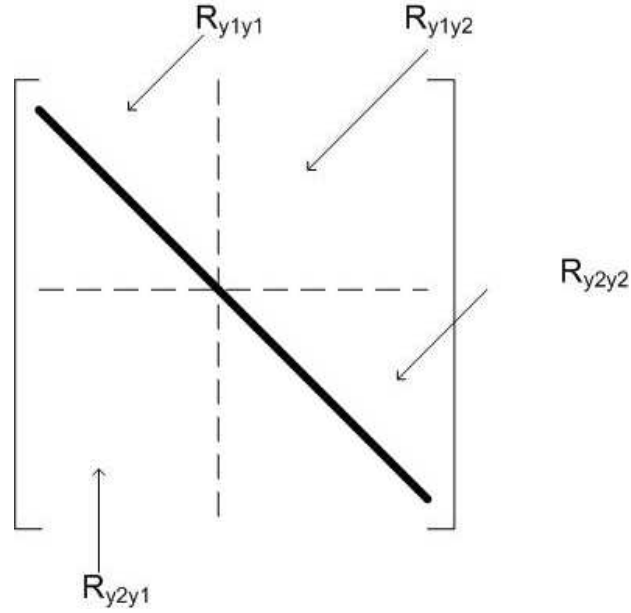


Figure 2.3. Correlation Matrix with whiteness

2.3. Gradient Approaches to the Update of Demixing Filter Coefficient

In the previous section, we introduced the cost function that will be minimized and hence used to solve the BSS problem in time-domain. Therefore, now the problem is reduced to solve a block-diagonalization problem and after this time the focus will be on how to block diagonalize the output correlation matrices over multiple time-lags. As stated, non-stationarity and non-whiteness properties are used in diagonalization procedure. Now, we have defined the cost function and we need to find a method to minimize it. In this study, first-order gradient approach will be used but extension to higher order gradients are straightforward albeit computationally expensive. We will need an update term that will be used to update the demixing filter coefficients and gradually the cost function will be reduced. There can be many different update methods that may be used, but we will focus on Standard Gradient Approach and Natural Gradient Approach.

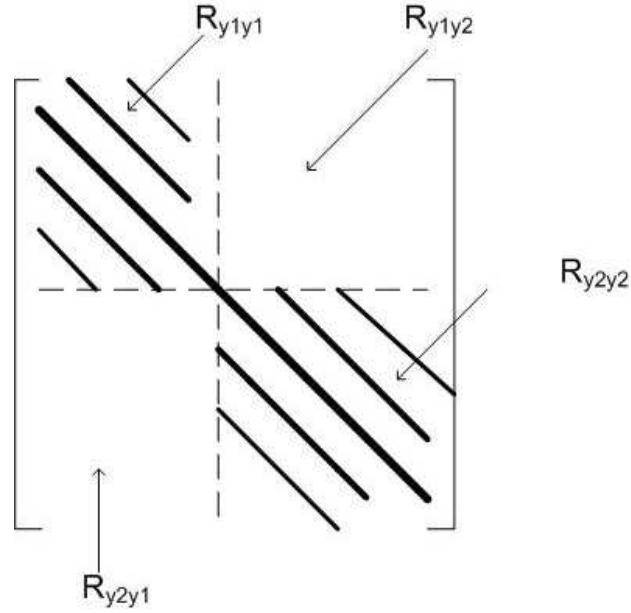


Figure 2.4. Correlation Matrix with nonwhiteness

2.3.1. Standard Gradient Approach

Since the update will be on demixing filter matrix $\mathbf{W}$, we will take gradient of the cost function with respect to it and we get

$$\nabla_{\mathbf{W}} J(m) = 2 \frac{\partial J(m)}{\partial \mathbf{W}^*} \quad (2.40)$$

when we write the output correlation matrix $\mathbf{R}_{yy}(m)$ in terms of demixing matrix, we can see the effect of taking gradient with respect to $\mathbf{W}$

$$\mathbf{R}_{yy} = \mathbf{W}^H \mathbf{X}^H \mathbf{X} \mathbf{W} \quad (2.41)$$

, and

$$\mathbf{R}_{yy} = \mathbf{W}^H \mathbf{R}_{xx} \mathbf{W} \quad (2.42)$$

and it was proven that in [18]

$$\frac{\partial}{\partial \mathbf{W}^*} \log \det \mathbf{W}^H \mathbf{R}_{xx} \mathbf{W} = \mathbf{R}_{xx} \mathbf{W} (\mathbf{W}^H \mathbf{R}_{xx} \mathbf{W})^{-1} \quad (2.43)$$

and

$$\frac{\partial}{\partial \mathbf{W}^*} \log \det bdiag \mathbf{W}^H \mathbf{R}_{xx} \mathbf{W} = \mathbf{R}_{xx} \mathbf{W} (bdiag \mathbf{W}^H \mathbf{R}_{xx} \mathbf{W})^{-1} \quad (2.44)$$

When we use these equations in the cost function, we get the gradient term below;

$$\nabla_{\mathbf{W}} J(m) = 2 \sum_{i=0}^{\infty} \beta(i, m) \mathbf{R}_{xx} \mathbf{W} \{ bdiag^{-1} \mathbf{R}_{yy} - \mathbf{R}_{yy}^{-1} \} \quad (2.45)$$

$$\nabla_{\mathbf{W}} J(m) = 2 \sum_{i=0}^{\infty} \beta(i, m) \mathbf{R}_{xy} \{ bdiag^{-1} \mathbf{R}_{yy} - \mathbf{R}_{yy}^{-1} \} \quad (2.46)$$

$$\nabla_{\mathbf{W}} J(m) = 2 \sum_{i=0}^{\infty} \beta(i, m) \mathbf{R}_{xy} \mathbf{R}_{yy}^{-1} \{ \mathbf{R}_{yy} - bdiag \mathbf{R}_{yy} \} . bdiag^{-1} \mathbf{R}_{yy} \quad (2.47)$$

where $bdiag^{-1} \mathbf{R}_{yy} = (bdiag \mathbf{R}_{yy})^{-1}$. Now we have an update term and using an iterative optimization procedure, the current demixing matrix can be obtained by the recursive update equation

$$\mathbf{W}(m) = \mathbf{W}(m-1) - \mu \Delta \mathbf{W}(m), \quad (2.48)$$

where μ is a stepsize parameter and $\Delta \mathbf{W}(m)$ is the update term which is equal to $\nabla_{\mathbf{W}} J(m)$ for gradient descent adaptation.

2.3.2. Natural Gradient Approach

Standard Gradient Approach suffers from slow convergence in many practical problems and its performance depends on the data being processed. We know in BSS applications, the performance of the separation algorithm depends on the mixing system because the data being processed is the samples of the sensor signal and they are convolutions of the mixing system and source signal samples. Therefore, we need to modify the ordinary gradient algorithm such that it largely removes the all effects of ill-conditioned mixing systems. This modification is termed Natural Gradient by Amari [30] and explained in appendix A. This modification is formulated as

$$\nabla_{\mathbf{W}}^{NG} J = \nabla_{\mathbf{W}} J \mathbf{W}^H \mathbf{W} \quad (2.49)$$

For our approach, we will reorder the equation as

$$\nabla_{\mathbf{W}}^{NG} J = \mathbf{W}^H \mathbf{W} \nabla_{\mathbf{W}} J \quad (2.50)$$

As a result, we have the following update term expression

$$\nabla_{\mathbf{W}}^{NG} J(m) = 2 \sum_{i=0}^{\infty} \beta(i, m) \mathbf{W} \{ \mathbf{R}_{yy} - bdiag \mathbf{R}_{yy} \} . bdiag^{-1} \mathbf{R}_{yy} \quad (2.51)$$

In this equation, term

$$\Gamma(i, \mathbf{W}) = \mathbf{W} \{ \mathbf{R}_{yy}(i) - bdiag \mathbf{R}_{yy}(i) \} bdiag^{-1} \mathbf{R}_{yy}(i) \quad (2.52)$$

is called the update term.

3. IMPLEMENTATION OF THE METHOD

3.1. Implementation of the Algorithm Using Different Weighting Functions

3.1.1. Off-line Implementation

Off-line or in other words batch implementation of the algorithm which requires all the data that will be processed is obtained and we process all data by segmenting it into K blocks using rectangular windows. Therefore, in that case the weighting function $\beta(i, m)$ corresponds to a rectangular window that can be seen in Figure (3.1) and it can be formulated as

$$\beta(i, m) = \frac{1}{K} \epsilon_{0, (K-1)}(i) \quad (3.1)$$

where $\epsilon_{a,b} = 1$ for $a \leq i \leq b$, $\epsilon_{a,b} = 0$ else. We take the entire signal samples, segment it into K blocks and using every block we compute an amount of update and then average these amounts to get overall update amount. We repeatedly do this for every iteration so the current demixing matrix for l -th iteration is

$$\mathbf{W}^l = \mathbf{W}^{l-1} - \frac{\mu}{K} \sum_{i=0}^{K-1} \Gamma(i) \quad (3.2)$$

where $\Gamma(i)$ denotes the amount of update for each block and can be formulated as,

$$\Gamma(i, \mathbf{W}) = \mathbf{W} \{ \mathbf{R}_{yy}(i) - bdiag \mathbf{R}_{yy}(i) \} bdiag^{-1} \mathbf{R}_{yy}(i) \quad (3.3)$$

As a result, in off-line implementation, the algorithm is visiting the signal data repeatedly for every iteration and so it usually gives us better performance compared to its on-line counterpart.

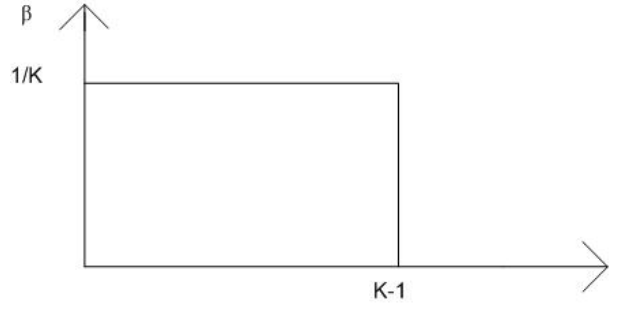


Figure 3.1. Weighting Function for Off-line Implementation

3.1.2. On-line Implementation

If our mixing system is time-varying, we have to update the demixing matrix in time so that we can adapt it to the new mixing system. Therefore, on-line realization of the algorithm is necessary and it is implemented using a weighting function shown in Figure 3.2 with an exponential forgetting factor λ and it is defined as

$$\beta(i, m) = (1 - \lambda)\lambda^{m-i}\epsilon_{0,m}(i) \quad (3.4)$$

where $0 \leq \lambda < 1$ and amount of update for current block becomes

$$\Delta \mathbf{W}(m) = (1 - \lambda) \sum_{i=0}^m \lambda^{m-i} \Gamma(i) \quad (3.5)$$

when we write the same equation for (m-1) th block we get

$$\Delta \mathbf{W}(m-1) = (1 - \lambda) \sum_{i=0}^{m-1} \lambda^{m-1-i} \Gamma(i) \quad (3.6)$$

and from these equations we conclude that current update term becomes

$$\Delta \mathbf{W}(m) = \lambda \Delta \mathbf{W}(m-1) + (1 - \lambda) \Gamma(m) \quad (3.7)$$

This enables us to update the current amount of update only using previous update amount and current samples. The method is appropriate for the case that we do not

have the entire signal samples and the mixing system is nonstationary.

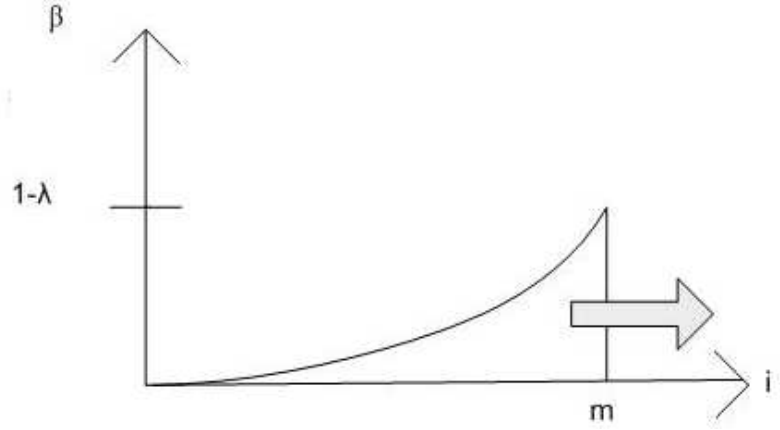


Figure 3.2. Weighting Function for On-line Implementation

3.1.3. Block-Online Implementation

While the on-line approach is appropriate for real-time processing and it can easily adopt to non-stationary environment and it is computationally cheaper as compared to off-line approach, off-line approach gives better separation performance in stationary environments so they can be combined to yield a faster convergence at moderate level of computational complexity with better performance. Block-online approach is developed to do this combination. In block-online approach, a block from the signal is taken and it is processed iteratively as in off-line approach and the resultant demixing filter coefficients are used as the initial demixing filter coefficient of the next block as in on-line approach. Moreover, the current block is segmented into overlapping subblocks in order to allow the exploitation of the non-stationarity property of the signal and for each subblock demixing filter updates are computed and then averaged to get overall demixing filter update for composite block. In this case, weighting function is

$$\beta(i, m) = \frac{1 - \lambda}{K} \sum_{m'=0}^m \lambda^{m-m'} \epsilon_{m'K, m'K+K-1}(i) \quad (3.8)$$

where m' denotes the composite block index up to the current block and m denotes the composite current block index and weighting function can be shown as in Figure

(3.3) We take a block that consists of $KL+N-L$ samples and we form K subblocks from

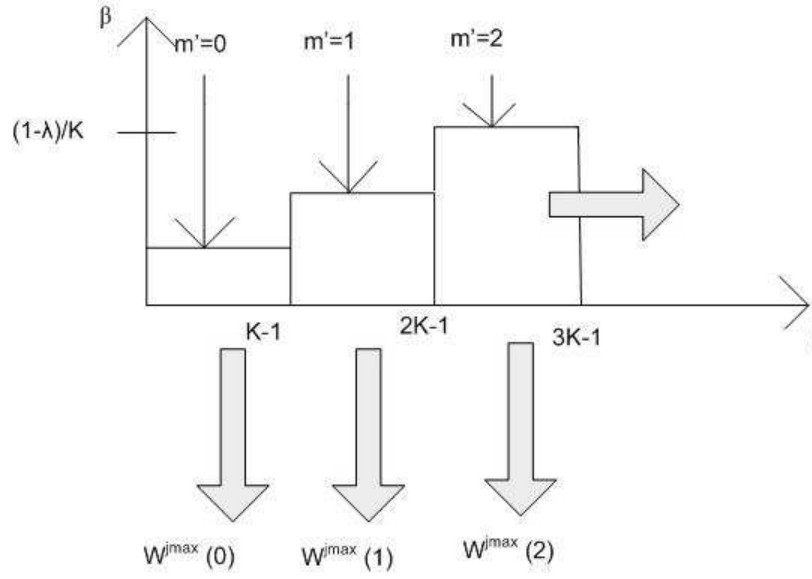


Figure 3.3. Weighting Function for Block On-line Implementation

these samples with length N and with block-shift L . We find demixing filter update for each subblock separately and then we averaged all the updates from these subblocks. We update the demixing coefficients with this average update term and we recalculate the output samples using new demixing filter coefficients. We can write this as;

$$\tilde{\mathbf{W}}^j(m) = \tilde{\mathbf{W}}^{j-1}(m) - \mu \Gamma(m, \tilde{\mathbf{W}}^{j-1}(m)), j = 1, \dots, j_{max} \quad (3.9)$$

where j_{max} represents the maximum number of offline iteration, and

$$\tilde{\Gamma}(m, \tilde{\mathbf{W}}^{j-1}(m)) = \frac{1}{K} \sum_{i=mK}^{mK+K-1} \Gamma(i, \tilde{\mathbf{W}}^{j-1}(m)) \quad (3.10)$$

where the stepsize $\mu = 2(1-\lambda)\tilde{\mu}$ and equation $\tilde{\mathbf{W}}^j(m)$ is the demixing filter matrix after j iterations formed using composite block m . Equation (3.10) contains K update terms which are computed using K subblocks. If we use only one block, we would impose only one condition to be satisfied, namely demixing filter matrix being block-diagonal, however, using subblocks we will impose K condition to be satisfied because of the fact that source signals are non-stationary and statistical properties are changing between these subblocks and these changing statistical properties produces new conditions to

be satisfied. We iteratively recompute the demixing filter matrix for j_{max} iterations and if we increase the maximum number of iterations, we provide a fast convergence of the natural gradient algorithm but it also increases the computational cost of the algorithm. After j_{max} iterations we obtain the demixing filter matrix for the current block and we use this demixing filter matrix as the initial demixing filter matrix for the next block. This is the block-online algorithm and can be written recursively as

$$\mathbf{W}(m) = \underbrace{\lambda \mathbf{W}(m-1)} + \underbrace{(1-\lambda) \tilde{\mathbf{W}}^{j_{max}}(m)} \quad (3.11)$$

In this equation, first part represents the on-line update and the second part represents off-line part of the algorithm. When we write this equation for (m-1) th block we have

$$\mathbf{W}(m-1) = \lambda^2 \mathbf{W}(m-2) + \lambda(1-\lambda) \tilde{\mathbf{W}}^{j_{max}}(m-1) + (1-\lambda) \tilde{\mathbf{W}}^{j_{max}}(m) \quad (3.12)$$

Figure (3.3) shows that the contribution of the previous blocks to the current demixing filters decrease as they are getting further from the current block. Figure (3.4) shows how to implement this algorithm for the first two blocks.

3.2. Approximated Version and Efficient Implementation of the Algorithm

We will explain how to implement the method using Natural Gradient Adaptation in a block-online manner. In order to decrease the computational complexity, we exploited the Sylvester structure of the demixing filter matrix and approximate the autocorrelation submatrices to output signal powers.

3.2.1. Estimation of Correlation Matrices using Correlation Method

There are mainly two methods to estimate correlation matrices of non-stationary signals. They are Correlation and Covariance Methods. If we assume stationarity in each block, we can approximate covariance method by correlation method that is computationally less expensive. With this assumption, we have Toeplitz structured cross-correlation and auto-correlation matrices for each i-th subblock. For every subblock

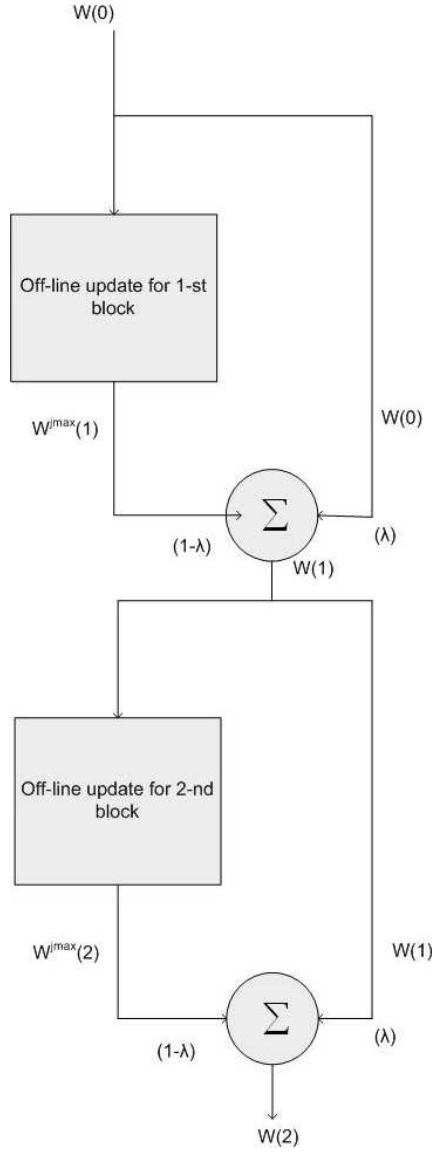


Figure 3.4. Block-Online Update Method Figure

we have correlation matrices with dimension $PL \times PL$ and we have a $L \times L$ correlation matrix for each channel from p -th sensor to q -th output and it is;

$$\mathbf{R}_{y_p y_q}(i) = \begin{pmatrix} r_{y_p y_q}(i, 0) & \cdots & r_{y_p y_q}(i, L-1) \\ r_{y_p y_q}(i, -1) & \cdots & r_{y_p y_q}(i, L-2) \\ \vdots & \vdots & \vdots \\ r_{y_p y_q}(i, -L+1) & \cdots & r_{y_p y_q}(i, 0) \end{pmatrix} \quad (3.13)$$

we can express each element in the matrix as for $u \geq 0$

$$r_{y_p y_q}(i, u) = \sum_{n=iL}^{iL+N-u-1} y_p(n+u)y_q(n) \quad (3.14)$$

and for $u < 0$

$$r_{y_p y_q}(i, u) = \sum_{n=iL-u}^{iL+N-1} y_p(n+u)y_q(n) \quad (3.15)$$

3.2.2. Approximation of the Normalization and Regularization Strategy

When we want to implement the equation

$$\Gamma(i, \mathbf{W}) = \mathbf{W} \{ \mathbf{R}_{yy}(i) - b \text{diag} \mathbf{R}_{yy}(i) \} b \text{diag}^{-1} \mathbf{R}_{yy}(i) \quad (3.16)$$

using equations (3.13),(3.14),(3.15) in a straightforward manner, we need to invert P auto-correlation Toeplitz matrices $\mathbf{R}_{y_q y_q}$ of size $L \times L$ which is normalizing the update term and this inversion requires a complexity of $\Theta(L^2)$. For reverberant environments requiring a large filter length L , inversion of such a big matrix is not an efficient implementation. Therefore, an efficient implementation, which is appropriate for reverberant environments that have large filter length L , was developed that use an approximate version of equation (3.16). This approximate version was heuristically introduced in [27, 39] and theoretically derived in [18]. The efficient version is obtained by approximating the autocorrelation submatrices in the normalization term by the output signal power. It can be shown as:

$$\mathbf{R}_{y_q y_q}(i) \approx \left(\sum_{n=iL}^{iL+N-1} y_q^2(n) \right) \mathbf{I} = \sigma_{y_q}^2(i) \mathbf{I} \quad (3.17)$$

for $q = 1, \dots, P$. With this approximation, matrix inversion is replaced by element-wise division.

3.2.3. Efficient Implementation of the Matrix-Matrix Multiplication

After getting rid of the matrix inversion problem, now we want to simplify the matrix-matrix multiplication in equation (3.16). We know from previous sections that the demixing filter matrix have a Sylvester structure. We can use this structure in simplification of the matrix-matrix multiplication. Since the columns of the Sylvester structured matrices are shifted versions of each other, if we compute the first column we can easily find the whole matrix by replicating this column. Therefore, it is not necessary to compute whole matrix elements. Another approach is to compute the L -th row of the matrix that contains all filter coefficients and using this row we can find whole matrix. We can show this as:

$$\begin{pmatrix} w_{pq}(0) & 0 & \cdots & 0 \\ w_{pq}(1) & w_{pq}(0) & \cdots & \cdots \\ \vdots & w_{pq}(1) & \cdots & 0 \\ w_{pq}(L-1) & \vdots & \vdots & w_{pq}(0) \\ 0 & w_{pq}(L-1) & \cdots & w_{pq}(1) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & w_{pq}(L-1) \\ 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} \begin{pmatrix} r_{y_p y_q}(i, 0) & \cdots & r_{y_p y_q}(i, L-1) \\ r_{y_p y_q}(i, -1) & \cdots & r_{y_p y_q}(i, L-2) \\ \vdots & \vdots & \vdots \\ r_{y_p y_q}(i, -L+1) & \cdots & r_{y_p y_q}(i, 0) \end{pmatrix} \quad (3.18)$$

$$= \begin{pmatrix} \mathbf{\Delta w_{pq}(0)} & 0 & \cdots & 0 \\ \mathbf{\Delta w_{pq}(1)} & \Delta w_{pq}(0) & \cdots & \cdots \\ \vdots & \Delta w_{pq}(1) & \cdots & 0 \\ \mathbf{\Delta w_{pq}(L-1)} & \vdots & \vdots & \mathbf{\Delta w_{pq}(0)} \\ 0 & \Delta w_{pq}(L-1) & \cdots & \Delta w_{pq}(1) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \Delta w_{pq}(L-1) \\ 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} \quad (3.19)$$

We can find all update filters finding first column elements or L-th row elements, as shown bold in (3.18) and using this simplification, matrix-matrix multiplication boils down to a matrix-vector multiplication. Moreover, this matrix-product can be easily computed as a convolution and this convolution can be found efficiently using FFT algorithm.

3.2.4. Appropriate Initialization

Until this point, we mentioned about the implementation details of the algorithm but we did not mention about where the algorithm starts from, that is, what are the initial demixing filter coefficients. Although Natural Gradient adaptation method diminishes the dependence of the separation performance of the algorithm on the mixing system and the data being processed, it is still an issue to choose suitable initial values for demixing filter coefficients to have good separation performance. Initial point choice depends on the mixing system type and in terms of causality property, we can divide the mixing systems into two groups; they are causal and non-causal mixing systems and they can be shown schematically as in Figure (3.5) For the causal mixing system shown in Figure 3.5(a), first source is closer to the first microphone and second source is closer to the second microphone so first sample at first microphone is from the first source and first sample at second microphone is from the second source. However, for the mixing system shown in Figure 3.5(b), the first sample at first microphone is from

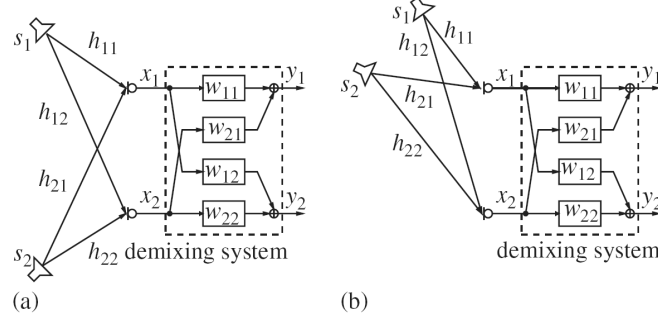


Figure 3.5. Setups for BSS requiring: (a) only causal delays and (b) causal and noncausal delays for the demixing system W .

the second source so the initialization demixing filters cannot be the same for these two configurations. For the first configuration, causal demixing filters are sufficient to achieve interference cancelation so an initialization with an unit impulse at first tap $w_{pp,0} = 1$ is sufficient. However, for the second configuration, we need a non-causal demixing filter w_{12} or w_{21} . This situation is similar to the supervised filtering algorithms used in adaptive beamforming [10] and this noncausality problem can be solved by initializing the w_{pp} with a shifted impulse. Amount of shift is dependent on the mixing system geometry. The choice of initialization method also determines how the matrix-matrix multiplication is performed as explained in previous section. For the causal mixing systems, both methods explained in the previous section is appropriate and it was shown in [28] that algorithms choosing the first column are slightly more robust. However, for the noncausal mixing system, it is necessary to initialize with a shifted unit impulse. Let us choose amount of shift as the half of the filter length $L/2$. In this case, if we choose the first column to adapt, first $L/2$ samples will not be adapted but if we choose L -th row to adapt, all samples will be adapted. Therefore, for the noncausal mixing system, it is suitable to choose L -th row for adaptation in the matrix-matrix multiplication simplification. As a result, if we do not know anything about the geometry of the mixing system, it is better to choose the L -th row for the adaptation.

3.2.5. Stepsize Control

Generally, it is important to choose appropriate stepsizes in adaptive algorithms. If the stepsize is used in an offline algorithm, several trials can be run to maximize the stepsize up to the stability margin. However, for an online procedure, usually the stepsize has to be chosen very conservatively to prevent instability problems. In real world applications, it is necessary to control the stepsize to have a robust application. A simple but effective strategy for updating the stepsize in real world BSS applications is proposed in [28]. The strategy is mainly based on the method presented in [3, 26]. The procedure is as follows: If the value of the cost function $J(m)$ is decreased compared to $J(m-1)$, increase the stepsize and if the current value of $J(m)$ exceeds the previous one $J(m-1)$ by more than a specified ratio, decrease the stepsize. If the stepsize is decreased at a step, at this step demixing filters will not be updated. Let us explain the details of the stepsize update rule. It is chosen to start a small stepsize $\mu(0)$ and its modification is done as follows:

$$\mu(m+1) = \begin{cases} a.\mu(m), & \text{if } J(m) < J(m-1), a > 1 \\ b.\mu(m), & \text{if } J(m) \geq c.J(m-1), b < 1, c > 1 \\ \mu(m), & \text{otherwise} \end{cases} \quad (3.20)$$

For a wide range of applications, the values $a = 1.1$, $b = 0.5$, $c = 1.3$ provided robust behavior. Practically, the adaptive stepsize should be restricted to a finite range $[\mu_{min}, \mu_{max}]$. Actually, if we use the equation (3.20), we have to evaluate the cost function given in equation (2.39) and it can be seen easily that this requires the computation of the determinant of a large PLxPL matrix $\mathbf{Y}^H \mathbf{Y}$. In order to avoid this computational complexity, we can use the already computed cross-correlation sequences $r_{y_p y_q}(m, u)$, $p \neq q$ instead. We can use the norm of the cross-correlation sequence and we can use a scalar value to use in (3.20) to make the stepsize decision.

4. SEPARATION PERFORMANCE RESULTS

4.1. Introduction

In these experiments, original source signals are filtered with room impulse responses that are created in computer. In order to measure the separation performance of the method, we have to do this experiment because of the fact that we need the mixing filters so as to calculate signal-to-interference ratio (SIR) after separation. In general case, SIR is defined by the ratio of the direct path energy to cross-talk (jammer) energy [14] and formulated as

$$SIR = 10 \log_{10} \frac{P_{target}}{P_{jammer}} \quad (4.1)$$

where P_{target} is the power of the wanted signal in the separated signal and P_{jammer} is the unwanted signal in the separated signal and it is measured in dB. Since we do experiments for 2 sources and 2 microphones, Let us analyze how we measure the separation performance. For microphone signals we have the equations

$$\begin{aligned} x_1 &= s_1 * h_{11} + s_2 * h_{21} \\ x_2 &= s_1 * h_{12} + s_2 * h_{22} \end{aligned} \quad (4.2)$$

and for demixing outputs

$$\begin{aligned} y_1 &= x_1 * w_{11} + x_2 * w_{21} \\ y_2 &= x_1 * w_{12} + x_2 * w_{22} \end{aligned} \quad (4.3)$$

If we express demixing outputs in terms of source signals we will have

$$\begin{aligned} y_1 &= s_1 * (h_{11} * w_{11} + h_{12} * w_{21}) + s_2 * (h_{21} * w_{11} + h_{22} * w_{21}) \\ y_2 &= s_1 * (h_{11} * w_{12} + h_{12} * w_{22}) + s_2 * (h_{21} * w_{12} + h_{22} * w_{22}) \end{aligned} \quad (4.4)$$

We should note that in order to evaluate the separation performance exactly, in the experiments we used the signals that have the same power so input powers do not affect the separation performance. In this case we measured SIR's for two demixing outputs as

$$SIR_{out1} = 10\log_{10} \frac{|h_{11} * w_{11} + h_{12} * w_{21}|^2}{|h_{21} * w_{11} + h_{22} * w_{21}|^2} \quad (4.5)$$

and

$$SIR_{out2} = 10\log_{10} \frac{|h_{21} * w_{12} + h_{22} * w_{22}|^2}{|h_{11} * w_{12} + h_{12} * w_{22}|^2} \quad (4.6)$$

4.2. Results with Synthetically Mixed Signals

4.2.1. Fixed Stepsize Results

For the first example, impulse responses are created in Matlab randomly and plotted in fig(4.1). Using the method, the demixing filter coefficients are computed

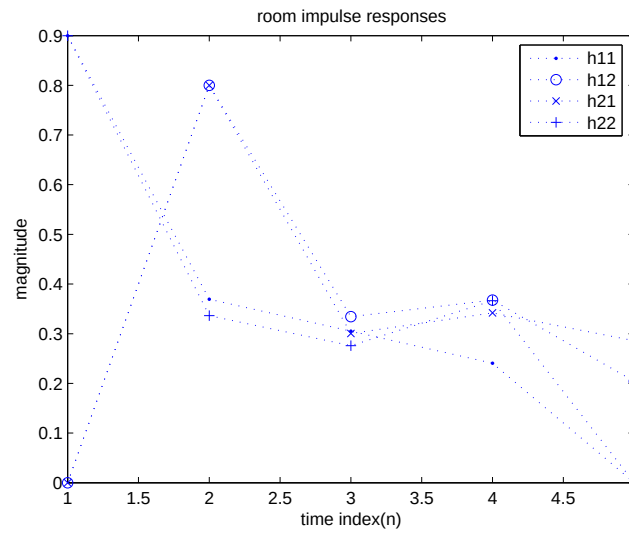


Figure 4.1. Room Impulse Responses created randomly

and hence using equation (4.5), corresponding SIR values are found for different offline iteration numbers and the results are plotted in Figure (4.2). Filter length L is 5,

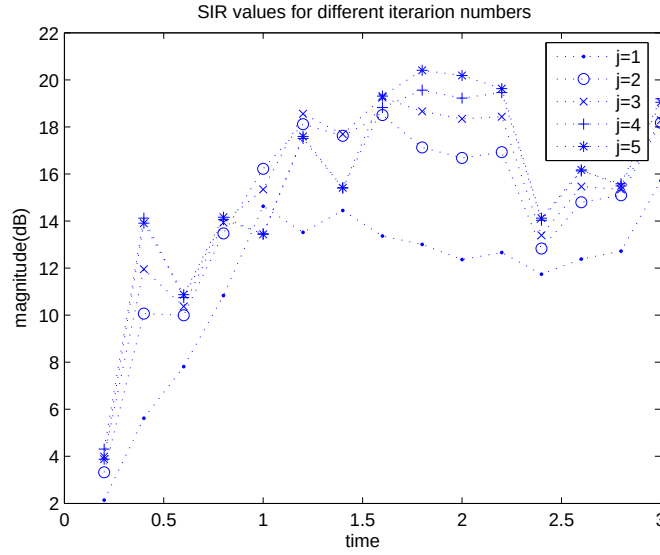


Figure 4.2. SIR values for different iteration numbers

Output Signal block length N is 10, Subblock number K is 4, step-size μ is 0.002 and forgetting factor λ is 0.2. The Figure (4.2) shows us that although for early parts of the mixture, with increasing iteration number we obtain better SIR, for the later parts iteration number does not affect the SIR because of the fact that for early parts there are not many enough samples that will be used to update demixing filters so with more iterations we improve the estimate for demixing filters. However, for the later parts, there are many enough samples to estimate them. Another reason why more iterations do not improve SIR is that the mixing impulse length is not so long, If it was so long, we would see the effect of increasing iteration number. In Figure (4.2), we observe that for some parts of the signal, SIR values are decreasing as the signals are speech signals and for these parts, we do not have speech waveforms. We can see this fact from Figures (4.3), (4.4) and (4.5) For different forgetting factors, SIR values are also computed and it was shown in Figure (4.6) that for all forgetting factors which are not equal to 0, the algorithm separates the mixture up to 20 dB. However, the performance is the best for $\lambda = 0.2$. Output signals are subblocked to make use of nonstationarity of the speech signals better so we divide the output block signals into subblock and it was shown in Figure (4.7) that the separation performance is the worst when no subblocking is done and it is best with subblock numbers 3 and 4. In section (3.2.5), algorithm stepsize decision is explained. In Figure (4.8), we did

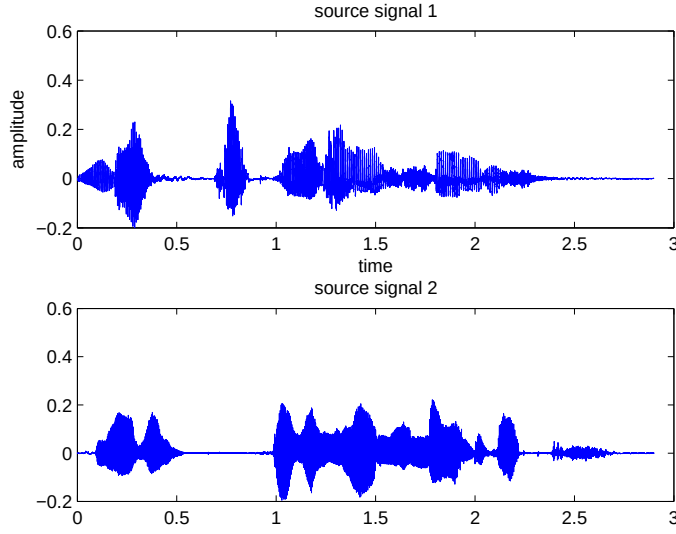


Figure 4.3. Source Signals

not use any adaptation in stepsize and results shows us that the separation results are good enough. However, the effect of adaptation in stepsize will be examined closely in the next section. As a result of these examples, it was shown that for several different parameters, the separation algorithm achieves good separation performance and the maximum SIR value is found to be 20.93 dB.

4.2.2. Adaptive Stepsize Results

This section provides the results with adaptive stepsize method described in section (3.2.5). Impulse response and sources are the same as in the previous section. When we compare the results of adaptive stepsize, Figures (4.9), (4.10),(4.11) with fixed stepsize, Figures (4.2), (4.6), (4.7), it can be shown that adaptive stepsize achieves the maximum separation performance, however, the mean of SIR values is worse when compared to fixed stepsize method. However, since we use the best demixing filters for the separation it can be concluded that adaptive stepsize approach is better than fixed stepsize approach.

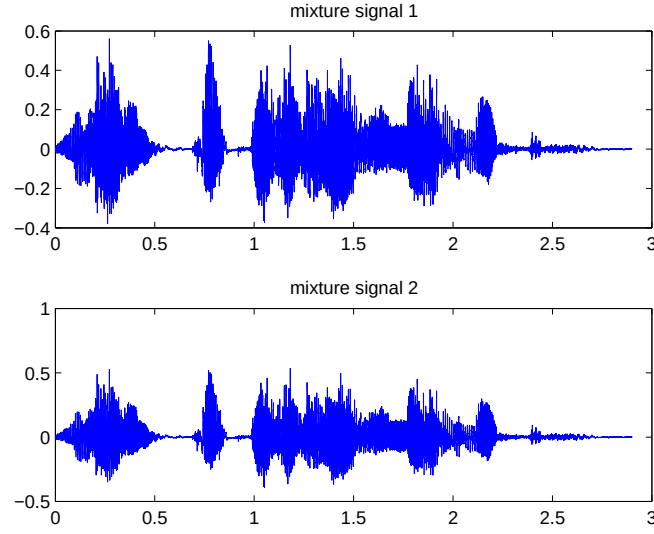


Figure 4.4. Mixture Signals

4.3. Results with signals mixed using Image Method

In this part, mixing filters will be created using Image Method [9] that uses the assumption that the room, where speakers are in, is a rectangular enclosure. The room we use is shown in Figure (4.12) and in 2D it is seen as in Figure (4.13). Now we will examine the effect of the geometrical structure on the separation performance. We will define three parameters

1. l =distance between midpoints of the speakers and midpoints of the microphones
2. d =distance between microphones
3. s =distance between sources

For the first configuration we took $s = 4m$ and $d = 0.2m$ and we found SIR values for $l=1,2$ and $4m$ and they are shown in Figure (4.14) with fixed stepsize method and in Figure (4.15) with adaptive stepsize method. From these figures, it was concluded that for this configuration, the method works well when l is $1m$ and $4m$, but it does not work for when l is $2m$ and it is also concluded that the separation performance is better with adaptive stepsize method. Since the SIR values do not increase or decrease with increasing l values, we cannot infer from these results about how the distance

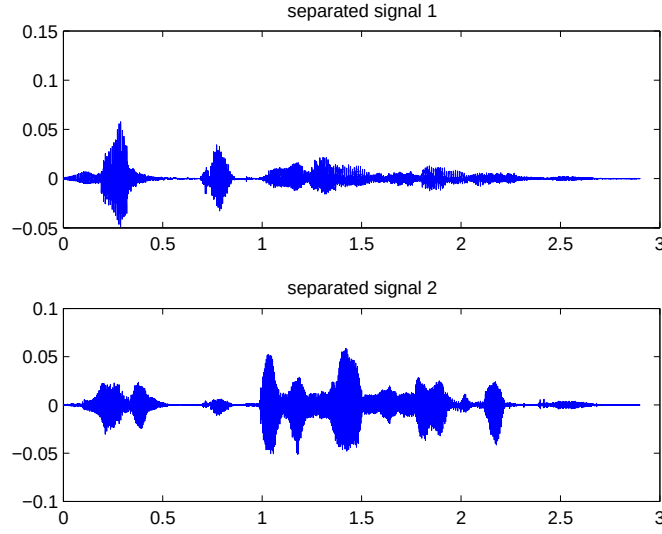


Figure 4.5. Separated Signals

between sources and microphones affects the separation performance. For the second configuration $s=4$ m and $l=4$ m and we found SIR values for $d=0.1, 0.3$ and 0.7 m which are shown in Figure (4.16) with fixed stepsize method and in Figure (4.17) with adaptive stepsize method. For this geometrical constellation, the method achieves the separation for three different values of distance between microphones and the result is best when d is 0.3 m. However, the SIR values do not correlate with distances so it cannot be concluded that how the distance between microphones affects the separation performance. For the third configuration $d=0.2$ m and $l=4$ m and we found SIR values for $s=1, 2$ and 4 m and is shown in Figure (4.18) with fixed stepsize method and in Figure (4.19) with adaptive stepsize method. For this configuration, the SIR values are best when distance between sources is 2 m with fixed stepsize method and 4 m with adaptive stepsize method. It cannot be also concluded that how the distance between sources affects the separation performance. In these experiments, since filter length we used is not long enough to reflect the acoustic properties of the medium, we can not conclude reasonable results for the effect of geometrical positions to the separation performance. However, it can be seen that adaptive stepsize approach is more successful and the method achieves separation for all configurations.

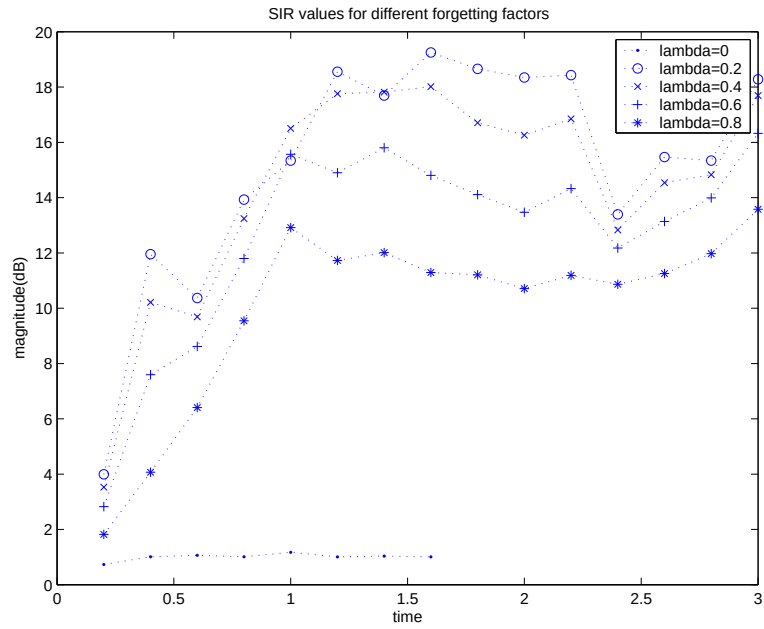


Figure 4.6. SIR Values for different forgetting factors

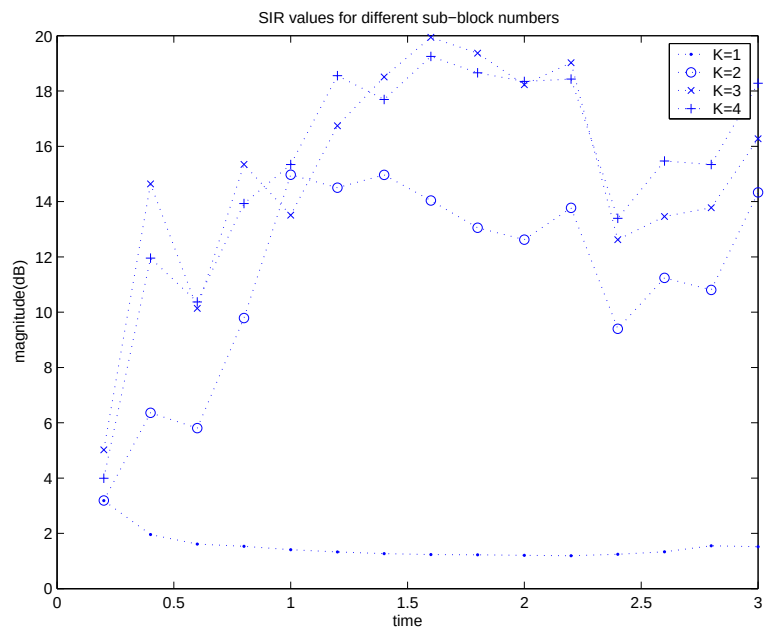


Figure 4.7. SIR Values for different subblock numbers

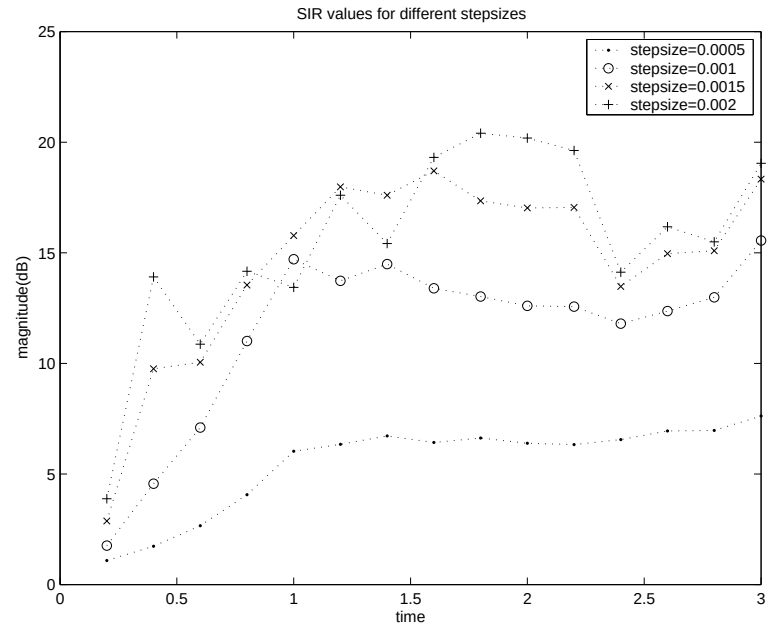


Figure 4.8. SIR Values for different stepsizes

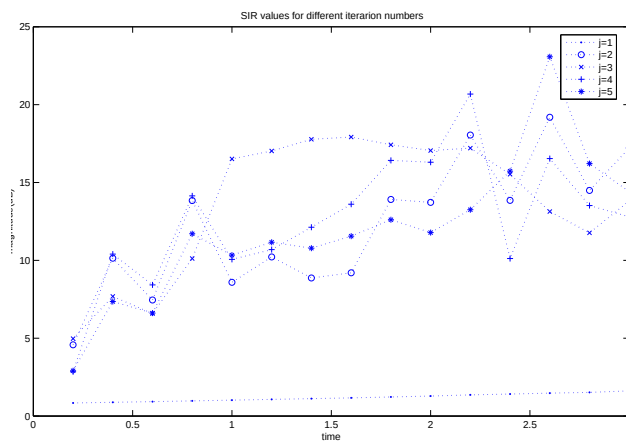


Figure 4.9. SIR values for different iteration numbers with adaptive stepsize

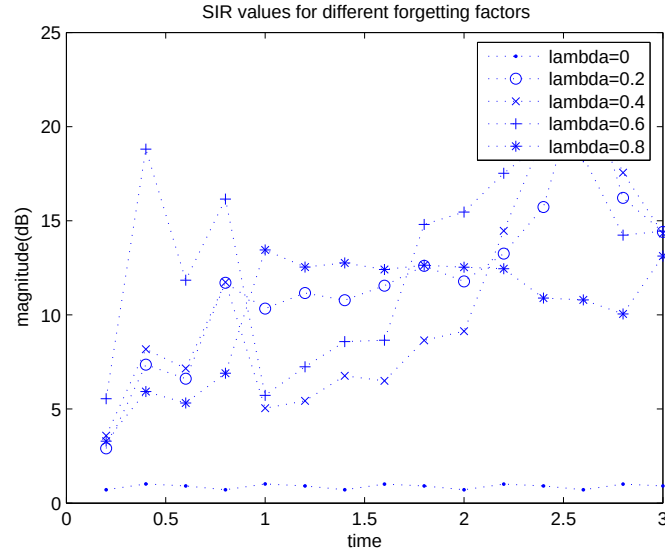


Figure 4.10. SIR Values for different forgetting factors with adaptive stepsize

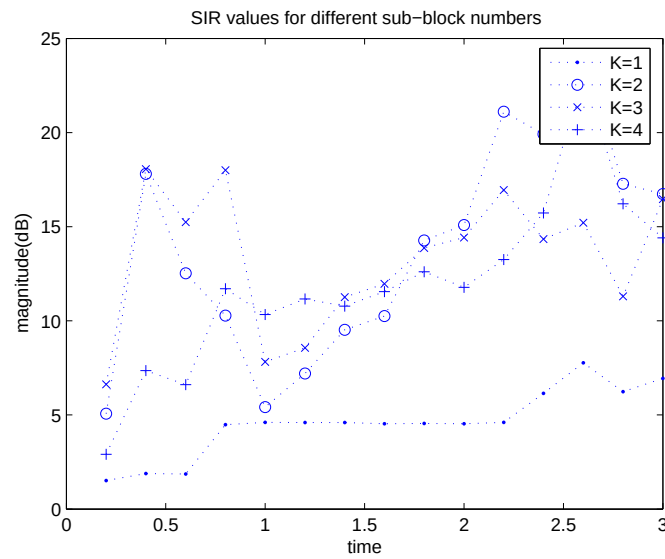


Figure 4.11. SIR values for different subblock numbers with adaptive stepsize

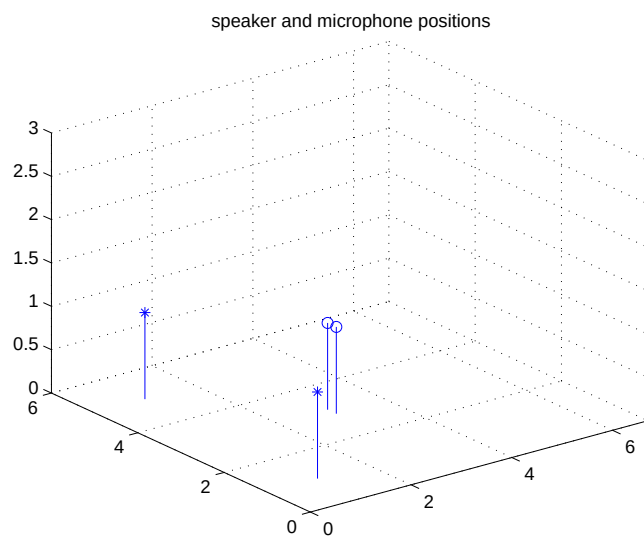


Figure 4.12. Speakers and microphones in the room in 3D

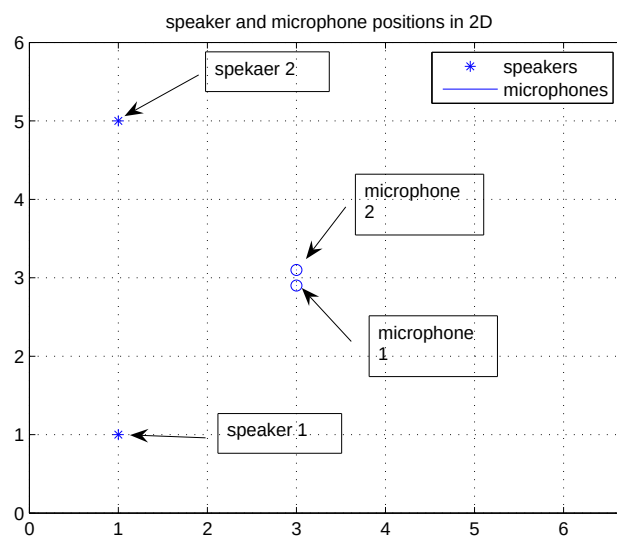


Figure 4.13. Speakers and microphones in the room in 2D

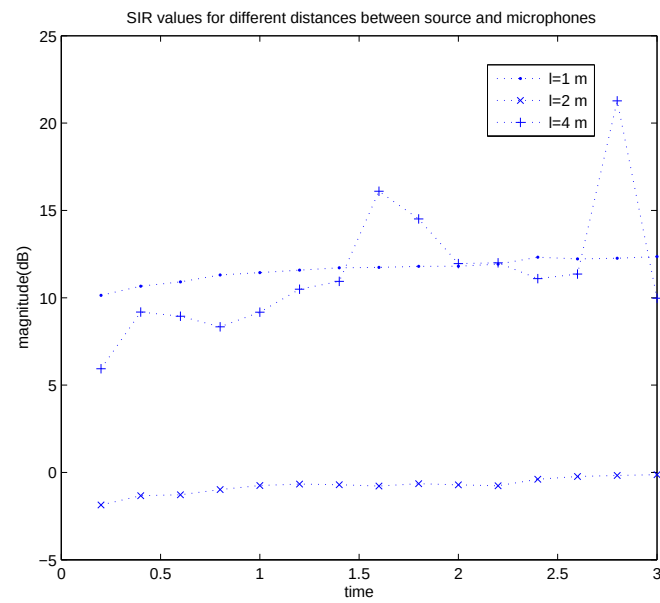


Figure 4.14. SIR values for different l values with fixed stepsize

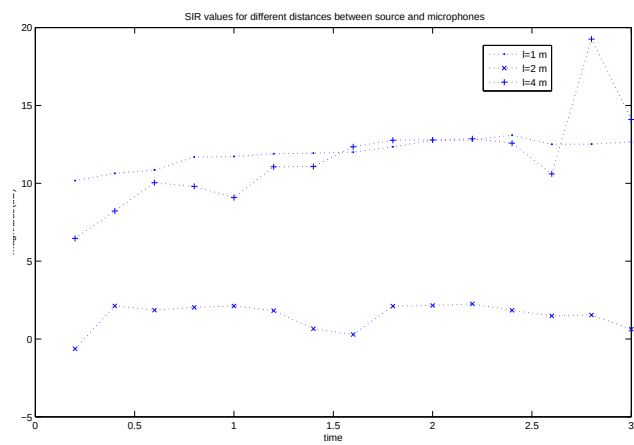


Figure 4.15. SIR values for different l values with adaptive stepsize

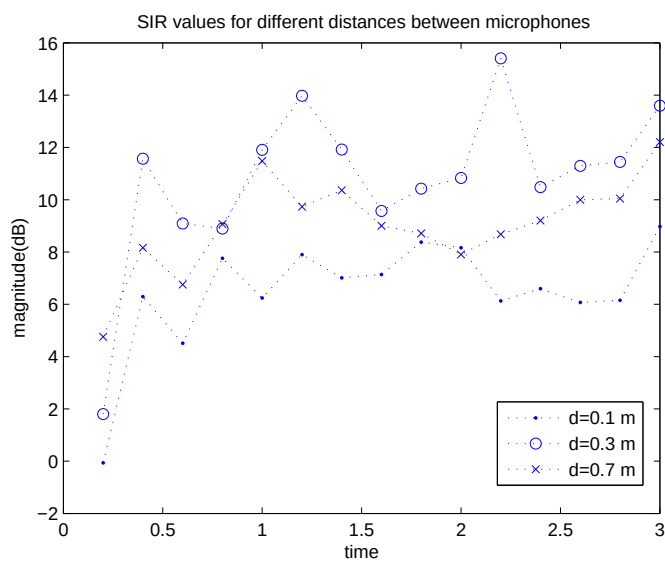


Figure 4.16. SIR values for different d values with fixed stepsize

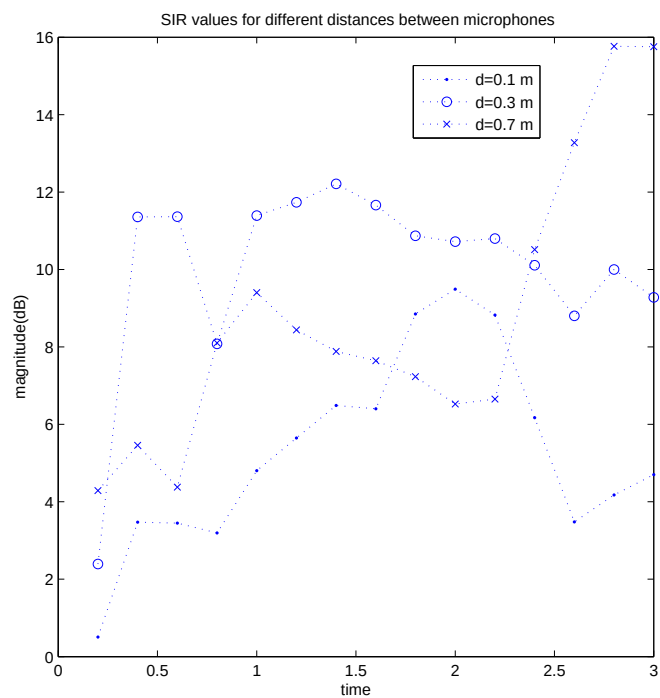
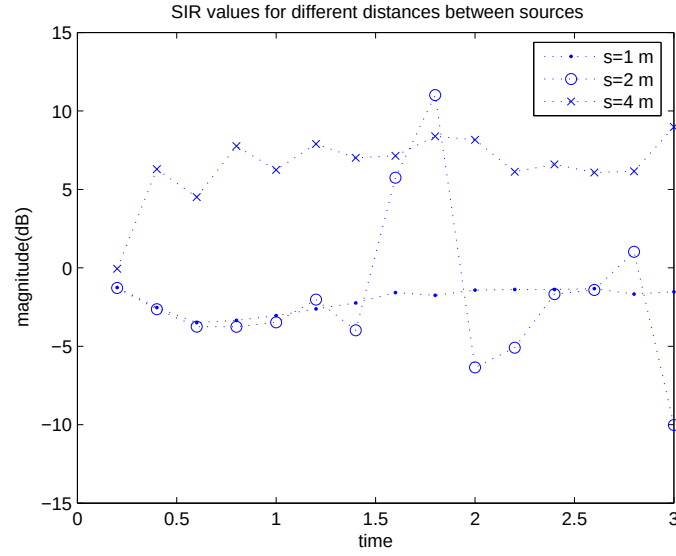
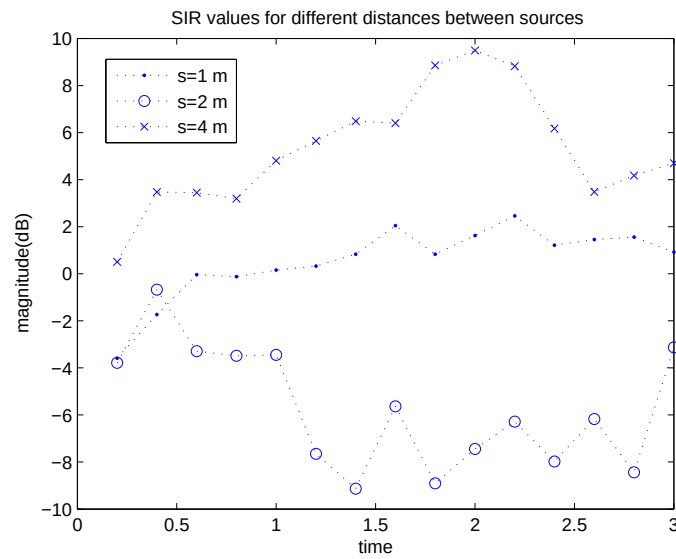


Figure 4.17. SIR values for different d values with adaptive stepsize

Figure 4.18. SIR values for different s valuesFigure 4.19. SIR values for different s values for adaptive stepsize

5. APPLICATION: IMPROVEMENT IN SPEECH RECOGNITION PERFORMANCE USING TIME-DOMAIN BSS

5.1. Speech Recognition Overview

Speech Recognition Systems provide us the transcription of the human speeches without using hand-writing or keyboard. Moreover, they are used in Human-Computer Interactions and Command and Control Systems. They mainly consist of two parts; Training and Decoding. In the training part, using training speech database and transcription of these speeches, model parameters that will be used in the decoding part are formed. In the decoding part, using that formed model parameters, input speech is tried to be decoded. We can model three different acoustic units to be used in decoding part; phones, triphones and words are major units that are used in the literature. It was shown that if adequate amount of training data is available, the model with words gives the best recognition results. However, since it is difficult to find such amount of training database and for every new word we need to have this words' training data, in large-vocabulary speech recognition applications, triphones are generally used as acoustic units. In this study, Sphinx [1], which has been developed in Carnegie Mellon University and based on Hidden Markov Models, is used. It consists of Sphinx Trainer and Sphinx Decoder parts. Sphinx Trainer needs

1. The acoustic signals
2. The corresponding transcriptions
3. A language dictionary
4. A filler dictionary

to form model parameters. Sphinx Decoder needs

1. The language dictionary

2. The filler dictionary
3. The language model
4. The test data

to decode the test data. We used triphones and words as acoustic units in this study and three-gram based language model is used. The dictionary consists of phones and words. Three different training databases (TDB) are formed using Turkish Telephone Speech (TURTEL) database. The properties of training databases are shown in Table 5.1

Table 5.1. Training Databases

Properties	TDB1	TDB2	TDB3
Number of speakers	65	65	65
acoustic unit	triphones	words	triphones
Database Duration(min)	40	40	185

5.2. Speech Recognition In Noisy Environments

As stated in previous section, speech recognition systems have training and decoding parts. Since we used model formed using training speech data, how much decoding and training databases match affects the performance of speech recognizer. Training databases are generally recorded as clean speech, that is, there is no noise in recording environment, so noise in speeches will be decoded can affect the recognition accuracy. In order to increase the recognition accuracy in noisy environments, there are two major methods can be applied, they are:

1. Adaptation of models formed with clean speech to noisy environment using some data from the noisy environment
2. Reduction of the noise from the speech

As the first method needs the adaptation of models to all environments with different signal-to-noise ratios, second model is more appropriate to apply.

5.3. BSS and Noise Reduction

Although BSS methods are found to separate the speeches of simultaneously speaking people in the same room, if we regard clean speech as one source and noise as another source, we can apply developed BSS method to extract the clean speech from the noisy speeches. However, separation performance depends on how good the noise signals hold the properties that are used in separation algorithm. These properties are non-stationarity and non-whiteness properties. Fortunately, noise in real environments generally is non-stationary and non-white. Therefore, studied method can be applied for noise suppression.

5.4. Simulation Results

The studied method is applied to synthetically noised speeches and effect on recognition performance is measured for 3 different noise type and for 3 different test databases. The noise is synthetically mixed with the speeches with level 0 dB and convolutively. Speech Recognition performance are measured in terms of word error rate (WER) and sentence error rate (SER). Test database includes 28 speakers and 15 sentences are uttered by the speakers. SER is measured for 420 sentences and WER is measured for 1904 words. For all types of models and for all types of noises, the method improved the recognition performance and results can be seen in Tables 5.2, 5.3 and 5.4. Since airplane noise is the most independent from the speech signal, WER with the separation methods is almost the same as WER with the clean speech. For Babble noise, the most improvements are obtained since Babble noise is the most similar noise to speech and reduces the performance at most.

Table 5.2. Airplane Noise Error Rates

	WER(%)			SER(%)		
	TDB1	TDB2	TDB3	TDB1	TDB2	TDB3
Clean	12.60	1.10	17.70	28.09	5.00	41.70
Noisy	34.60	13.60	34.60	64.76	26.90	61.91
Separated	19.40	0.80	19.00	45.23	5.00	42.86

Table 5.3. Babble Noise Error Rates

	WER(%)			SER(%)		
	TDB1	TDB2	TDB3	TDB1	TDB2	TDB3
Clean	12.60	1.10	17.70	28.09	5.00	41.70
Noisy	91.40	92.20	93.90	91.40	92.20	93.90
Separated	19.60	44.50	54.30	45.20	53.10	71.40

Table 5.4. Office Noise Error Rates

	WER(%)			SER(%)		
	TDB1	TDB2	TDB3	TDB1	TDB2	TDB3
Clean	12.60	1.10	17.70	28.09	5.00	41.70
Noisy	24.00	4.70	20.70	51.40	10.70	45.50
Separated	17.70	1.60	18.00	43.80	4.00	41.00

6. CONCLUSION AND FUTURE WORK

In this thesis, a time-domain BSS algorithm for convolutive mixtures is formulated and implemented. To separate the mixed sources, uncorrelatedness, nonstationarity and nonwhiteness of the audio signals are used. The cost function that uses these three properties together, is used to find the optimal demixing filters so that we have separated sources at the output. The demixing filters are updated with the cost function and Natural Gradient Approach. Moreover, by exploiting the Sylvester structure of the demixing matrix, the method is implemented with low computational complexity. Although the algorithm is formulated for the general case of P sources and P microphones, the performance measure and experiments are done for the 2 by 2 case. It was demonstrated that although performance of the method depends on the parameters that adjust the usage of the properties, for many parameter sets, its performance is good enough. For randomly created mixing parameters and mixing parameters which are created using Image Method, the performance is measured. Also, in image method experiments, the effect of geometrical position of the sources and microphones are investigated. Lastly, we proposed the usage of the method as a noise reduction method that is the contribution of this thesis and it was suggested that the method can be used a front-end processor for the speech recognition engines in order to improve the performance in noisy environments. As a future work, the separation performance must be measured in real acoustic environments to validate the success of the method. Moreover, in order to understand the effect of the method on speech recognition performance, larger train and test databases of noised speeches recorded in real acoustic environments must be used.

APPENDIX A: Natural Gradient Adaptation Algorithm

Although often not explicitly stated, the usefulness of the gradient descent depends on the structure of the cost function being optimized. The behavior of gradient descent is most useful for cost functions (i) that have a single minimum and (ii) whose gradients are isotropic in magnitude with respect to any direction away from this minimum. In practice, however, the cost function is being optimized is multimodel, and the gradient magnitudes are nonisotropic about any minimum. In such cases, the parameters are only guaranteed to locally minimize the cost function, and convergence to any local minimum can be extremely slow. The reason for these is that gradient descent is not considering the structure of the space that the cost function being optimized. Natural Gradient Adaptation, which is proposed by Amari in 1998, modifies the the standard gradient search direction according to the Riemannian structure of the parameter space. The chief drawback of this method is that it needs the knowledge to determine the Riemannian structure of the parameter space, however, it is shown that for certain estimation problems, it can be quite simple.

A.1. Euclidean and Riemannian Geometry

In order to understand natural gradient adaptation, we should understand Riemannian Geometry a little. If we asked ourselves, what is the distance between two points? We answered it is the length of the straight line that passes these two points, however, it is valid only if we assume that distance is Euclidean. In Euclidean terms, the distance between two points in N-dimensional space spanned by $\mathbf{v} = [v_1 v_2 \dots v_N]^T$ is

$$d_E(\mathbf{v}, \mathbf{v} + \delta\mathbf{v}) = \|\delta\mathbf{v}\|_2 \quad (\text{A.1})$$

$$= \sqrt{\sum_{i=1}^N \delta v_i^2} = \sqrt{\delta\mathbf{v}^T \delta\mathbf{v}} \quad (\text{A.2})$$

where $\delta \mathbf{v} = [\delta v_1 \delta v_2 \dots \delta v_N]^T$ and $\|\mathbf{v}\|_2$ denotes the L_2 or Euclidean norm of $\mathbf{v}$. This distance definition is valid with the Euclidean Geometry assumption. Actually, this definition is an approximation to a more complex notion of distance. For example, distance between two points from two sides of a mount is not the length of the straight line between these points, it is the length of the arc that connects the points. The Euclidean geometry was not found to adequately describe the fundamental physical laws was amazing to the scientific community, and it led to a field known as *differential* or *Riemannian Geometry*. Natural Gradient adaptation is based on Riemannian Geometry. In this new geometry, the distance definition is changed and the distance between two vectors $\mathbf{w}$ and $\mathbf{w} + \delta \mathbf{w}$ is

$$d_{\mathbf{w}}(\mathbf{w}, \mathbf{w} + \delta \mathbf{w}) = \sqrt{\sum_{i=1}^N \sum_{j=1}^N \delta w_i \delta w_j g_{ij}(\mathbf{w})} \quad (\text{A.3})$$

$$= \sqrt{\delta \mathbf{w}^T \mathbf{G}(\mathbf{w}) \delta \mathbf{w}} \quad (\text{A.4})$$

where $\mathbf{G}(\mathbf{w})$ is the Riemannian metric tensor matrix that is an NxN positive definite matrix. This matrix characterizes the intrinsic curvature of a particular manifold in N-dimensional space. For Euclidean coordinate system, variations in each component of $\mathbf{w}$ are orthogonal to each other and $\mathbf{G}(\mathbf{w}) = \mathbf{I}$ is the identity matrix. We can describe the nature of the manifold in terms of a transformation of Euclidean orthogonal space with coordinate vector $\mathbf{v}$ to $\mathbf{w}$ and determine the form of $\mathbf{G}(\mathbf{w})$ through the relationship

$$d_E(\mathbf{v}, \mathbf{v} + \delta \mathbf{v}) = d_{\mathbf{w}}(\mathbf{w}, \mathbf{w} + \delta \mathbf{w}) \quad (\text{A.5})$$

where $\delta \mathbf{v}$ is small and $\mathbf{w} + \delta \mathbf{w}$ is transformed value of $\mathbf{v} + \delta \mathbf{v}$.

Examples:

1. For Polar Coordinate System, Riemannian Metric Tensor Matrix is

$$\mathbf{G}(\mathbf{w}) = \begin{pmatrix} 1 & 0 \\ 0 & r^2 \end{pmatrix} \quad (\text{A.6})$$

2. For Spherical Coordinate System, Riemannian Metric Tensor Matrix is

$$\mathbf{G}(\mathbf{w}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & r^2 & 0 \\ 0 & 0 & r^2 \sin^2 \theta \end{pmatrix} \quad (\text{A.7})$$

3. For Linearly Transformed Euclidean Space with transformation matrix $\mathbf{T}$, Riemannian Metric Tensor Matrix is

$$\mathbf{G}(\mathbf{w}) = \mathbf{T}^T \mathbf{T} \quad (\text{A.8})$$

A.2. Natural-Gradient Descent

In gradient adaptation methods, we calculate the gradient of a cost-function surface iteratively and then change the current parameter estimates according to this gradient estimate. Let us assume that the cost function is

$$J(\mathbf{w}) = c \|\mathbf{w} - \mathbf{w}_{opt}\|_2^2 \quad (\text{A.9})$$

where c is an arbitrary constant, then its standard gradient is

$$\frac{-\partial J(\mathbf{w}(k))}{\partial \mathbf{w}} = -2c(\mathbf{w} - \mathbf{w}_{opt}) \quad (\text{A.10})$$

and a step-size value of $\mu = (1/2c)$ allows the steepest descent procedure to converge to $\mathbf{w}_{opt}$ in one step, irrespective of $\mathbf{w}(k)$ because of the fact that negative gradient of

this particular cost function at any $\mathbf{w}$ points towards the minimum of the error surface. However, this is not the case in general and thus a gradient in Euclidean space of the parameters is not the best for searching this cost function. Since the underlying space of the parameters is not Euclidean and it is curved and distorted, that is, Riemannian, and the negative of the standard gradient $-\partial J(\mathbf{w})/\partial \mathbf{w}$ does not represent the steepest-descent direction of the cost function. We have to use the information about how the parameter space is curved and distorted. The Riemannian Metric Tensor is the source of that information. Therefore, we should change the steepest-descent direction from the direction of $J(\mathbf{w})$ at $\mathbf{w}$ to the direction of the direction vector $\partial \mathbf{w}$ that minimizes $J(\mathbf{w} + \delta \mathbf{w})$ under the constraint

$$d_{\mathbf{w}}(\mathbf{w}, \mathbf{w} + \delta \mathbf{w}) = \sqrt{\delta \mathbf{w}^T \mathbf{G}(\mathbf{w}) \delta \mathbf{w}} = \varepsilon \quad (\text{A.11})$$

when ε is small. With some variational calculus, we have

$$\delta \mathbf{w} = -\mu \mathbf{G}^{-1}(\mathbf{w}) \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \quad (\text{A.12})$$

where μ is a positive constant. Now we can define an alternative iterative procedure for minimizing $J(\mathbf{w})$ which we called it as *natural gradient adaptation*. For any suitably smooth gradient-searchable cost function $J(\mathbf{w}(k))$, natural-gradient adaptation is defined as

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu(k) \mathbf{G}^{-1}(\mathbf{w}(k)) \frac{\partial J(\mathbf{w}(k))}{\partial \mathbf{w}} \quad (\text{A.13})$$

where $\mathbf{G}(\mathbf{w})$ is the Riemannian metric tensor matrix for the parameter vector $\mathbf{w}$. The natural-gradient search method applies the notion of curved parameter space to the gradient-adaptation task. For parameter space that is curved, the standard-gradient direction is no longer appropriate for use within a gradient search method. The natural-gradient direction uses the information knowledge of the Riemannian distance structure of the parameter space to alter the gradient direction, thus providing fast and accurate adaptation behavior.

A.3. Application of Natural Gradient to Instantaneous Blind Source Separation

In section 2.1 we described the Instantaneous Blind Source Separation problem in detail. Here, the question of how natural-gradient adaptation can be used to obtain an efficient way of solving this problem will be answered. In [30], it is shown that the (i,j,k,l)th entry of the Riemannian metric tensor in the space of separating matrices $\mathbf{W}$ is

$$[\mathbf{G}(\mathbf{W})]_{ij,kl} = \sum_{m=1}^n \delta_{ij} [\mathbf{W}^{-1}]_{jm} [\mathbf{W}^{-1}]_{lm} \quad (\text{A.14})$$

where δ_{ij} is the Kronecker delta function, and the natural-gradient algorithm for this task is

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \mu(k) \frac{\partial J(\mathbf{W}(k))}{\partial \mathbf{W}} \mathbf{W}^T(k) \mathbf{W}(k) \quad (\text{A.15})$$

If we use standard-gradient algorithm we need to inverse the separation matrix to update but now we do not need it. Moreover, it is also shown that natural-gradient performance is improved over the standard-gradient performance [30].

REFERENCES

1. <http://www.cs.cmu.edu/robust/tutorial>.
2. Bell A. and T. Sejnowski, *Information-maximization approach to blind separation and blind deconvolution*, Neural Computation **vol. 7** (1995), 1125–59.
3. Cichocki A. and R. Unbehauen, *Neural networks for optimization and signal processing*, Wiley, Chichester, 1994.
4. Hiroe A., *Solution of permutation problem in frequency domain ica, using multivariate probability density functions*, Lecture Notes in Computer Science (2006), pp. 601–608.
5. Hyvaerinen A., J. Karhunen, and E. Oja, *Independent component analysis*, John Wiley & Sons, 2001.
6. Hyvarinen A. and E. Oja, *A fast fixed-point algorithm for independent component analysis*, Neural Computation **vol. 9** (1997), 1483–92.
7. Mansour A., N. Benckroun, and C. Gervaise, *Blind separation of underwater acoustic signals*, in ICA06, 2006.
8. Oppenheim A., *Inequalities connected with definite hermitian forms*, J. London Math. Soc. **vol. 5** (1930), pp. 114–119.
9. Allen J. B. and D. A. Berkley, *Image method for efficiently simulating small-room acoustics*, J. Acoust. Soc. Amer. 65 (1979), no. 4, pp. 943–950.
10. Van Veen B. and K. Buckley, *Beamforming: a versatile approach to spatial filtering*, IEEE Trans. Acoust., Speech, Signal Process. (1988), pp. 4–24.
11. Cherry E. C., *Some experiments on the recognition of speech, with one and two*

- ears, J. Acoust. Soc. Am. **vol. 25** (1953), pp. 975–979.
12. Jutten C. and J. Herault, *Space or time adaptive signal processing by neural network models*, in Proc. AIP Conf., May 1986, pp. 206–11.
 13. Markel J. D. and A. H. Gray, *Linear prediction of speech*, Springer Verlag, 1976.
 14. Schobben D., K. Torkkola, and P. Samaragdis, *Evaluation of blind signal separation methods*, in: Proc. Int. Symposium on Independent Component Analysis and Blind Signal Separation (ICA), Jan. 1999, pp. 261–266.
 15. Weinstein E., M. Feder, and A. Oppenheim, *Multi-channel signal separation by decorrelation*, IEEE Trans. Speech Audio Process **vol. 1** (1993), 405–13.
 16. Haykin S. (ed.), *Unsupervised adaptive filtering, volume 1 blind source separation*, John Wiley & Sons, 2000.
 17. Buchner H., R. Aichner, and W. Kellermann, *A generalization of a class of blind source separation algorithms for convolutive mixtures*, in: Proceedings of the International Symposium on Independent Component Analysis and Blind Signal Separation (ICA), Nara, Japan, April 2003, pp. 945–950.
 18. Buchnner H., R. Aichner, and W. Kellermann, *A generalization of blind source separation algorithms for convolutive mixtures based on second-order statistics*, IEEE Trans. Speech Audio Processing **vol. 13** (2005), no. 1, pp. 120–134.
 19. Sawada H., S. Araki, R. Mukai, and S. Makino, *Blind extraction of a dominant source signal from mixtures of many sources using ica and time-frequency masking*, in: Proc. Int. Symp. Circuits and Systems, 2005, pp. 5882–5885.
 20. Lee I., T. Kim, and T.-W. Lee, *Complex fastiva: A robust maximum likelihood approach of mica for convolutive bss*, Lecture Notes in Computer Science (2006), pp. 625–632.

21. Cardoso J.-F., *Blind signal separation: statistical principles*, in Proc. IEEE '98 **vol. 9** (1998), pp. 2009–2025.
22. Matsuoka K., M. Ohya, and M. Kawamoto, *A neural net for blind separation of nonstationary signals*, Neural Computation **vol. 8** (1995), 411–19.
23. Torkkola K., *Blind separation of convolved sources based on information maximization*, in: Proc. of the 1996 IEEE Signal Processing Society Workshop, 1996, pp. 423–32.
24. Parra L. and C. Alvino, *Geometric source separation: merging convolutive source separation with geometric beamforming*, IEEE Trans. on Speech and Audio Processing **vol. 10** (2002), no. 6, pp. 352–362.
25. Zibulevsky M. and B.A. Pearlmutter, *Blind source separation by sparse decomposition in a signal dictionary*, Neural Computation **vol. 13** (2001), 863–882.
26. Vogl T. P., J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Allcon, *Accelerating the convergence of the back propagation method*, Biol. Cybern. 59 (1988), pp. 257–263.
27. Aichner R., S. Araki, S. Makino, T. Nishikawa, and H. Saruwatari, *Time-domain blind source separation of nonstationary convolved signals by utilizing geometric beamforming*, in: IEEE International Workshop on Neural Networks for Signal Processing (NNSP), Martigny, Switzerland, Sept. 2002, pp. 445–454.
28. Aichner R., H. Buchner, and W. Kellermann, *A real-time blind source separation scheme and its application to reverberant and noisy environments*, Signal Processing **vol. 86** (2006), 1260–1277.
29. Aichner R., H. Buchner, F. Yan, and W. Kellermann, *Real-time convolutive blind source separation based on a broadband approach*, in Int. Symp. Independent Component Analysis and Blind Signal Separation (ICA), Sept. 2004, pp. 833–840.

30. Amari S., A. Cichocki, and H. H. Yang, *A new learning algorithm for blind signal separations*, in Advances in Neural Information Processing Systems. Cambridge, MA: MIT **vol. 8** (1996), pp. 757–763.
31. Araki S., *The fundamental limitation of frequency-domain blind source separation for convolutive mixtures of speech*, IEEE Trans. Speech Audio Process **vol. 11** (2003), 109–116.
32. Araki S., S. Makino, A. Blin, R. Mukai, and H. Sawada, *Blind separation of more speech than sensors with less distortion by combining sparseness and ica*, in Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC), Kyoto, Japan, Sept. 2003, pp. 271–274.
33. Araki S., S. Makino, R. Mukai, and H. Saruwatari, *Fundamental limitation of frequency domain blind source separation for convolutive mixture of speech*, in Proc. ICAASP '01, May 2001.
34. Haykin S. and Z. Chen, *The cocktail party problem*, Neural Computation **vol. 17** (2005), pp. 1875–1902.
35. Kurita S., H. Saruwatari, and S. Kajita, *Evaluation of blind signal separation method using directivity pattern under reverberant conditions*, in: Proc. Int. IEEE ICASSP, 2000, pp. 3140–3143.
36. Roberts S. and R. Everson, *Independent component analysis: Principles and practice*, Cambridge University Press, 2001.
37. Kim T., H. T. Attias, S.-Y. Lee, and T.-W. Lee, *Blind source separation exploiting higher-order frequency dependencies*, IEEE Trans. on Speech and Audio Processing **vol. 15** (2007), pp. 70–79.
38. Lee T., *Independent component analysis*, Kluwer Academic Publishers, 1998.
39. Nishikawa T., H. Saruwatari, and K. Shikano, *Comparison of time-domain ica*,

frequency-domain ica and multistage ica for blind source separation, in: Proceedings of the European Signal Processing Conference, Sept. 2002, pp. 15–18.