Multichannel blind adaptive source separation in the frequency domain

van Vugt, D.W.

Award date:
1998

7579

TECHNISCHE UNIVERSITEIT EINDHOVEN

FACULTEIT ELEKTROTECHNIEK

LEERSTOEL Signaalverwerking

esp
SIGNAL
PROCESSING

Multichannel blind adaptive source separation in the frequency domain

door

D.W. van Vugt

ESP-05-98

Verslag van een afstudeeronderzoek
verricht binnen de leerstoel ESP
o.l.v. dr.ir. P.C.W. Sommen
en ir. D.W.E. Schobben
in de periode september 1997 - mei 1998

Eindhoven, 11 juni 1998

Abstract

In this graduation report a frequency domain blind source separation algorithm is presented. The sources, e.g. two persons talking in the same room at the same time, are recorded by sensors, e.g. microphones. These observed signals are convolutive mixtures of the original sources, due to delays, echos and filtering by the room. These observed mixtures are transformed to the frequency domain in order to reduce the convolutive mixing problem to an instantaneous mixing problem in such a way that separation is performed for every freqeuncy bin. This frequency transformation is implemented using the overlap-save method. An algorithm developed for instantaneous mixtures is then modified for use in the frequency domain.

The frequency domain algorithm is capable of separating convolutive mixtures. To improve the frequency spectrum of the outputs, a normalisation is introduced. This normalisation is placed outside the update cycle of the algorithm and is used only for optimising the frequency spectrum of the outputs. The updating is a learning rule based on an information theoretic approach, that achieves separation by maximising the entropy of the outputs, and thereby minimising the mutual information between the outputs. Other approaches to blind source separation are briefly discussed.

The algorithm is suitable for separation of multiple sources. Although in this report only simulations have been done using two sources, the algorithm is not restricted to two sources. The developed algorithm is such that it can be implemented in real-time.

Contents

Chapter 1

Introduction

1.1 Objective

The research presented in this report is a graduation project at the Signal Processing Group, department of Electrical Engineering at the Eindhoven University of Technology. The objective of the research is to develop an algorithm that performs multichannel blind adaptive source separation. A well-known application is the so-called cocktail-party problem: e.g. two persons are talking at the same time in the same room and they are recorded by two microphones. The algorithm then separates both speakers in real time, so other persons can listen to the separated speakers. So the sources that have to be separated can be speech signals. The next section will give an introduction to the blind source separation problem.

1.2 Problem Statement

In the past few years the signal processing problem called *blind source separation* has received much attention. It can be used in many applications, such as noise cancellation, speech enhancement, hearing aids and teleconferencing. In a more popular term the problem is sometimes referred to as a *cocktail party* problem. This refers to a cocktail party where a lot of people are talking at the same time in the same room, and a lot of background noise such as music is also present. People who are having a conversation are capable of 'filtering out' all irrelevant sources, so that they are able to concentrate on the person they are talking to.

Blind source separation attempts to separate independent sources from their mixtures only. The mixtures are signals as observed by sensors, for example microphones. The problem is that there is no a priori knowledge of the characteristics of the transmission channel from the sources to the sensors. Because there are no reference signals of the original sources, the problem is called blind.

In general the problem of blind source separation can be referred to as a $M \times N$ problem, which means that there are M sources and N sensors. Typically for most situations is that $M = N$. This is however not always the case. Sometimes the number of sources is unknown. Another (more difficult) problem is the case where there are less sensors than sources.

For the 2x2-case, which means 2 sources x 2 sensors, a possible situation is depicted in Figure 1.1. $s_1(t)$ and $s_2(t)$ are the two sources, for example two persons speaking at the same time. The observed signals at the two microphones ($x_1(t)$ and $x_2(t)$) are mixtures of the two sources. So the objective is to extract the original sources $s_1(t)$ and $s_2(t)$ from the observed signals $x_1(t)$ and $x_2(t)$. The algorithm has to be developed such that the outputs of the separator ($y_1(t)$ and $y_2(t)$) will be filtered versions of the original sources. It is not possible to reconstruct the original sources perfectly, because the characteristics of the transmission channel are unknown. In the case

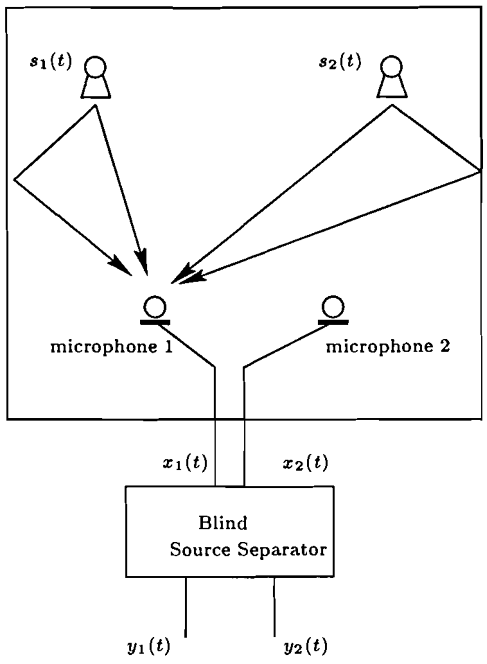

Figure 1.1: 2x2 situation

of two persons speaking the objective is to hear one speaker at $y_1(t)$ and the other at $y_2(t)$.

From now on we assume that the number of sources, the number of sensors and the number of outputs are all the same. Let S be a vector containing n sources, **X** a vector containing n observed mixtures and **Y** a vector containing the n outputs, as can be seen in (1.1).

$$\mathbf{S} = \begin{pmatrix} s_1(t) \\ \vdots \\ s_n(t) \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix} \text{and}\, \mathbf{Y} = \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix} \tag{1.1}$$

Furthermore let **A** be the mixing matrix. This matrix contains a transfer function for every source i to every sensor j. So in the 2x2 case **A** is a 2x2 matrix with elements a_{ij} , with a_{ij} the response from source j to sensor i. This leads to

$$\mathbf{X} = \mathbf{AS} \tag{1.2}$$

The goal is to find an unmixing matrix **W** so that **Y** = **S**.

$$\mathbf{Y} = \mathbf{WX} = \mathbf{WAS} \tag{1.3}$$

This mixing and unmixing system can be seen in Figure 1.2. So to make **Y** = **S**, the unmixing matrix has to become $\mathbf{W} = \mathbf{A}^{-1}$. Then (1.3) becomes $\mathbf{Y} = \mathbf{A}^{-1}\mathbf{AS} = \mathbf{S}$. In general, the exact inverse of the mixing matrix will not be found. So **WA** will not be the identity matrix, but a matrix which is permuted and scaled (at least if separation is achieved). Such a permuted and scaled matrix has only one high value in each row and column. An example of a 3x3 permuted and scaled matrix **WA** that can be found is

$$\mathbf{WA} = \begin{pmatrix} 0.97 & 0.03 & 0.11 \\ 0.02 & 0.08 & 0.89 \\ 0.13 & 0.93 & 0.01 \end{pmatrix} \tag{1.4}$$

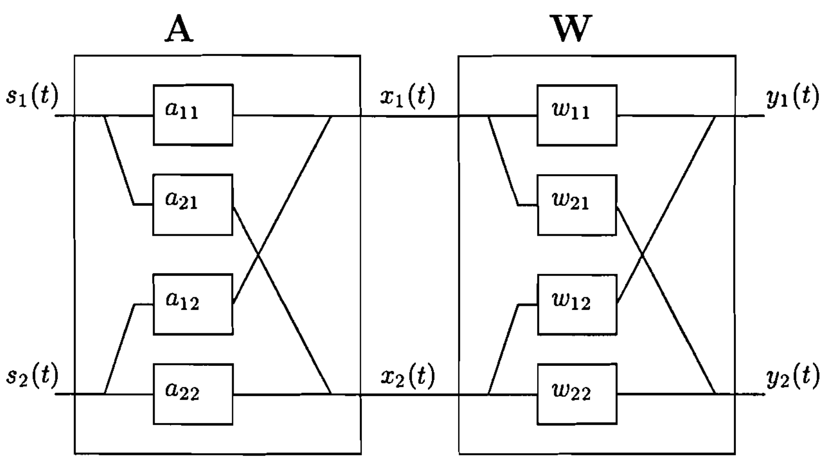

Figure 1.2: Mixing (**A**) and unmixing (**W**) system

Several approaches have been developed in order to find the unmixing matrix **W**. A discussion is given in the next section.

1.3 Approaches to Blind Source Separation

How can independent sources be separated from their mixtures only? In the literature there are several approaches described, each of them with their own specific applications and (dis)advantages. In this section some of these approaches will be discussed and at the end one particular approach will be chosen.

1.3.1 Output Decorrelation

The first approach tries to achieve separated outputs by using output decorrelation. Output decorrelation assumes that the sources are statistically independent and zero-mean. It tries to obtain uncorrelated outputs by minimising the square of a number of cross-correlation lags. I.e. it tries to minimise

$$(E[y_i(n)y_j(n+l)])^2 \tag{1.5}$$

where $i \neq j$, $l = l_1, ..., l_2$ with l_1 and l_2 determining the range of the cross-correlation lags and $E[.]$ denotes mathematical expectation. This method uses second order moments only. This means that in theory it has less separating capabilities than higher order methods. But the fact that it uses only second order moments makes it 'simple' and fast. This is because the calculation of higher order moments is computationally intensive. When the separation has to run in real-time, this becomes an important issue. Chan [30] proposed an algorithm for $n \times n$ convolutive mixtures (convolutive mixtures will be discussed later) using output decorrelation as the separation criterion. Chan uses a constraint in order to prevent **W** from collapsing (to zero) called the Constant Diagonal Constraint. The weight update for the elements of the unmixing matrix **W** using this constraint is

$$\underline{w}_{lm} = -\left(\sum_{j\neq l}\sum_{c=1}^{n}\sum_{d=1}^{n}\mathbf{A}_{jc}^T\mathbf{R}_{x_m x_c}\mathbf{R}_{x_m x_d}^T\mathbf{A}_{jd}\right)^{-1}\left(\sum_{j\neq l}\sum_{c=1}^{n}\sum_{b\neq m}\sum_{d=1}^{n}\mathbf{A}_{jc}^T\mathbf{R}_{x_m x_c}\mathbf{R}_{x_b x_d}^T\mathbf{A}_{jd}\underline{w}_{lb}\right) \tag{1.6}$$

where $\mathbf{A}_{jc}$ is a matrix which is a function of $\underline{w}_{jc}$ only, $\mathbf{R}_{x_a x_c}$ is a matrix which is a function of the cross-correlation of input $x_a(n)$ and $x_c(n)$. $\underline{w}_{lm}$ is iteratively updated according to (1.6) until the costfunction

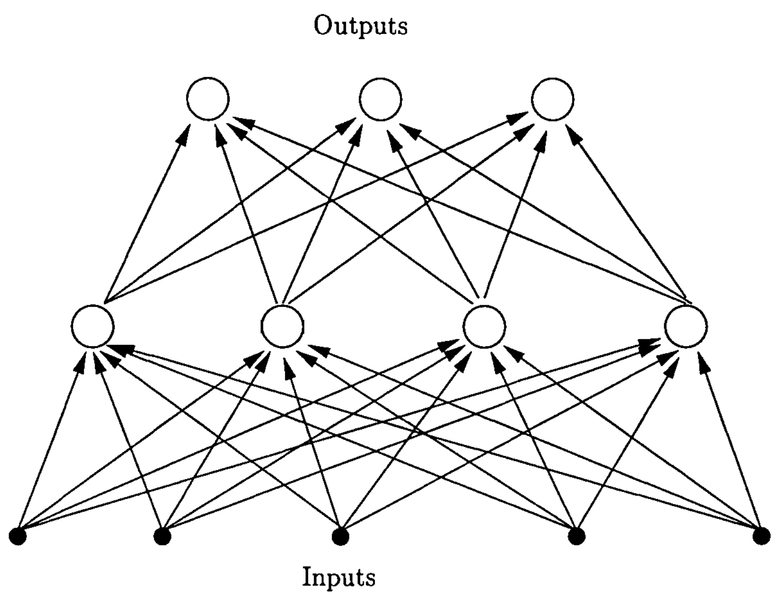

Figure 1.3: A two layer neural network

$$C = \sum_{i=1}^{n} \sum_{j=1,j\neq l}^{n} \sum_{a,b,c,d=1}^{n} \underline{w}_{ia}^T \mathbf{A}_{jc}^T \mathbf{R}_{x_a x_c} \mathbf{R}_{x_b x_d}^T \mathbf{A}_{jd} \underline{w}_{ib} \tag{1.7}$$

is minimised with respect to all $\underline{w}_{lm}$ and convergence is achieved. For more details on this algorithm see Chan [30]. The results Chan presented using his algorithm were not very satisfying, especially for large filters. There was some separation audible, but still the suppressed source was present. The algorithm also is not so fast and it uses a matrix R which is a function of the cross-correlation of the inputs. This matrix is computed off-line. One of the objectives of the algorithm developed in this report is to let it run in real-time. Therefore this algorithm is not suited.

1.3.2 Neural Networks

A way of introducing Higher Order Statistics, and thereby using more information available in the signals than by using second order statistics only, is by using neural nets. This approach is very much related to the previous one. Neural networks make use of non-linear functions. These non-linear functions are also called activation functions. They introduce the higher order moments. Neural networks can have multiple layers. In Figure 1.3 a two layer neural network is given. The neurons (circles) are the non-linear functions ($f(n_i)$) with weighted inputs (w_{ij}) as can be seen in Figure 1.4. The weighted sum of the inputs (=the outputs of previous neurons) are passed through a non-linear function to produce the new output of this particular neuron, which in general will also be used as an input to another neuron (multiple layers).

The weights are adaptively adjusted by learning rules. This approach is very interesting, because it tries to give a link to human perception or the way the human brain works. Recalling the cocktail party problem, it is amazing that a human is capable of 'extracting' one particular source in a room full of talking people and background noise.

In the literature researchers have come to good results using neural nets. Cichocki & Unbehauen [18] developed the following robust, efficient and fast time-continuous algorithm:

$$\Delta \mathbf{W} = \lambda(t)\{\mathbf{\Lambda} - f(\mathbf{Y})g^T\mathbf{Y}\}\mathbf{W} \tag{1.8}$$

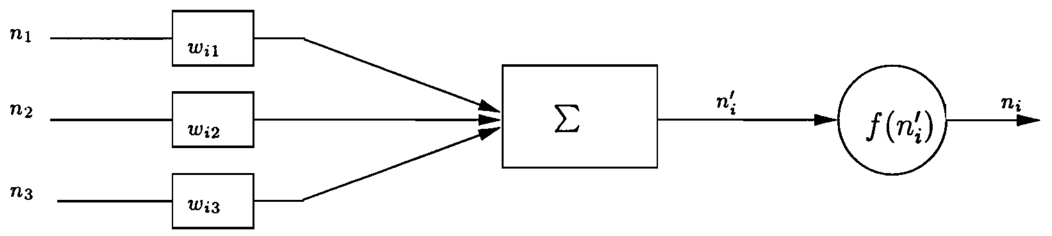

Figure 1.4: Schematic diagram of a neuron

where $\mathbf{W}$ is the unmixing matrix, $\Delta\mathbf{W}$ the update for $\mathbf{W}$, $\lambda(t) > 0$ the learning rate which can vary in time, $\mathbf{\Lambda} = \text{diag}\{\mu_1, \mu_2, ..., \mu_n\}$ is a diagonal matrix with amplitude scaling factors $\mu_i > 0 \, \forall i$, $f(\mathbf{Y})$ and $g(\mathbf{Y})$ non-linear functions with $\mathbf{Y}$ a column vector containing all outputs $y_i(t)$ as can be seen in (1.1).

This algorithm gives good performance, but only for stationary signals. For non-stationary signals the algorithm may fail to converge because the weights w_{ij} depend not only on the characteristic of the mixture, but also on the energy of the signals. This makes this algorithm less suitable for speech separation.

1.3.3 Information Theoretic

A different approach is the information theoretic approach. The idea behind this is to maximise the mutual information between the inputs and outputs by maximising the entropy of the outputs alone. By doing this the mutual information between the outputs will be minimised. This will lead to separation. If the outputs have no or little mutual information, this means that the outputs are independent of each other and that separation is achieved.

Bell & Sejnowski [16] developed an *infomax* approach. Infomax is an abbrevation of information maximisation. Their algorithm is given in (1.9).

$$\Delta\mathbf{W} = \lambda\{[\mathbf{W}^T]^{-1} + \mathbf{X}(\mathbf{1} - 2\mathbf{V})^T\} \tag{1.9}$$

where $\mathbf{W}$ is the unmixing matrix, λ is the learning rate, $\mathbf{1}$ is a vector of ones, $\mathbf{X}$ is a column vector as in (1.1) and $\mathbf{V}$ as in (1.10).

$$\mathbf{V} = (\mathbf{1} + e^{-\mathbf{WX}})^{-1} \tag{1.10}$$

This algorithm will be extensively discussed in Chapter 2. The algorithm in (1.9) was the foundation of further research. A lot of researchers have extended this algorithm. For example Amari [3] removed the computationally intensive matrix inversion in Bell's algorithm to obtain

$$\Delta\mathbf{W} = \lambda\{\mathbf{I} - f(\mathbf{Y})\mathbf{Y}^T\}\mathbf{W} \tag{1.11}$$

where λ is the learning rate and $f(\mathbf{Y})$ is a non-linear function of $\mathbf{Y}$ with $\mathbf{Y} = \mathbf{WX}$.

Over the last few years researchers using the information theoretic approach have come to good performances in separating independent sources.

1.3.4 Other Approaches

There have been several other approaches developed for source separation.

In order to use all of the information available in the signals, Higher Order Statistics (HOS) can be used. As mentioned before, this is computationally intensive. Furthermore it's difficult to estimate higher order moments reliable. This will become even more difficult when the signals are non-stationary (like for example speech). And since one of our objectives is to separate speech and another one to do this in real-time, this approach doesn't seem to fit our objectives. Especially considering the fact that only very few researchers in general (using any approach) claimed to have developed an algorithm that works in real-time.

Puntonet et al. [14] for example developed a geometry-based approach. This works purely on geometrical considerations concerning the mixtures. For the 2x2 case a polygon representing the mixtures is tried to turn into a rectangle by an unmixing matrix. If a rectangle is achieved, these output signals form an orthonormal base. Then the output signals are independent and thus separated. Furthermore they claim that the sources do not have to be statistically independent, what is interesting because most other approaches assume statistical independence of the sources.

Another approach is beam-forming. Beam-forming uses an array of microphones. The idea is to make these microphones more sensitive towards the direction of the sources. By doing this beams are formed pointing towards the sources. The problem that occurs with this approach are the reflections from the walls. The path from a source to the microphones is not just a direct path, but the microphones also pick up reflections of the sources from the walls. These reflections come from different directions. Therefore even if the algorithm is able to 'choose' the right direction, it will also pick up reflections from the other sources.

The most important approaches have been briefly discussed in this section. In the next section the choice for a particular approach will be discussed.

1.4 Chosing An Approach

It's not possible to study all approaches thoroughly within the time set for this research project. Therefore a relatively short study of the literature has been done. Following this study a choice was made on which approach will be used. Output decorrelation has the advantage that is it fast and quite simple, so an implementation in real-time is possible. But the results of Chan obtained with output decorrelation are not very impressive and restricted to short filters. Listening to the results the suppressed source was still heard clearly. Furthermore Chan's algorithm uses a priori knowledge of the cross-correlation of the inputs. This is computed off-line and therefore his algorithm is not useful for real-time implementation.

Higher Order Statistics requires too much computational effort to calculate the higher order moments. The estimation of these higher order moments is not very reliable, especially for non-stationary signals such as speech. This makes Higher Order Statistics less suitable for the objectives stated in Section 1.1.

Neural Networks are in general very complex. They can have multiple layers. This approach is very interesting. It tries to give a link to human perception. Results using this approach are promising, separation is achieved. The algorithm (1.8) gives 'perfect' separation, but only for stationary signals. For non-stationary signals such as speech, it may not converge. However neural nets seem a very interesting approach.

The information theoretic approach gives excellent results. Bell [16] even separates up to ten sources. This approach is easy to implement. A lot of researchers extended Bell's algorithm (1.9) to improve it even more. Simulations are done in order to verify the working of the algorithm and the results were perfect. The suppressed sources could not be heard at all. Even for very bad conditioned mixing matrices such as $\mathbf{A} = \begin{pmatrix} 1 & 0.001 \\ 1 & 0.002 \end{pmatrix}$ separation is achieved. Listening to the mixtures, only one source could be heard because that source was 500 times as loud as the other source. But after passing these mixtures through the algorithm, both sources were separated perfectly.

Only two approaches seem to be interesting for our objectives, the Neural Networks and the Information Theoretic approach. Because of the impressive separating capabilities and easy implementation of Bell's algorithm and the high complexity of Neural Networks, the Information Theoretic approach is chosen. This Information Theoretic approach will be discussed in detail in the next chapter.

Chapter 2

Bell's Algorithm

Bell and Sejnowski [5] developed an algorithm based on *infomax*, information maximisation. As mentioned earlier, this algorithm is used as the basis for many other researches. The algorithm that will be discussed in this report, also uses Bell's algorithm as a starting point. Therefore Bell's algorithm will be discussed in this chapter.

2.1 Achieving Separation

First *information maximisation* is discussed. Let $x(t)$ be the input to a non-linear function with $v(t)$ as the output. The mutual information between input $x(t)$ and output $v(t)$ can be written as

$$I(v(t), x(t)) = H(v(t)) - H(v(t))' \tag{2.1}$$

with $I(v(t), x(t))$ is the mutual information, $H(v(t))$ is the entropy of the output and $H(v(t))'$ is whatever entropy the output has that didn't come from $x(t)$. The entropy of the output is

$$H(v(t)) = -E[\ln f_{v(t)}(v(t))] = -\int_{-\infty}^{\infty} f_{v(t)}(v(t)) \ln f_{v(t)}(v(t)) dv(t) \tag{2.2}$$

where $f_{v(t)}(v(t))$ is the probability density function (pdf) of $v(t)$. To find a maximum of (2.1), and thereby maximising information, it has to be differentiated with respect to a parameter w, involved in the mapping from $x(t)$ to $v(t)$. This leads to

$$\frac{\partial}{\partial w} I(v(t), x(t)) = \frac{\partial}{\partial w} H(v(t)) \tag{2.3}$$

because $H(v(t))'$ is independent of w, since w is involved in the mapping from $x(t)$ to $v(t)$.

This means that maximisation of the mutual information between input $x(t)$ and output $v(t)$ can be achieved by maximising the entropy of the output alone.

How can information maximisation be used in blind sources separation? In order to achieve separation, the mutual information between the outputs $y_j(t)$ of the unmixing system has to be minimised. When the outputs have no mutual information, they will be statistically independent. The joint entropy (information) of two outputs $y_1(t)$ and $y_2(t)$ can be written as

$$H(y_1(t), y_2(t)) = H(y_1(t)) + H(y_2(t)) - I(y_1(t), y_2(t)) \tag{2.4}$$

with $H(y_j(t))$ the entropy of output $y_j(t)$ and $I(y_1(t), y_2(t))$ the mutual information between the outputs. From (2.4) it can be seen that by maximising the joint entropy of the outputs, the mutual information of the outputs will be minimised. Now the joint entropy of the outputs is maximised by maximising the entropy of the outputs alone, which is done by information maximisation. Therefore information maximisation can be used for blind source separation. In the next section the maximisation of the entropy of the outputs will be discussed.

2.2 Maximising the Entropy of the Outputs

2.2.1 1-input 1-output case

For simplicity first the 1-input 1-output case is discussed. Of course there is no separation involved here. A derivation is given in this section on how to maximise the entropy of an output. In the next section the results will be extended to multiple inputs and outputs, where separation becomes a subject.

A single input $x(t)$ is passed through a non-linear function $f(x(t))$ to give an output $v(t)$. The non-linear function $f(x(t))$ is the mapping between input $x(t)$ and output $v(t)$, which contains the weight w mentioned earlier. In the case that $f(x(t))$ is monotonically increasing or decreasing, i.e. it has a unique inverse, the probability density function (pdf) of the output $f_{v(t)}(v(t))$ can be written as a function of the pdf of the input $f_{x(t)}(x(t))$

$$f_{v(t)}(v(t)) = \frac{f_{x(t)}(x(t))}{|\frac{\partial v(t)}{\partial x(t)}|} \tag{2.5}$$

where the bars denote absolute value. Substituting (2.5) in (2.2) gives

$$H(v(t)) = E[\ln|\frac{\partial v(t)}{\partial x(t)}|] - E[\ln f_{x(t)}(x(t))] \tag{2.6}$$

The term $E[\ln f_{x(t)}(x(t))]$ is independent of changes in the parameter w involved in the mapping of $x(t)$ to $v(t)$ (the function $f(x(t))$). The maximisation of the entropy of $v(t)$ can be done by changing the parameter w. So only the first term on the right side of (2.6) can be maximimised. To find the optimum for w, an 'online' stochastic gradient ascent learning rule is derived.

$$\Delta w \propto \frac{\partial \hat{H}}{\partial w} = \frac{\partial}{\partial w}(\ln|\frac{\partial v(t)}{\partial x(t)}|) = (\frac{\partial v(t)}{\partial x(t)})^{-1}\frac{\partial}{\partial w}(\frac{\partial v(t)}{\partial x(t)}) \tag{2.7}$$

with

$$\hat{H} = \ln|\frac{\partial v(t)}{\partial x(t)}| - \ln f_{x(t)}(x(t)) \tag{2.8}$$

the 'online', stochastic version of (2.6). Bell then derives a learning rule in case of the non-linear function

$$v(t) = \frac{1}{1+e^{-u}} \tag{2.9}$$

where $u = wx(t) + w_0$ with w_0 is a bias weight. This function is plotted in Figure 2.1. The function is an approximation of the cumulative density function (cdf) of the pdf of a normally distributed signal. In case of a normally distributed signal, this will lead to an output pdf $f_{v(t)}(v(t))$ according to (2.5) which will be close to a flat distribution, which is the maximum entropy distribution for a variable. In that situation the information flow through the non-linear function is optimal. In Figure 2.2 some functions are plotted to illustrate (2.5).

The bias-weight w_0 is meant to centre the steepest part of $f(x(t))$ to the peak in the pdf of input $x(t)$. But the first assumption for separation was that the sources are zero-mean and statistically independent. So normally there's no need for a bias. Therefore in the rest of this report the bias-weight will be omitted.

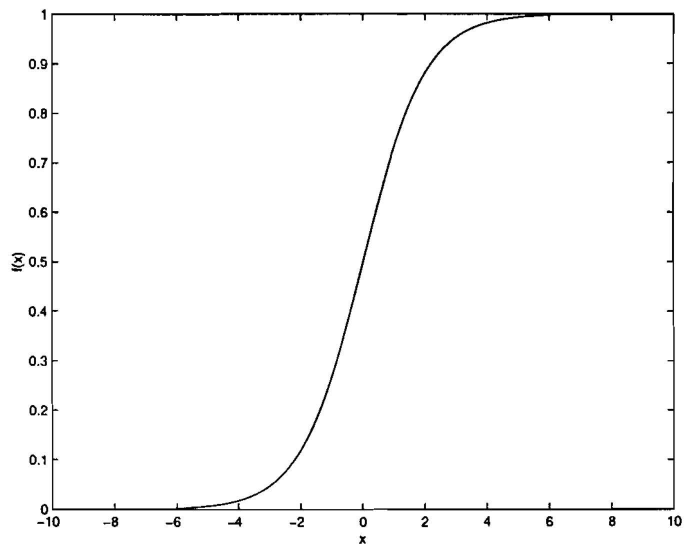

Figure 2.1: Non-linear function given in (2.9)

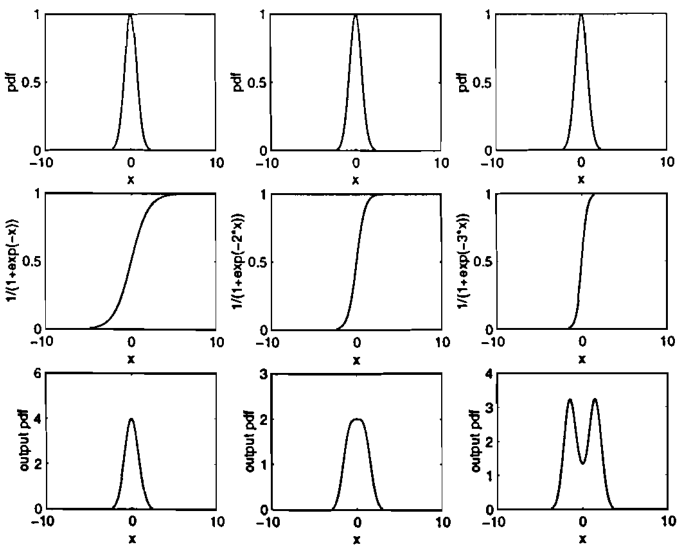

Figure 2.2: Plots to illustrate the influence of w on the pdf of the output.

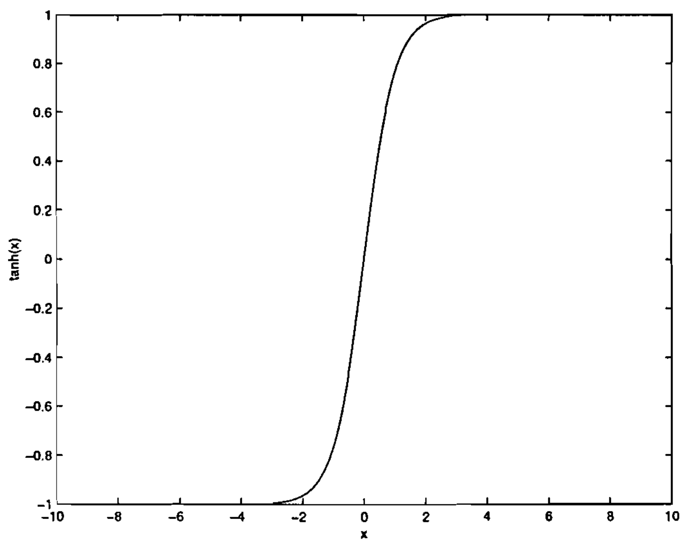

Figure 2.3: Hyperbolic tangent

The learning rule in case of the non-linear function (2.9) follows by substituting (2.9) in (2.7)

$$\Delta w \propto \frac{1}{w} + x(t)(1 - 2v(t)) \tag{2.10}$$

Bell also uses another non-linear function, the hyperbolic tangent.

$$y(t) = \tanh(wx(t)) \tag{2.11}$$

In Figure 2.3 the hyperbolic tangent is plotted.
Substituting (2.11) in (2.7) gives

$$\Delta w \propto \frac{1}{w} - 2x(t)v(t) \tag{2.12}$$

The Δw-rules (2.10) and (2.12) try to scale the slope of $f(x(t))$ so that it matches the variance of the pdf $f_{x(t)}(x(t))$ of input $x(t)$. For example, a narrow pdf will need a sharply sloping $f(x(t))$ to obtain a flat distribution. This can be seen in Figure 2.2.

So if the non-linear function $f(x(t))$ is properly adjusted to approximate the cdf of the input $x(t)$, the output pdf will be close to a flat distribution (see (2.5)). A flat distribution means optimal information flow from input $x(t)$ to output $v(t)$.

2.2.2 N-input N-output case

The 1-input 1-output case can easily be extended to the N-input N-output case. Consider a network with an input vector $\mathbf{X}$, a weight matrix $\mathbf{W}$ and a non-linear transformed output vector $\mathbf{V} = f(\mathbf{WX})$.

$$\mathbf{V} = \begin{pmatrix} v_1(t) \\ \vdots \\ v_n(t) \end{pmatrix} \tag{2.13}$$

The probability density function of the outputs $\mathbf{V}$, with $\mathbf{V}$ as in (2.13), can now be written as

$$f_{\mathbf{V}}(\mathbf{V}) = \frac{f_{\mathbf{X}}(\mathbf{X})}{|J|} \tag{2.14}$$

with the Jacobian J is the determinant of the matrix of partial derivatives

$$J = \det \begin{pmatrix} \frac{\partial v_1}{\partial x_1} & \cdots & \frac{\partial v_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial v_n}{\partial x_1} & \cdots & \frac{\partial v_n}{\partial x_n} \end{pmatrix} \tag{2.15}$$

The derivation is the same as in the previous section, except that now $\ln|J|$ is maximised instead of $\ln|\frac{\partial v(t)}{\partial x(t)}|$. In case of the non-linear function (2.9), the learning rule now becomes

$$\Delta \mathbf{W} \propto [\mathbf{W}^T]^{-1} + (\mathbf{1} - 2\mathbf{V})\mathbf{X}^T \tag{2.16}$$

with $\mathbf{X}$,$\mathbf{V}$ and $\mathbf{1}$ vectors ($\mathbf{1}$ is a vector of ones) and $\mathbf{W}$ a matrix.

In case of the hyperbolic tangent as the non-linear function (2.11), the learning rule becomes

$$\Delta \mathbf{W} \propto [\mathbf{W}^T]^{-1} - 2\mathbf{V}\mathbf{X}^T \tag{2.17}$$

The learning rule (2.17) derived by Bell [5] contains a computationally intensive matrix inversion. To avoid this matrix inversion, Amari [3] derived a learning rule using a natural gradient ascent algorithm (2.19) instead of the stochastic gradient ascent algorithm (2.18).

$$\Delta \mathbf{W} \propto \frac{\partial H(\mathbf{V})}{\partial \mathbf{W}} \tag{2.18}$$

$$\Delta \mathbf{W} \propto \frac{\partial H(\mathbf{V})}{\partial \mathbf{W}} \mathbf{W}^T \mathbf{W} \tag{2.19}$$

$\mathbf{W}^T\mathbf{W}$ in (2.19) is an optimal rescaling of the entropy gradient, as reported by Amari [35]. Using the previous results, (2.17) can be multiplied with $\mathbf{W}^T\mathbf{W}$. First, (2.17) can be rewritten using $\mathbf{x}^T = \mathbf{y}^T\mathbf{W}^{-T}$. This leads to

$$\Delta \mathbf{W} \propto \{\mathbf{I} - 2f(\mathbf{Y})\mathbf{Y}^T\}\mathbf{W}^{-T} \tag{2.20}$$

Multiplying (2.20) with $\mathbf{W}^T\mathbf{W}$ gives

$$\Delta \mathbf{W} \propto \{\mathbf{I} - 2f(\mathbf{Y})\mathbf{Y}^T\}\mathbf{W}^{-T}\mathbf{W}^T\mathbf{W} \tag{2.21}$$

This results in a new learning rule that does not contain the matrix inversion and is given in (2.22).

$$\Delta \mathbf{W} = \lambda\{\mathbf{I} - f(\mathbf{Y})\mathbf{Y}^T\}\mathbf{W} \tag{2.22}$$

with $f(\mathbf{Y})$ a non-linear function and $\mathbf{Y} = \mathbf{W}\mathbf{X}$.

Experiments are done with the algorithm given in (2.22). Two music sources, as discussed in Chapter 5, are digitally mixed using a random mixing matrix, e.g. $\mathbf{A} = \begin{pmatrix} 0.45 & 0.23 \\ 0.29 & 0.30 \end{pmatrix}$. The learning rate λ was set to 0.01. The non-linear function used was (2.9). The music sources were 10 seconds long. The algorithm achieved 'perfect' separation after one pass through the data. Each output represented a different music source. It was impossible to hear any interference from the other source. Amari also has proven that the performance of the algorithm is independent of the mixing matrix. So even for extremely bad scaled mixing matrices, such as $\mathbf{A} = \begin{pmatrix} 1 & 0.001 \\ 1 & 0.002 \end{pmatrix}$, the algorithm achieves separation.

For these instantaneous mixing matrices the algorithm performs very well. But in a real-world environment, the mixing matrices are not instantaneous any more. This subject will be discussed in the next section.

2.3 Real World Environment

The mixing matrix **A** in the previous sections was an instantaneous mixing matrix. The outputs at time t have nothing to do with signals at a time other than t. In a real environment, for example when people talk in a room, there are acoustical delays, echos and filtering by the room. This leads to a convolutive mixing matrix. The mixing coefficients then will become vectors of coefficients. The instantaneous mixing matrix $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ then becomes the convolutive mixing matrix $\mathbf{A} = \begin{pmatrix} \underline{a}_{11} & \underline{a}_{12} \\ \underline{a}_{21} & \underline{a}_{22} \end{pmatrix}$, with $\underline{a}_{ij}$ vectors. The definition for a matrix multiplication has to be redefined. The matrix multiplication

$$\begin{pmatrix} y_1(n) \\ y_2(n) \end{pmatrix} = \begin{pmatrix} \underline{w}_{11} & \underline{w}_{12} \\ \underline{w}_{21} & \underline{w}_{22} \end{pmatrix} \begin{pmatrix} x_1(n) \\ x_2(n) \end{pmatrix} \tag{2.23}$$

is equal to

$$y_1(n) = \underline{w}_{11} * x_1(n) + \underline{w}_{12} * x_2(n) \tag{2.24}$$

and

$$y_2(n) = \underline{w}_{21} * x_1(n) + \underline{w}_{22} * x_2(n) \tag{2.25}$$

where $*$ denotes convolution. For more information on this FIR matrix algebra see Lambert [8].

So now the algorithm not only has to deal with an unknown instantaneous mixing matrix, but also with delays, echos and filtering. The algorithm derived by Bell is not capable of separating convolutive mixtures. New methods have to be found to solve the convolutive problem. The idea was to go to the frequency domain in order to simplify the problem from the convolutive mixing case to the instantaneous mixing case. This will be discussed in the next chapter.

Chapter 3

Convolutive Mixtures

The mixing matrix in real-world environments is convolutive instead of instantaneous. Before starting to discuss on how to obtain the unmixing matrix, first it is important to show that an unmixing matrix does exist. After that the transformation to the frequency domain will be discussed.

3.1 Inverse Mixing Matrix

3.1.1 Theory

In this section the problem is discussed if an unmixing matrix exists given a convolutive mixing matrix. Suppose we have a mixing matrix

$$\mathbf{A} = \begin{pmatrix} \underline{a}_{11} & \underline{a}_{12} \\ \underline{a}_{21} & \underline{a}_{22} \end{pmatrix} \tag{3.1}$$

with $\underline{a}_{ij}$ a vector. The objective is then to find an unmixing matrix **W**, so that **WA** = **I**. The identity matrix I becomes with FIR matrix algebra

$$\mathbf{I} = \begin{pmatrix} \underline{i}_{11} & \underline{i}_{12} \\ \underline{i}_{21} & \underline{i}_{22} \end{pmatrix} \tag{3.2}$$

with

$$\underline{i}_{ij} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{if } i = j \ , \qquad \underline{i}_{ij} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{if } i \neq j \tag{3.3}$$

For the 2x2-case the transfer function from $s_1(n)$ (source 1) to $y_2(n)$ (output 2) is given by

$$\underline{h}_{s_1 y_2} = \underline{a}_{11} * \underline{w}_{21} + \underline{a}_{21} * \underline{w}_{22} \tag{3.4}$$

where $*$ denotes convolution. With an ideal unmixing matrix **W** the total transfer functions will become $\underline{h}_{s_i y_j} = \underline{i}_{ij}$. Therefore (3.4) can also be written in matrix form by

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{21} \end{pmatrix} \begin{pmatrix} \underline{w}_{21} \\ \underline{w}_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.5}$$

with

$$\mathbf{A}_{ij} = \begin{pmatrix} \underline{a}_{ij} & 0 & \dots & 0 \\ 0 & \underline{a}_{ij} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \underline{a}_{ij} \end{pmatrix} \tag{3.6}$$

where $\underline{a}_{ij}$ and $\underline{w}_{ij}$ are column vectors.

For the transfer function from $s_2(n)$ to $y_2(n)$ a similar notation can be made, which leads to

$$\begin{pmatrix} \mathbf{A}_{12} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \underline{w}_{21} \\ \underline{w}_{22} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.7}$$

Combining (3.5) with (3.7) gives

$$\begin{pmatrix} \mathbf{A}_{12} & \mathbf{A}_{22} \\ \mathbf{A}_{11} & \mathbf{A}_{21} \end{pmatrix} \begin{pmatrix} \underline{w}_{21} \\ \underline{w}_{22} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.8}$$

If (3.8) is true, that means that $y_2(n) = s_2(n)$. There will be no interference from $s_1(n)$ at all and no delays or echos from $s_2(n)$.

In general the condition in (3.8) is solvable if the width of the left matrix is greater than or equal to the height of the matrix. Let T_W be the length of the unmixing filters $\underline{w}_{ij}$ and T_A the length of the mixing filters $\underline{a}_{ij}$. The width of the left matrix of (3.8) is $2T_W$ and the heigth is $2(T_W + T_A - 1)$. This leads to the condition

$$2T_W \geq 2(T_W + T_A - 1) \tag{3.9}$$

which cannot be satisfied for $T_A > 1$. Therefore the ideal unmixing matrix for a given mixing matrix cannot be found using only two microphones. But when using more microphones, the height of the left matrix of (3.8) does not change, but the width does. When we define O the number of microphones (observations), the matrix dimensions become

$$\text{height matrix} = 2(T_W + T_A - 1) \tag{3.10}$$

$$\text{width matrix} = OT_W \tag{3.11}$$

Now (3.9) becomes

$$OT_W \geq 2(T_W + T_A - 1) \tag{3.12}$$

$$T_W \geq \frac{2(T_A - 1)}{O - 2} \tag{3.13}$$

From (3.13) it can be seen that the problem is solvable by using more than 2 microphones. But the idea was to separate n sources with the use of n observations (microphones). Perfect unmixing is not possible, but it is possible to achieve separation. When the transfer function for the interfering sources is a vector containing only zeros, there will be no signal from the interfering source in the output. This will lead to separated outputs. What the transfer functions for $s_1(n) \to y_1(n)$ and $s_2(n) \to y_2(n)$ look like, is less important for separation. The right side of (3.7) is left unknown. The important thing is that the right side of (3.5) remains a vector with only zeros. This means that the outputs will be filtered versions of the sources. (3.13) now becomes

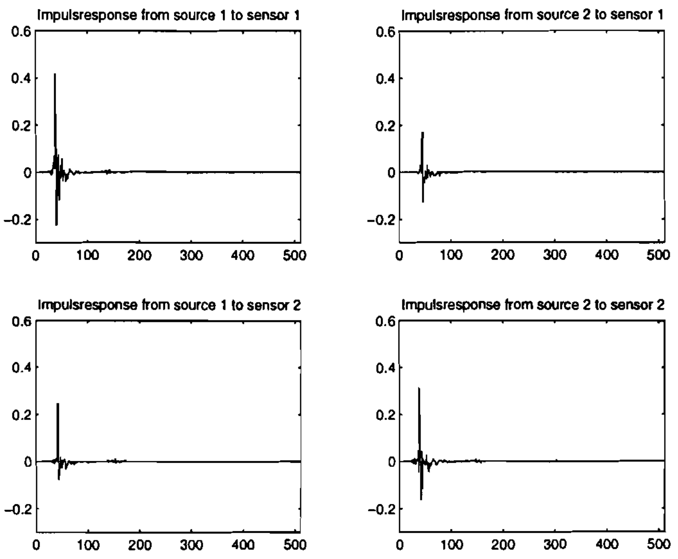

Figure 3.1: Mixing filters $\underline{a}_{ij}$. E.g. upper right plot is $\underline{a}_{12}$.

$$OT_W \geq T_W + T_A - 1 \tag{3.14}$$

$$T_W \geq \frac{T_A - 1}{O - 1} \tag{3.15}$$

Now the problem is solvable with two microphones. Then the length of the unmixing filters become

$$T_W \geq T_A - 1 \tag{3.16}$$

So by using two microphones and chosing the length of the unmixing filters one less than the length of the mixing filters, an unmixing filter can be found that achieves perfect separation in the sense that there are no interfering signals left in the outputs.

3.1.2 Verification

A simulation has been done in Matlab to verify the existence of an inverse mixing matrix. In this experiment the length of the unmixing filters is equal to the length of the mixing filters. The mixing filters are plotted in Figure 3.1.

These filters have length 512. Two sources and two microphones are used, i.e. the mixing matrix is a 2x2 FIR matrix. The unmixing filters $\underline{w}_{ij}$ are found under the condition that only (3.5) is valid, while (3.7) is not true. The found unmixing filters are plotted in Figure 3.2.

In Figure 3.3 the resulting transfer functions $\underline{h}_{ij} = \underline{a}_{1i} * \underline{w}_{j1} + \underline{a}_{2i} * \underline{w}_{j2}$, where * denotes convolution, are plotted.

It can be seen in Figure 3.3 that indeed $\underline{h}_{12}$ and $\underline{h}_{21}$ are zero. The amplitude of 10^{-16} is due to the limited accuracy of the computer. It also can be seen that $\underline{h}_{11}$ and $\underline{h}_{22}$ are not a 'delta'-function as it would be in the ideal situation. The outputs will be filtered versions of the sources, but with no interference from the other source. These results verify the existence of an unmixing matrix that achieves separation.

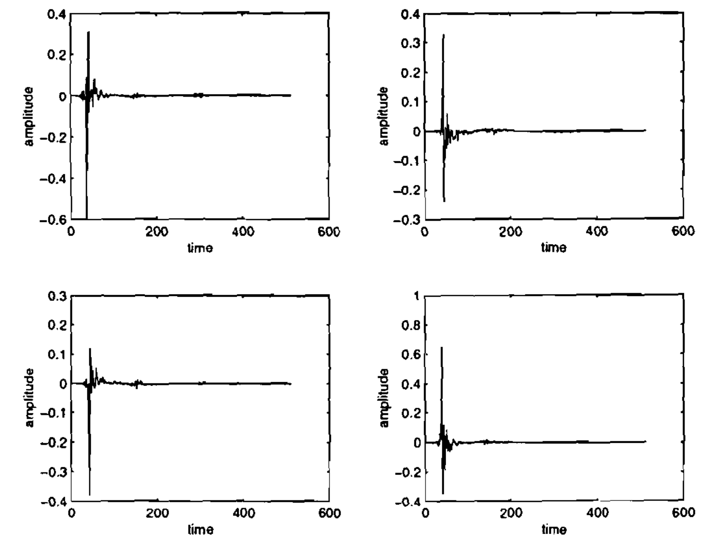

Figure 3.2: Unmixing filters $\underline{w}_{ij}$. E.g. upper right plot is $\underline{w}_{12}$.

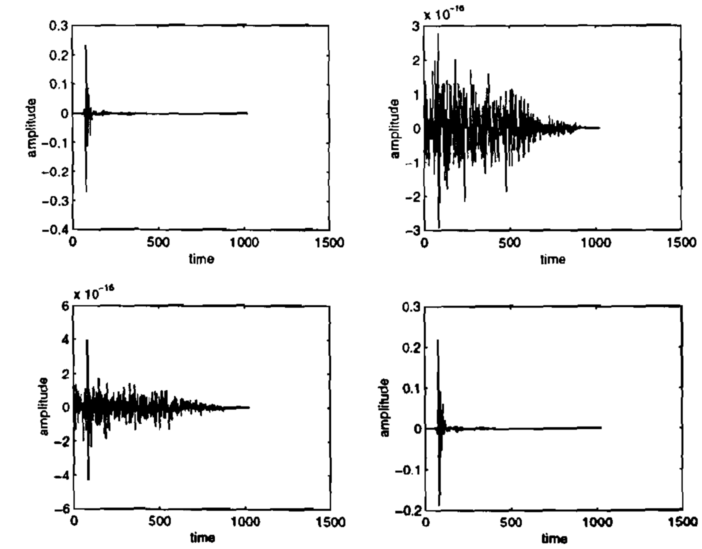

Figure 3.3: Total transfer functions $\underline{h}_{ij}$. E.g. upper right plot is transfer function for source 2 to output 1.

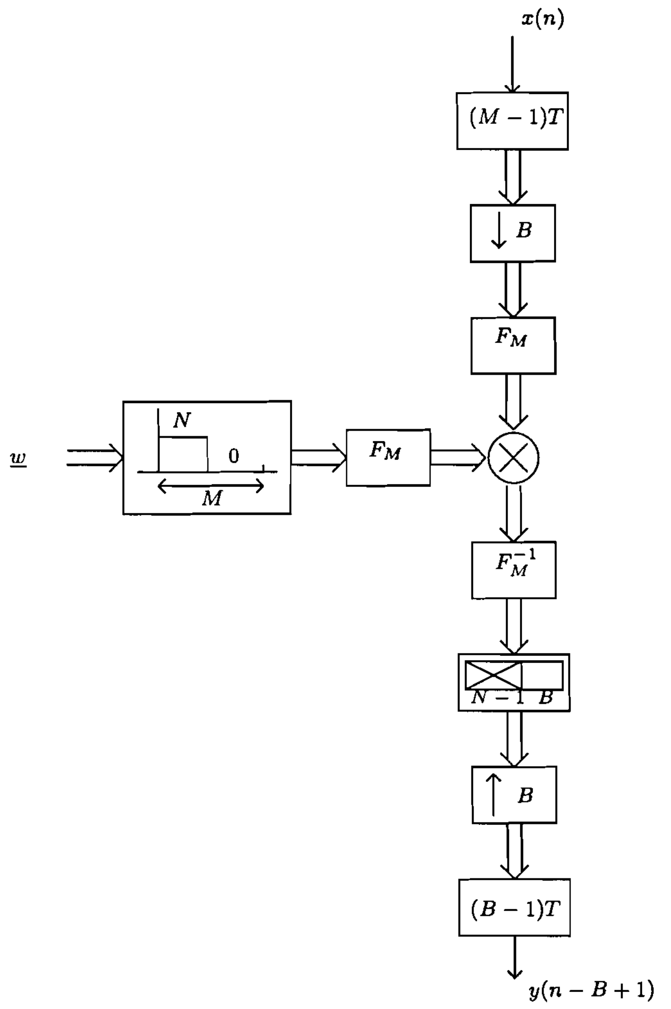

Figure 3.4: Frequency domain convolution of $\underline{w}$ and $x(n)$

3.3 Algorithm Modification for Complex Values

The instantaneous unmixing algorithm as given in (2.22) is not suitable for complex values. For clarity it is repeated:

$$\Delta \mathbf{W} = \lambda\{\mathbf{I} - f(\mathbf{y})\mathbf{y}^T\}\mathbf{W} \tag{3.18}$$

with $f(\mathbf{y})$ is a non-linear function, $\mathbf{y} = \mathbf{W}\mathbf{x}$ and $\mathbf{W}$ is the unmixing matrix.

In order to deal with complex values, two modifications have to be made. First the matrix transpositions have to be changed to hermitian transpositions (conjugate transpose). In notation this means that $.^T$ is replaced by $.^H$. The algorithm (3.18) then becomes

$$\Delta \mathbf{W} = \lambda\{\mathbf{I} - f(\mathbf{y})\mathbf{y}^H\}\mathbf{W} \tag{3.19}$$

The second modification is a non-linear function that is suitable for complex values. The hyperbolic tangent is not suitable for complex values. Smaragdis [36] proposes the non-linear function

$$f(z) = \tanh(\mathrm{Re}\{z\}) + i\tanh(\mathrm{Im}\{z\}) \tag{3.20}$$

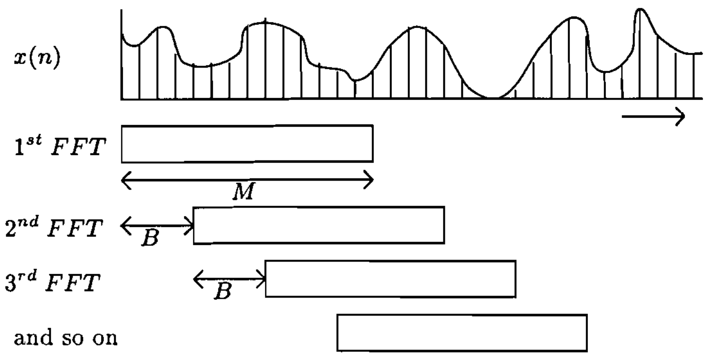

Figure 3.5: Overlap in FFT

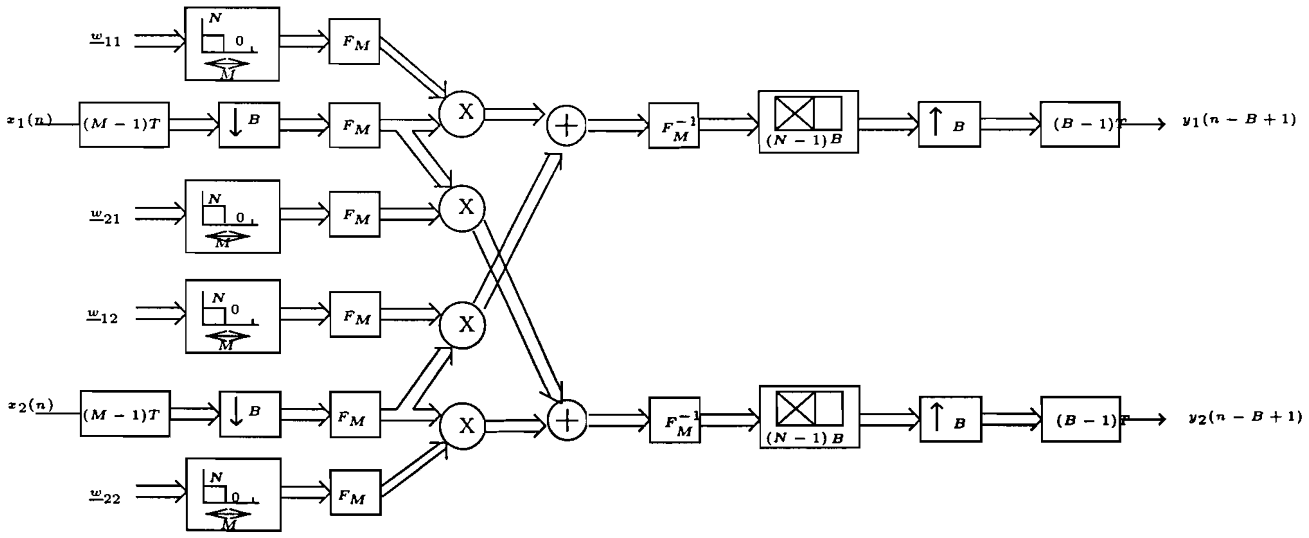

Figure 3.6: Frequency domain transformation for the 2x2 case

where $\text{Re}(z)$ denotes the real part and $\text{Im}(z)$ the imaginary part of z with z a complex value. This non-linear function is suitable for use in the frequency domain.

The instantaneous unmixing algorithm given in (3.19) with a non-linear function that is suitable for complex values, e.g. as in (3.20), can now be used for separation within each frequency bin. The frequency domain blind source separation algorithm is discussed in the next chapter.

Chapter 4

Frequency Domain Blind Source Separation Algorithm

In this chapter a frequency domain blind source separation algorithm will be discussed. It uses Bell's algorithm (2.22) for instantaneous mixtures, which is now modified for use in the frequency domain (3.19). The transformation to the frequency domain is done in order to reduce the problem of convolutive mixtures to the problem of instantaneous mixtures.

4.1 Block Diagram

In Figure 4.1 a block diagram of the frequency domain algorithm is given. Depicted is the 2x2 case. The frequency domain algoritm presented in this chapter is not restricted to the 2x2 case. It can be applied to NxN multichannel problems. However in this report only the 2x2 case is discussed and all simulations are done using two sources and two sensors. The tools used for these simulations are especially designed for the 2x2 case. But as stated before the frequency domain algorithm can be extended for multiple sources and multiple sensors.

The observations $x_1(n)$ and $x_2(n)$ are first transformed to the frequency domain by a Fast Fourier Transform. In the frequency domain the signals $X_1[k, mB]$ and $X_2[k, mB]$ consist of multiple frequency bins k. After every B samples a new block is processed (see Section 3.2.1.). The time index mB indicates this block processing (for every new block, $m = m + 1$). As discussed before, within each frequency bin an independent separation is performed. The frequency bins of the mixtures are instantaneous mixtures of the sources' frequency bins. Therefore within each

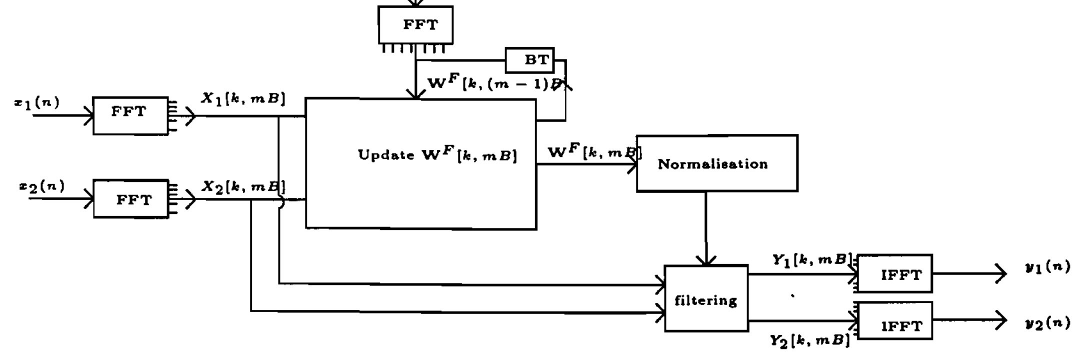

Figure 4.1: Block diagram of the frequency domain algorithm

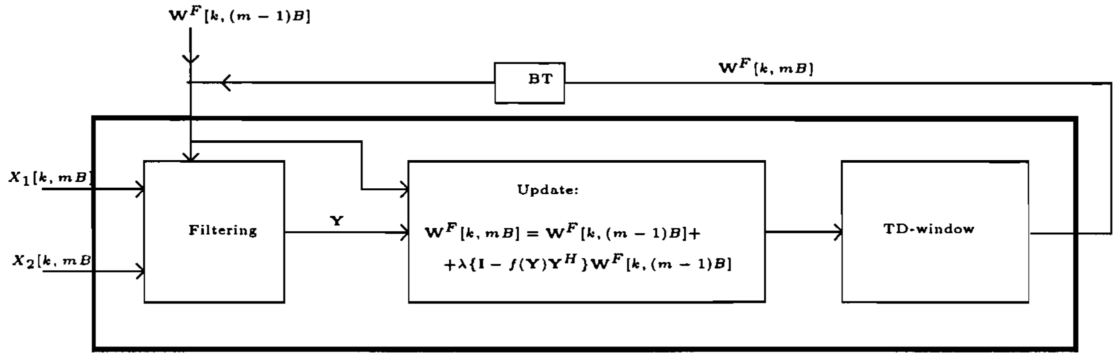

Figure 4.2: Update block

frequency bin, the unmixing matrix $\mathbf{W}^F[k,mB]$ is a 2x2 scalar matrix. The time domain unmixing FIR matrix $\mathbf{W}$ is initialised and then transformed to the frequency domain. The Fourier transform of $\mathbf{W}$ is denoted as $\mathbf{W}^F[k,mB]$. The FFT of $\mathbf{W}$ in Figure 4.1 is done only once for the initialisation. In Figure 4.1 only one frequency bin is depicted. The other frequency bins have the same block diagram. This is schematicly drawn by multiple lines after the FFT, from which only one line is further specified. On the other side, the IFFT has multiple lines as input from which also only one is specified.

Next, $\mathbf{W}^F[k,mB]$ is updated with the use of the variables $X_1[k,mB]$, $X_2[k,mB]$ and $\mathbf{W}^F[k,(m-1)B]$. The updated $\mathbf{W}^F[k,mB]$ is used for the next update. But before filtering the inputs $X_1[k,mB]$ and $X_2[k,mB]$ with the unmixing matrix $\mathbf{W}^F[k,mB]$ to achieve separated outputs $Y_1[k,mB]$ and $Y_2[k,mB]$ (the block called *filtering* in Figure 4.1), the unmixing matrix $\mathbf{W}^F[k,mB]$ has to be normalised. This normalisation is needed to ensure that in all frequency bins the unmixing matrices have the same order of magnitude. Because the separations per frequency bin are independent, the unmixing matrices per frequency bin will not have the same scaling. Transforming these different scaled bins back to the time domain, will give an unmixing filter that will suppress certain frequencies more than others, leading to for example a high-pass filter. More details on this normalisation will be discussed in Section 4.2.

In Figure 4.2 the update block from Figure 4.1 is further specified.

First the outputs $Y_1[k,mB]$ and $Y_2[k,mB]$ are calculated. In Figure 4.2 this is written as $\mathbf{Y}$ for convenience with $\mathbf{Y}$ as in (4.1).

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1[k,mB] \\ \mathbf{Y}_2[k,mB] \end{pmatrix} \tag{4.1}$$

These outputs are used in the update of $\mathbf{W}^F[k,mB]$, which is the frequency domain modified algorithm given in (3.19). The update of $\mathbf{W}^F[k,mB]$ becomes

$$\mathbf{W}^F[k,mB] = \mathbf{W}^F[k,(m-1)B] + \lambda\{\mathbf{I} - f(\mathbf{Y})\mathbf{Y}^H\}\mathbf{W}^F[k,(m-1)B] \tag{4.2}$$

where λ is the learning rate, $\mathbf{I}$ the identity matrix, $f(\mathbf{Y})$ a non-linear function of $\mathbf{Y}$ and $\mathbf{Y}^H$ the Hermitian transpose of $\mathbf{Y}$.

The last block in Figure 4.2 is called *TD-window*. This block does not work independently on all bins, but after all $\mathbf{W}^F[k,mB]$ are updated, the results over all frequency bins are used in *TD-window*. $\mathbf{W}^F[k,mB]$ can be written as

$$\mathbf{W}^F[k,mB] = \begin{pmatrix} \underline{w}^F_{11}[k,mB] & \underline{w}^F_{12}[k,mB] \\ \underline{w}^F_{21}[k,mB] & \underline{w}^F_{22}[k,mB] \end{pmatrix} \tag{4.3}$$

with $\underline{w}_{ij}^F[k,mB]$ the frequency transform of the time domain unmixing filter $\underline{w}_{ij}$. When transforming $\underline{w}_{ij}$ to the frequency domain, the length of the vector is changed to match the FFT-size. Zeros are added to the vector $\underline{w}_{ij}$ until the vector has the length of the FFT-size. Adding zeros in the time domain will not affect the filter $\underline{w}_{ij}$. But after the transformation to the frequency domain, every bin of $\underline{w}_{ij}^F[k,mB]$ will be updated independently. When transforming $\underline{w}_{ij}^F[k,mB]$ back to the time domain, the previously added zeros will not be zero anymore. Thus the unmixing filter $\underline{w}_{ij}$ has been changed. In order to make sure that the time domain unmixing filter $\underline{w}_{ij}$ remain of the same length, the block *TD-window* is introduced. *TD-window* transforms a frequency domain signal $\underline{w}_{ij}^F[k,mB]$ to the time domain, substitutes the last number of values that were originally added (as zeros) again with zeros and then transforms the new $\underline{w}_{ij}$ back to the frequency domain. The results of *TD-window* is then delayed by B samples before it is used in the next update. B is the number of samples before the next block is processed.

4.2 Normalisation

To explain the reason behind normalisation, the inputs $X_1[k,mB]$ and $X_2[k,mB]$ per frequency bin have to be examined. What do these signals represent? They represent the frequency content of the signals $x_1(n)$ and $x_2(n)$ in frequency bin k at block mB. That means that when for example $x_1(n)$ contains a strong component at 300 Hz, $X_1[k,mB]$ with frequency bin k corresponding with the frequency 300 Hz will have a high value. On the other hand when $x_1(n)$ contains almost no signal at a certain frequency, the value of $X_1[k,mB]$ at that frequency will be very low. The algorithm tries to find a stable situation for the unmixing matrices $\mathbf{W}^F[k,mB]$. When an unmixing matrix $\mathbf{W}^F[k,mB]$ is found that achieves separation, it will slightly vary around this solution (because for every new block the signals are not exactly the same). Recalling (4.2), it can be seen that in order to let $\mathbf{W}^F[k,mB] = \mathbf{W}^F[k,(m-1)B]$, the algorithm will try to make

$$f(\mathrm{Y})\mathrm{Y}^H = \mathrm{I} \tag{4.4}$$

where $\mathbf{Y} = \mathbf{W}^F[k,mB]\mathbf{X}$ with $\mathbf{X}$ as in (4.5).

$$\mathrm{X} = \begin{pmatrix} \mathrm{X}_1[k,mB] \\ \mathrm{X}_2[k,mB] \end{pmatrix} \tag{4.5}$$

When (4.4) is true, this means that when $\mathbf{X}$ becomes bigger, $\mathbf{W}^F[k,mB]$ will have to become smaller. When $x_1(n)$ for example is a mixture of multiple speakers, the frequency bins corresponding with frequencies between 0-5 kHz (typically for speech) will have relatively high values in relation to the bins corresponding with higher frequencies. This leads to a set of $\mathbf{W}^F[k,mB]$ that are not in the same order of magnitude for all bins k, because $\mathbf{X}$ is not in the same order of magnitude for all bins k. By the transformation back to the time domain, this will lead to an unmixing filter that behaves as a high pass filter.

In order to scale all $\mathbf{W}^F[k,mB]$ to the same order of magnitude, a normalisation is introduced.

$$\mathrm{W}^F[k,mB]' = \mathrm{W}^F[k,mB]\mathrm{det}(\mathrm{W}^F[k,mB])^{-\frac{1}{2}} \tag{4.6}$$

with $\mathbf{W}^F[k,mB]'$ the normalised unmixing matrix. The normalisation given in (4.6) makes the determinant of every scalar matrix $\mathbf{W}^F[k,mB]$ one. By doing this all values of $\mathbf{W}^F[k,mB]$ for every bin k are in the same order of magnitude. As a result the time domain unmixing filters $\underline{w}_{ij}$ will not behave as high pass filters, but as all-pass filters.

As can be seen in Figure 4.1 the normalisation is placed outside the update cycle. The normalisation is used for normalising the output only. When the unmixing matrices $\mathbf{W}^F[k,mB]$ are normalised after every update and from the beginning of the separation, the separating capabilities of the algorithm are damaged. Therefore it is important to use the normalisation only for normalising the output. Simulations have been done with the normalisation placed inside the update cycle to show the effect on the separating capabilities of the algorithm. The results indeed showed that the performance of the algorithm degraded when the normalisation was placed inside the update cycle. For the same situation, the algorithm separated the sources better when the normalisation was used only to normalise the outputs.

More simulations are done with and without the normalisation. These simulations show that the time domain unmixing filters become high-pass filters when the normalisation has been omitted, while they behave more as an all-pass filter with the normalisation included. So the frequency spectra of the sources are less affected when using the normalisation than without the normalisation. More details on these simulations will be discussed in Chapter 5. So the updating of the unmixing matrices is done without normalisation, but in the final linear filtering the unmixing matrices are normalised to avoid high-pass filtering.

4.3 Non-Linear Function

The non-linear function in (4.2) is very important for good performance of the algorithm. By chosing a wrong non-linear function the algorithm might not separate at all. The question then arises what the non-linear function would have to look like. As discussed in Chapter 2 Bell [5] claims that the ideal non-linearity would be the cumulative density function (cdf) of the probability density functions (pdf) of the independent sources. Bell mainly uses the hyperbolic tangent

$$f(z) = \tanh(z) \tag{4.7}$$

or the non-linear function

$$f(z) = \frac{1}{1+e^{-z}} \tag{4.8}$$

as the non-linearity. Both functions approximate the cdf of normally distributed functions. Both functions show good separating capabilities in simulations done by Bell [5].

The hyperbolic tangent seems a simple but still good performing non-linearity. However the problem is that by transforming to the frequency domain, the real values become complex values. The hyperbolic tangent is not suitable for complex values. Therefore the non-linearity needs to be modified. Smaragdis [36] proposes a non-linearity for complex values

$$f(z) = \tanh(\mathrm{Re}(z)) + i\tanh(\mathrm{Im}(z)) \tag{4.9}$$

with z a complex value. Re(z) denotes the real part of z and Im(z) the imaginary part. In Figure 4.3 this non-linear function is depicted.

Smaragdis [36] did some simulations using this non-linearity. Mixed sources were separated using his proposed non-linear function. His adaptation of Bell's non-linear function is quite simple and straightforward, but it seems to work. In fact Smaragdis takes the hyperbolic tangent of the imaginary part and the real part separate.

Another way of introducing an imaginary part in the non-linear function is by using a different notation of the signal. A complex value z can also be written as

$$z = |z|e^{i\,\mathrm{angle}(z)} \tag{4.10}$$

If the hyperbolic tangent is applied only to the absolute value of z, the phase is preserved. The new non-linear function, suitable for complex values, now becomes

$$f(z) = \tanh(|z|)e^{i\,\mathrm{angle}(z)} \tag{4.11}$$

This non-linear function preserves the phase in contrast to the non-linear function in (4.9). It makes sense to keep the phase, because the phase information may help the algorithm to separate

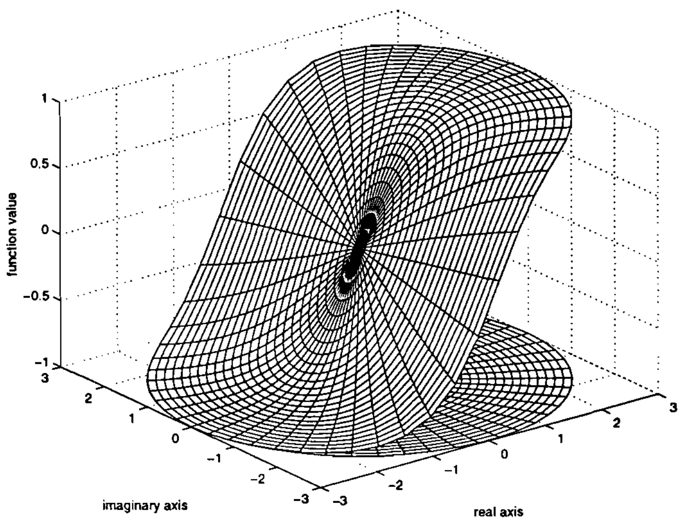

Figure 4.3: The hyperbolic tangent non-linear function modified for complex values

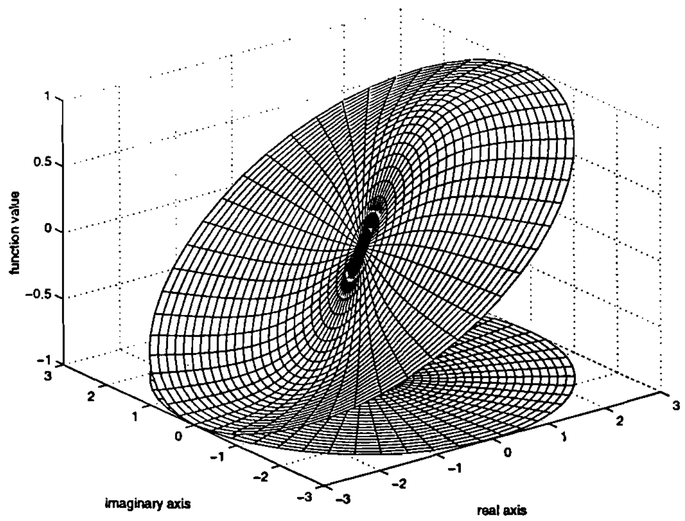

Figure 4.4: Non-linear function that preserves the phase

better. When there is more information available from the sources, it is more likely to achieve separation. In Figure 4.4 the plot of (4.11) is depicted.

Simulations have been done with both non-linear functions. There was no clear difference between any of the simulations. The resulting plots looked the same except some details, but those differences occured in regions where a deviation of a few dB barely affects the result, e.g. for high frequencies a attenuation of -40 dB instead of -35 dB. For high frequencies the sources will contain little information (in case of speech). However, there is no audible difference between the results. For more details on these simulations see Chapter 5.

Both functions seem to have the same separating capabilities. Therefore it is more convenient to choose (4.9) instead of (4.11), because the computation of (4.11) is more intensive than the computation of (4.9). And since one of the original objectives was to make the algorithm work in real-time, the straightforward non-linear function in (4.9) is the best choice of the two given functions.

4.4 Permutation Problem

By moving to the frequency domain, separation has to be performed for every frequency bin. For each bin, the ordering of the outputs does not have to be the same as in any other bin. For example in the 2x2 case, one bin may have source 1 as output 1, while another bin has source 1 as output 2. When putting all bins of a certain output together and transforming it back to the time domain, this will give a distorted signal. Obviously every bin must have the same ordering of the outputs. But the separation within each bin is performed independent of other bins. The problem here is on how to force every single bin to the same ordering of the outputs.

One way to try to achieve this, is by letting a bin being influenced by its neighbour bins. The idea behind this is that when the bins are small enough, i.e. when the FFT-size is big enough, the signals will be approximately the same (the frequency content of a signal will not differ much for two close frequencies). So by letting one bin have a little influence on the other bin, it may help the other bin to achieve the same ordering as the first bin. Then the second bin has a little influence on the third bin, so the third bin will have the same ordering, and so on.

Now the question arises how to let the neighbouring bins influence eachother. In Figure 4.2 there is a block called *TD-window*. As stated before this block transforms a signal to the time domain, then substitutes the last values with zeros and transforms it back to the frequency domain. The substitution is done by multiplying the signal with a window as can be seen in Figure 4.5.

A multiplication with *TD-window* in the time domain corresponds to a convolution in the frequency domain with the frequency transform of *TD-window*. The frequency transform of *TD-window* depends on the number of zeros and the FFT-size. Some plots are given to illustrate the dependency of the frequency transform of *TD-window* on the length of *TD-window*. In all situations a FFT-size is used of 256. In Figure 4.6 the frequency transform of *TD-window* with 25 ones and 256-25=231 zeros is given, in Figure 4.7 with 100 ones and 156 zeros and in Figure 4.8 with 128 zeros and 128 ones. In Figure 4.6 the frequency transform has the shape of a $\frac{\sin(x)}{x}$-function. Convolving the frequency bins of $\mathbf{W}^F[k, mB]$ with this frequency transform, will introduce a 'leakage' from other bins. A bin is influenced by its neighbour bins. This is exactly what was described at the beginning of this section as a possible solution to the permutation problem. But in Figure 4.7 this influence is already reduced and the shape looks less as a $\frac{\sin(x)}{x}$-function. In Figure 4.8 the frequency transform is almost a delta function. Then the neighbour bins will have almost no influence. But in Figure 4.8 only the real part of the frequency transform is plotted. The imaginary part is given in Figure 4.9. It can be seen that there still is some influence from the other bins. The operation of *TD-window* introduces an influence of neighbouring bins that

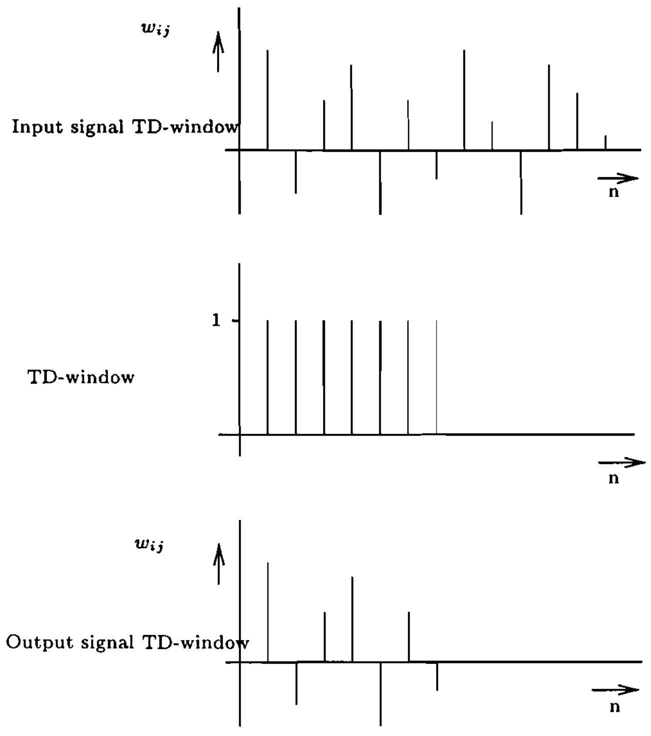

Figure 4.5: Operation of TD-window

might help the permutation problem.

In order to verify the use of *TD-window* in solving the permutation problem, simulations are done. A previous simulation where separation is achieved without any permutation problems, is done again but this time without the block *TD-window*. With this new simulation, there were permutation problems. For some frequencies, the ordering of the outputs suddenly switched. For more details on these simulations see Chapter 5. With the inclusion of *TD-window* in the update of $\mathbf{W}^F[k, mB]$, no permutation problems occured in simulations done for this report.

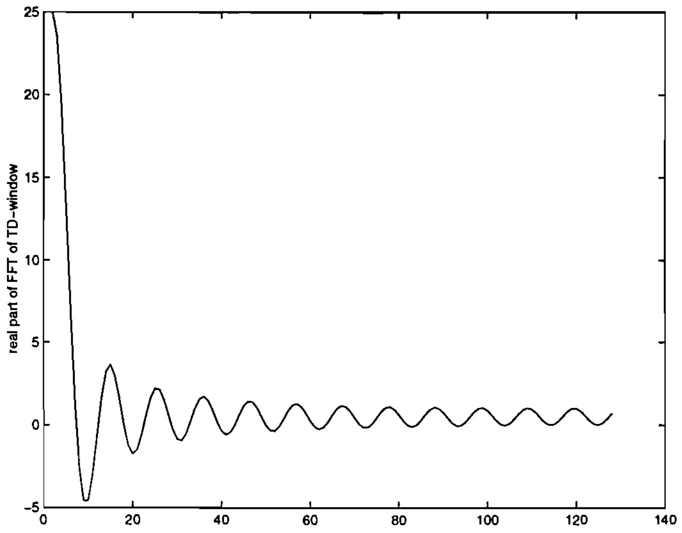

Figure 4.6: Frequency transform of *TD-window* with 25 ones and 231 zeros

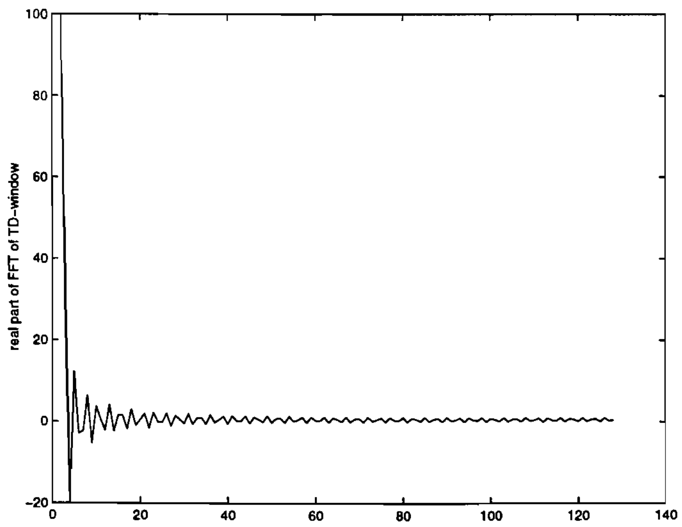

Figure 4.7: Frequency transform of *TD-window* with 100 ones and 156 zeros

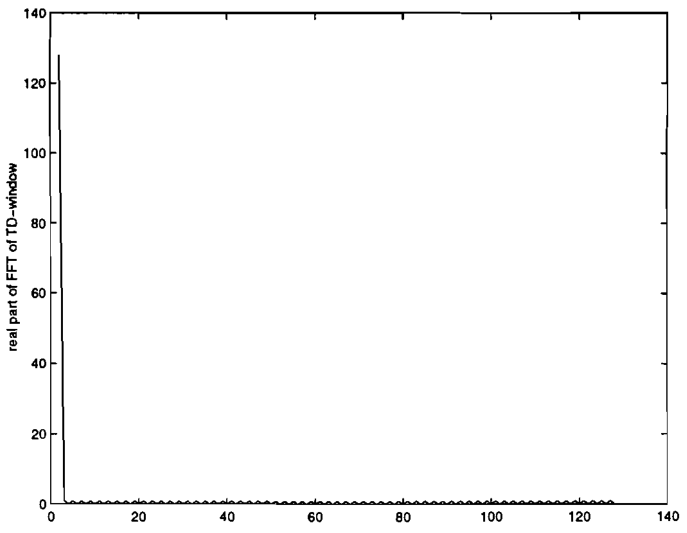

Figure 4.8: Frequency transform of *TD-window* with 128 ones and 128 zeros

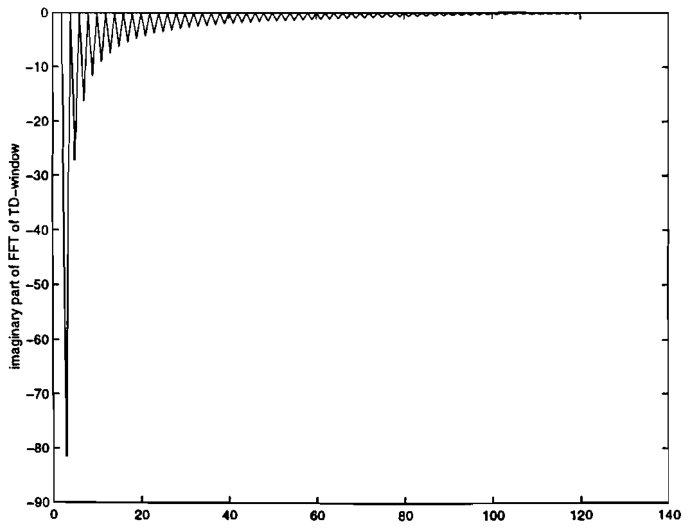

Figure 4.9: Imaginary part of frequency transform of *TD-window* with 128 ones and 129 zeros

Chapter 5

Simulations

In this chapter simulations done in Matlab with the frequency domain blind source separation algorithm from Chapter 4 will be discussed. First two music sources are mixed and then it was tried to separate them. This is done because the results are easier to interpret than when two speakers (e.g. two persons talking at the same time in the same room) are mixed. When trying to separate two speakers, sometimes situations occur where one of the speakers or even both of them are silent. This will make it more difficult to interpret how much one signal is suppressed. When using continuous signals, there are always two signals so that it is possible to compare one signal with the other. The use of speakers may also affect the algorithm, because the algorithm assumes the presence of two sources. When one source disappears, the algorithm may deviate from the wanted solutions. When the person then starts to speak again, the algorithm has to learn the solutions again. Therefore the first simulations will be done with two music sources. Later the case of two speakers will be discussed.

5.1 Known Mixing Matrix

For an objective judgement on the performance of the algorithm, it is useful to have knowledge about the mixing matrix. When the mixing matrix is known, it is possible to compute the transfer function from every source to every output. These functions can be plotted to give an objective measurement of the performance of the algorithm. In Figure 5.1 an example is given of such a plot.

On the horizontal axis the frequency in Hz is plotted on a logarithmic scale. On the vertical axis the attenuation is given in dB. The left plot (a) contains the transfer functions from source 1 (solid line) and source 2 (dashed line) to output 1. The right plot (b) gives the transfer functions from source 1 (solid line) and source 2 (dashed line) to output 2. These transfer functions are computed with the use of the known mixing filters ($\underline{a}_{ij}$) and the unmixing filters ($\underline{w}_{ij}$), according to

$$\underline{h}_{ij} = \underline{a}_{1j} * \underline{w}_{i1} + \underline{a}_{2j} * \underline{w}_{i2} \tag{5.1}$$

where * denotes convolution. In the rest of this chapter likewise plots will have the same interpretation.

5.2 Separation of Two Music Sources

In this section the simulations will be discussed concerning two music signals. Both music sources are sampled at 16 kHz and last approximately 10 seconds. Music source 1 is a sample from the song 'spOOL' performed by Queensryche, music source 2 is a sample from 'Peruvian Skies' performed by Dream Theater. In Figure 5.2 these music sources are plotted.

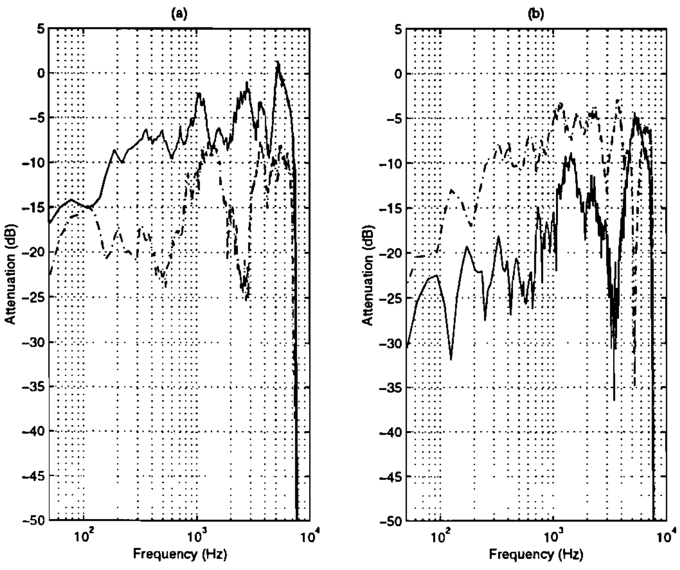

Figure 5.1: Objective measurement of the performance of the algorithm

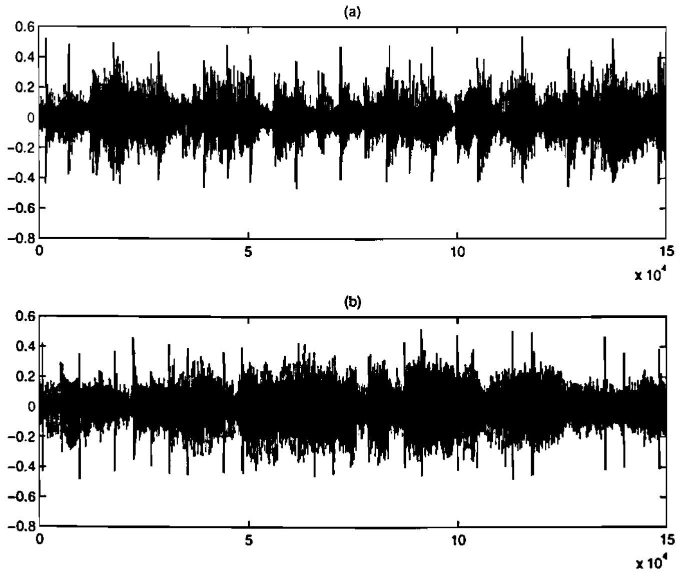

Figure 5.2: Wave-file plots of music sources (a) Source 1 (b) Source 2

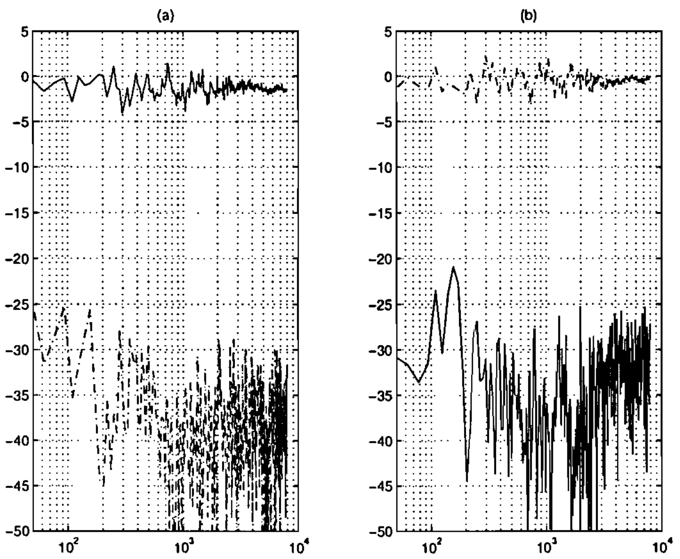

Figure 5.3: Result for instantaneous mixing matrix with adaptive unmixing filter length 512

5.2.1 Instantaneous Mixing Matrix

In order to verify the working of the algorithm, an instantaneous mixing matrix is used for the first simulation. The mixing matrix is given in (5.2).

$$A = \begin{pmatrix} 1 & 0.43 \\ 0.38 & 1 \end{pmatrix} \tag{5.2}$$

Further parameters for this simulation are a FFT-size of 1024, an adaptive filter length of 1 and a learning rate of 0.001. The used non-linear function is the function in (4.9). The sources are succesfully separated. Both outputs represent one source, while the other source is suppressed by 40 dB. The suppressed source cannot be heard anymore, so the separation sounds perfect. There is no use in plotting the transfer function for this situation, because the mixing and the adaptive filter length are only 1. This means that the transfer functions are simply constants.

Another way of judging the performance of the algorithm is by listening to the results. Therefore a CD is made as a useful attachment to this report. References will be given throughout the report to the corresponding tracks on the CD. These references will be printed *italic.* However the CD is not required to read this report. Now the first references to the CD will be given. Throughout the rest of this chapter similar references will be given.

Track 1: Music source 1 'spOOL'
Track 2: Music source 2 'Peruvian Skies'
Track 3: $x_1(n)$ *and* $x_2(n)$ *for instantaneous mixing matrix*
Track 4: $y_1(n)$ *and* $y_2(n)$ *for instantaneous mixing matrix with adaptive filter length 1*

The same simulation was done again, but this time with an adaptive filter length of 512. The normalisation was also applied. The results can be seen in Figure 5.3.

Figure 5.3 shows that one line lies around 0 dB, and the other one (representing the transfer function for the suppressed source) lies at approximately -25 dB to -35 dB. With an unmixing filter length of only 1, a suppression of 40 dB was achieved. Making the unmixing filters unnecessary long, degrades the performance of the algorithm in case of an instantaneous mixing matrix. When only one weight is needed, the other weights introduce unnecessary delays, because they are not exactly zero, but vary a little around zero due to the updating. However when listening to the

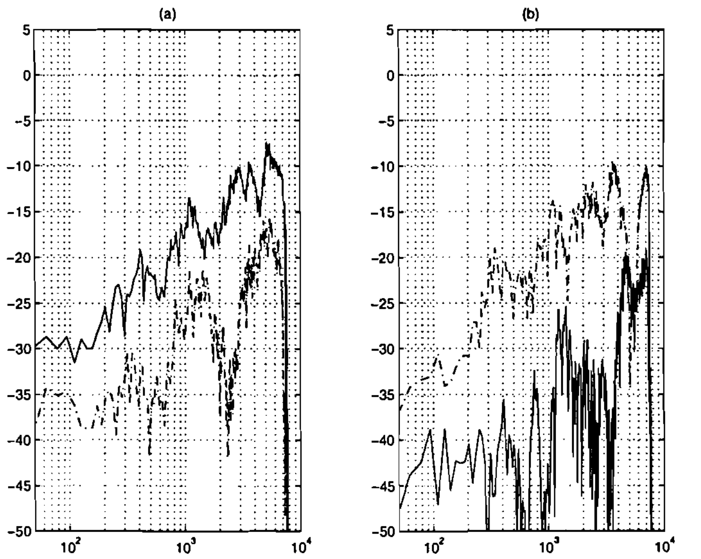

Figure 5.4: Performance for convolutive mixing matrix without normalisation

results the suppressed sources still cannot be heard. Again separation is nearly perfect.

Track 5: $y_1(n)$ and $y_2(n)$ for instantaneous mixing matrix with adaptive filter length 512

Now that the algorithm seems to have separating capabilities, it is time to try a more challenging mixing matrix. Instead of instantaneous mixtures now convolutive mixtures will be given to the algorithm.

5.2.2 Convolutive Mixing Matrix

The convolutive mixing matrix used in most simulations is plotted in Figure 3.1. The transfer functions $s_1 \to y_1$, $s_2 \to y_1$, $s_1 \to y_2$ and $s_2 \to y_2$ are plotted with e.g. $s_2 \to y_1$ the upper right plot. The music sources are now mixed with this convolutive mixing matrix.

Track 6: $x_1(n)$ and $x_2(n)$ with convolutive mixing matrix

Further simulation parameters are an FFT-size of 1024, an adaptive filter length of 512, which is the same as the mixing filter length. Sample rate is 16 kHz. Learning rate is 0.001. The used non-linear function is given in (4.9).

The unmixing matrix was initialised with

$$\underline{w}_{11} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \underline{w}_{12} = \begin{pmatrix} -0.5 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \underline{w}_{21} = \begin{pmatrix} -0.5 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad \underline{w}_{22} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{5.3}$$

Without normalisation the results of this simulation are plotted in Figure 5.4.

Figure 5.4 shows that the unmixing matrices $\mathbf{W}^F[k, mB]$ are not in the same order of magnitude over all frequency bins. Because of the high information in the low frequencies, the unmixing matrices for these low frequencies have small values. The attenuation for these frequencies is in the order of -30 dB, while for high frequencies it is -10 dB. This will give an unmixing filter that behaves as an high-pass filter. In the outputs, the low 'bass' part of the music is suppressed.

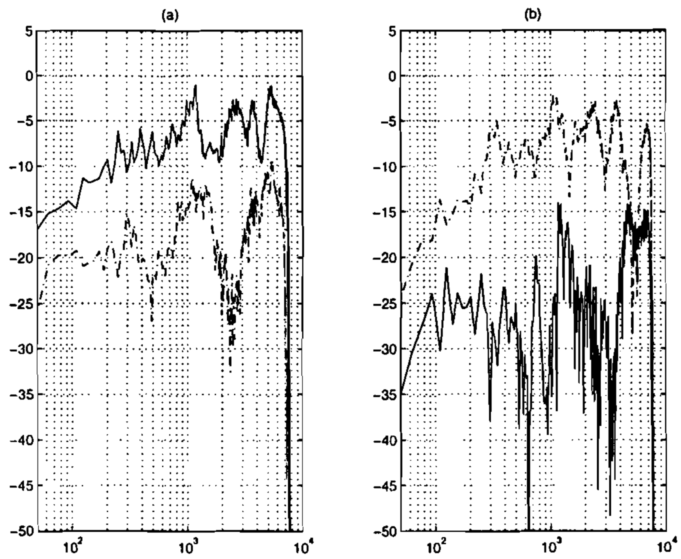

Figure 5.5: Performance for convolutive mixing matrix with normalisation

Track 7: $y_1(n)$ and $y_2(n)$ for convolutive mixing matrix without normalisation

As discussed in Section 4.2, normalisation is applied to the unmixing matrices before the final filtering (see Figure 4.1). Applying this normalisation to the simulation from Figure 5.4 leads to the results plotted in Figure 5.5.

As can be seen in Figure 5.5, the low frequencies are boosted. Still there is some difference between the attenuation of low and high frequencies, but this difference is not as big as before. These differences cannot be removed by the normalisation, because the normalisation only scales the unmixing matrices. The ratio between the elements of the unmixing matrix is not affected. This ratio will differ per frequency bin and therefore the transfer function will not be flat.

The normalisation has no further side-effects. No distortions or degrading of the music due to the normalisation can be heard.

Track 8: $y_1(n)$ and $y_2(n)$ for convolutive mixing matrix with normalisation

5.2.3 Convergence Considerations

Separation in the previous simulations was achieved within 1 or 2 seconds. After that the unmixing matrices keep changing a little, though not as fast as in the first few seconds. After all the data is passed (i.e. after 10 seconds), the unmixing matrices are still changing. They decrease towards zero very slowly. If this decreasing continues, the algorithm does not converge. In order to investigate the convergence behaviour of the algorithm, a simulation is done where, in contrast to all previous simulations, the 10 second data is given to the algorithm more than once. In fact in this simulation, the data is given 100 times to the algorithm. In real time this would mean a period of 16 minutes and 40 seconds. During this simulation, an unmixing matrix for a certain frequency bin k is studied. Convergence for this matrix can be examined. The conditions for this simulation are the same as for the simulation of Figure 5.5.

In Figure 5.6 the result is plotted for this simulation.

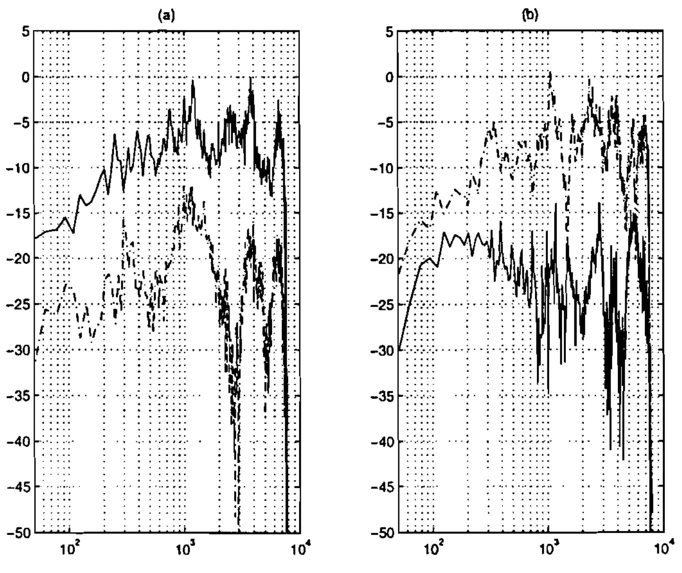

Figure 5.6: Performance after 100 passes through the data

From Figure 5.6 it can be seen that the transfer function from source 1 to output 1 doesn't change much between one pass of the data and 100 passes of the data. The transfer function from source 2 to output 1 is suppressed a little more, by approximately 5 dB. Apparently most of the separation is done in the first few seconds. After that only little improvement is achieved. This is a promising conclusion for implementation in real-time. For practical use of the algorithm (separation of two persons speaking at the same time in the same room), the mixing matrix will change in time. This will happen if for example one person turns his head a little bit. Then the transfer functions from that person to the microphones will change immediately. Because the algorithm adapts quickly, these changing conditions might not be a problem. Real-time implementation of this algorithm has not yet been done. This problem is open for further research.

From Figure 5.6 little can be said about convergence due to the normalisation. Therefore the unmixing matrix for one particular bin is followed during the updating. In Figure 5.7 these results are plotted for bin $k = 22$. Considering the FFT-size of 1024 and the sample rate of 16 kHz, this corresponds to a frequency of $\frac{\text{sample rate}}{\text{FFT-size}}k = \frac{16000}{1024}22 = 344$ Hz. On the vertical axis the weight value is plotted, the horizontal axis is the number of updates (time index). Again for example the upper right plot represents the weight for mixture 2 to output 1 in frequency bin $k = 22$.

From Figure 5.7 it can be seen that in the end the unmixing matrix for frequency bin $k = 22$ becomes

$$\mathbf{W}^F[22] = \begin{pmatrix} 0.05 & -0.032 \\ -0.01 & 0.04 \end{pmatrix} \tag{5.4}$$

The fast variations in Figure 5.7 are due to the repeating of the 10 second data. At the end of the data, the signal is suddenly cut off and the data starts from the beginning again. But at the end of the data the signal is different than at the beginning of the data. This may lead to a sudden change in the update of the weights. These fast variations are thus caused by the way of simulation. When applying a real long data stream, these variations will not occur. The other, very slow, variations can be explained as follows. The solution to the problem, separating independent signals, can be achieved by many matrices. There is a wide range of matrices that achieve separation. The algorithm can slowly vary from one solution to another without giving up the separation. Eventually it will 'choose' one solution and the unmixing matrix converges.

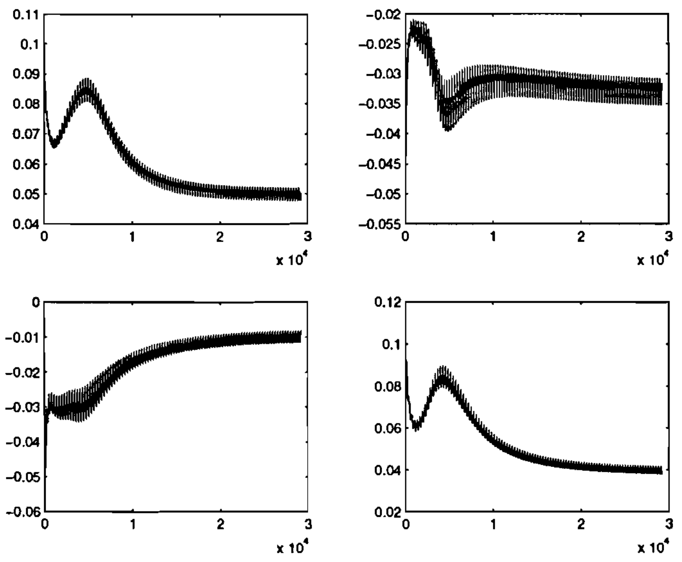

Figure 5.7: Convergence plot for unmixing matrix at frequency bin k

In conclusion the algorithm seems to converge. Convergence is not mathematically derived. This may be interesting for future research.

Track 9: $y_1(n)$ and $y_2(n)$ after 100 passes through the data

5.2.4 Non-Linear Function

Two non-linear functions are discussed in Section 4.3. Simulations have been done with both functions. As mentioned earlier, all previous simulations have been done with (4.9) as the non-linear function. Here the same simulation is done as in Figure 5.5, except that the non-linear function is now (4.11). In Figure 5.8 the result is plotted.

Comparing Figure 5.8 with Figure 5.5, the transfer functions are almost identical. Listening to the results, it is not possible to hear any differences between both results. In conclusion both non-linear functions have the same separating capabilities and therefore both functions can be used.

Track 10: $y_1(n)$ and $y_2(n)$ as Track 8 but with non-linear function (4.11)

5.2.5 Time Domain window

The influence of the block *TD-window* in Figure 4.2 on the permutation problem will be examined by simply omitting it from the algorithm. Again the same simulation is used as in Figure 5.5. The result of omitting *TD-window* can be seen in Figure 5.9.

It can be seen in Figure 5.9 that omitting *TD-window* causes permutation problems. At frequencies around 100 Hz and 300 Hz the ordering of the outputs switches. This is typically for permutation problems. Listening to the results, components of the other (suppressed) signal at those particular frequencies will sound louder than the same components of the wanted signal. Only twice the permutation was changed. This will not make the separation worthless, but it certainly is a degration of the performance. In the case where *TD-window* is not omitted, the 'leakage' from neighbour bins, as discussed before, is very small, because the length of *TD-window* (512) is half the FFT-size (1024). Therefore the frequency transform of *TD-window* will be similar to the one depicted in Figure 4.8. If even without this *TD-window* some permutation problems occur, while with it no permutation problems occured, it is clear that the presence of *TD-window*

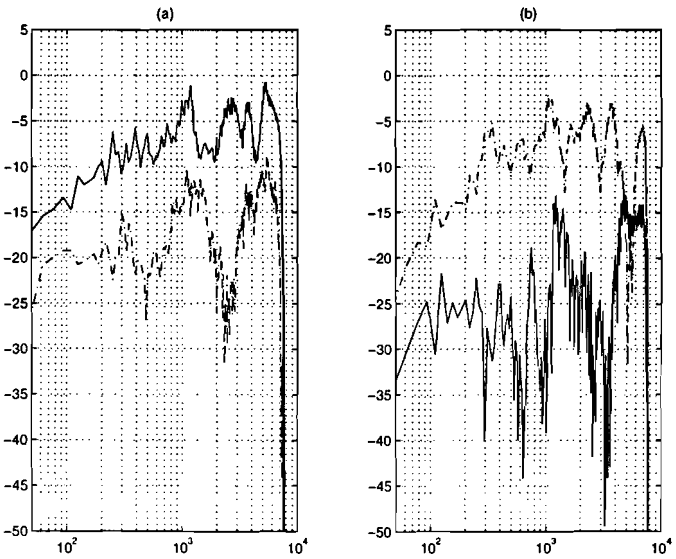

Figure 5.8: Performance with different non-linear function

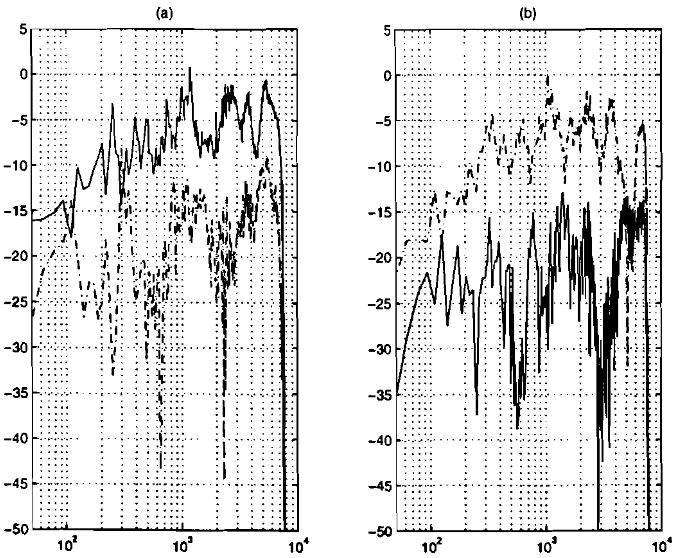

Figure 5.9: Performance without *TD-window*

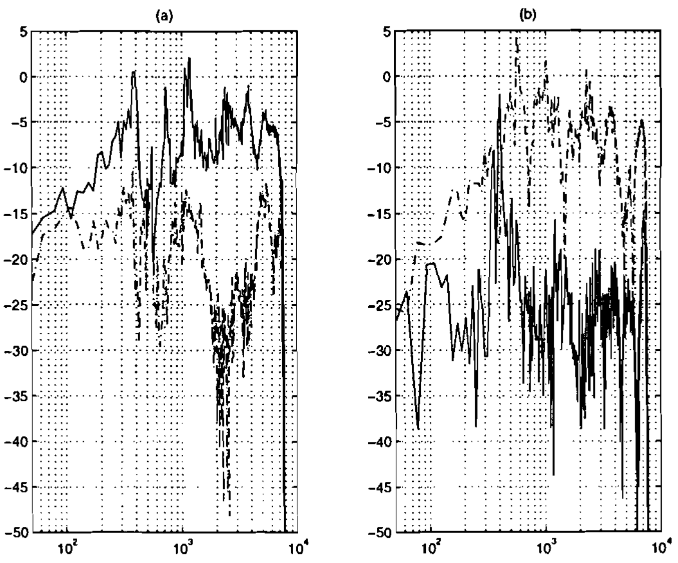

Figure 5.10: Performance with normalisation inside update cycle

helps avoiding permutation problems. With the inclusion of *TD-window*, permutation problems never occured in simulations done for this report.

Track 11: $y_1(n)$ and $y_2(n)$ as Track 8 but without TD-window in the update

5.2.6 Normalisation in Update

A simulation has been done to investigate the behaviour of the algorithm when the normalisation is placed inside the update cycle, thus when after every update the unmixing matrices are normalised. Again the same simulation parameters are used as in Figure 5.5. Only this time the initialisation is

$$\underline{w}_{11} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \underline{w}_{12} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \underline{w}_{21} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad \underline{w}_{22} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{5.5}$$

The effect of the normalisation in the update cycle is more clear using this initialisation. The original initialisation was a better approximation of the unmixing matrix than this new initialisation. The effect of the normalisation becomes more evident when the initialisation does not help the separation too much. The simulation from Figure 5.5 maintains the same result with this new initialisation. The result of the simulation with the normalisation inside the update cycle is plotted in Figure 5.10.

It can be seen that the performance is degraded. Permutation problems occur. These permutation problems result in poor audio quality. The separation achieved is also less than in Figure 5.5. Source 2 is suppressed approximately 5 dB more for output 1 in Figure 5.5 than in Figure 5.10.

It is clear that the normalisation must be placed outside the update cycle. The effect of the normalisation on the separating capabilities of the algorithm is too much.

Track 12: $y_1(n)$ and $y_2(n)$ with normalisation placed inside update cycle

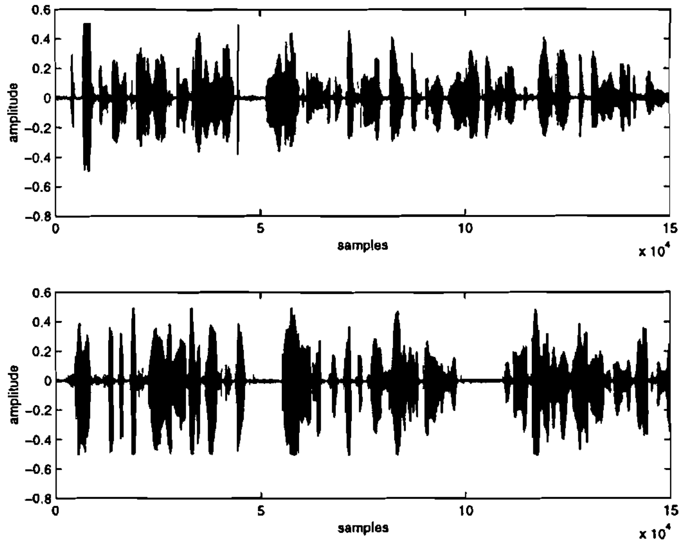

Figure 5.11: Wave-file plots for speech sources 1 and 2

5.3 Separation of Speech

After the frequency domain blind source separation algorithm is studied using two music sources, it is now tested for two speech signals. When two persons are talking, there are moments when they are silent. In that situation, there will be only one or even no source. It is interesting to see what the algorithm does when such a situation occurs.

Two speech signals are recorded, one female and one male reading a book. The persons are talking directly into the microphone. These signals are plotted in Figure 5.11.

These sources are then mixed using the same mixing matrix as in the simulations with the music sources. Exactly the same simulation parameters are used as in the simulation from Figure 5.5. The results are plotted in Figure 5.12.

It can be seen that the performance of the algorithm is not as good as in the case of music sources. Separation is achieved, though not as good as in Figure 5.5. The suppression is a few dB less. The mixed speech signals and the separated speech signals are plotted in Figure 5.13. The upper two signals are the mixtures, while the lower two signals are the outputs.

Track 13: $x_1(n)$ *and* $x_2(n)$ *for speech sources*
Track 14: $y_1(n)$ *and* $y_2(n)$ *for speech sources*

When one person stops talking, the algorithm may try to find another 'source'. For example it may try to separate some background noise. By doing this, the unmixing matrices will change. When the person starts talking again, the unmixing matrices are degraded for separating the speaker. This will hinder convergence.

In general problems might occur when there are less sources than assumed. This problem is still open for research. A possible solution is to let the learning rate depend on the power of the input signal. When the input power is very low, the speaker (or source in general) probably will be silent. If the learning rate is then changed to a very low value, the effect of the source being silent will be negligible. On the other hand, when the input power is very high, much information is given to the algorithm and the learning rate can be increased.

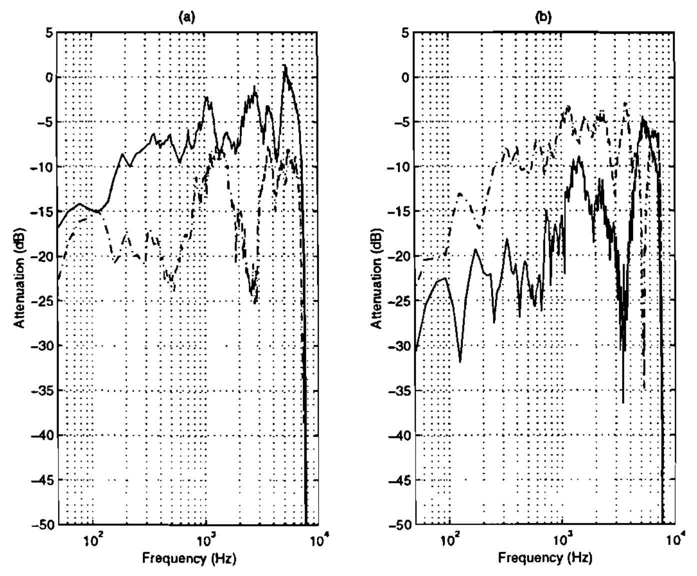

Figure 5.12: Performance for two speech sources

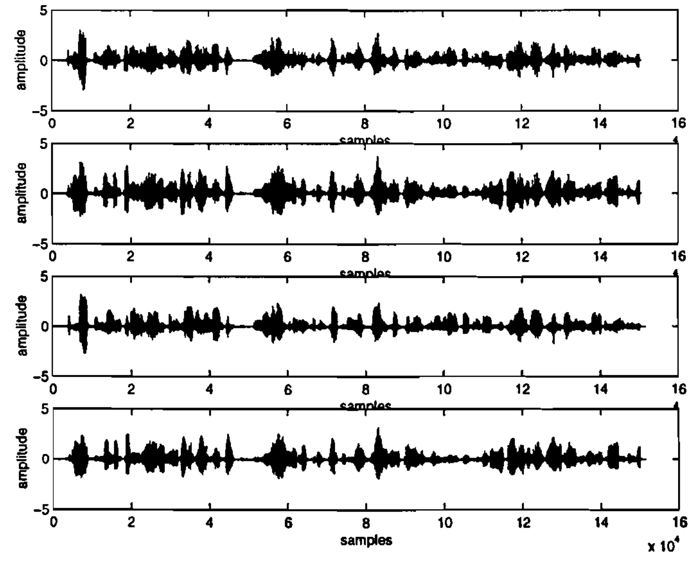

Figure 5.13: Wave-file plots for mixed and separated speech signals

As discussed in this section, separation of speech is more complex due to silent intervals. But it is also shown, that separation is possible. If this algorithm is implemented in real-time, and maybe improved for silent intervals, it will be very interesting for a lot of practical real-time applications, such as teleconferencing.

Chapter 6

Conclusions

The frequency domain blind source separation algorithm presented in this report is able to perform blind source separation for convolutive mixtures. Separation is achieved within a few seconds. The unwanted sources are suppressed by approximately 10-15 dB, depending on the frequency. For frequencies where the signals dominate, better separation is achieved. The performance of the algorithm (dB suppression), depends on the mixing matrix. For difficult situations, such as a large hall with a lot of reverberation, the mixing filters will be longer. This will make it harder for the algorithm to achieve separation. For small rooms, the algorithm has shown its capability of separating the sources.

The unmixing filters found by the algorithm behave as high-pass filters for low frequent sources. Therefore a normalisation is introduced. The normalisation reduces the attenuation of the low frequencies by approximately 15 dB, leading to better audio quality in the outputs. The normalisation is only applied on the outputs. It does not interfere with the update of the unmixing matrices, because then the separating capabilities of the algorithm are damaged.

The algorithm achieves separation very fast (1 or 2 seconds). After that only little improvement is gained. In all simulations done the algorithm converged to a solution.

A time domain window is applied to ensure that the unmixing filters remain linear. This window also helps to solve the permutation problem. The algorithm gives the same permutation of the outputs in every frequency bin. Only for high frequencies, sometimes the permutation was changed. However this does not affect the separation, because the signals to be separated contained no or very little information in those frequencies. These permutation problems are inaudible.

Separating speech signals slightly degrades the performance of the algorithm, because speech signals contain silent intervals. During these intervals, the algorithm cannot separate or even maybe starts trying to separate e.g. background noise. However separation is still achieved. To perform better in the situation where one or more sources are silent for a short period of time, a modification such as an input power dependent learning rate can be made.

Implementation of the frequency domain algorithm in real-time is possible. This implementation can then be used as a demonstration on Blind Source Separation.

Chapter 7

Future Work

Due to the limited time set for this research, some problems are left open for future research. One of them is the modification of the algorithm to deal with silent intervals. This situation occurs for example when the sources are speakers. Persons are not talking all the time. In order to make sure the algorithm performs correctly during these silent intervals, a modification may be needed. Possibly a learning rate that is dependent on the power of the input signals is a solution to this problem.

The implementation in real-time is another subject for future work. Implementing the frequency domain algorithm discussed in this report will give a nice demonstration on Blind Source Separation.

Another subject is to give a mathematical derivation on why it is possible to achieve separation by performing independent separation within each frequency bin. The simulations discussed in this report show that it is possible, but a proper derivation has not been given yet.

The convergence properties of the algorithm also have not been derived mathematically. The algorithm converges for all simulations done in this report, but a systematic study has not been done yet.

Blind Source Separation has received a lot of attention in the past few years, because there are a lot of applications. Therefore the research on this area of signal processing will continue in order to improve the algorithms and achieve better separation.

Appendix A

List of tracks on CD

All tracks are sampled with 16 kHz, but for this CD upsampled to 44.1 kHz.

Track 1: Music source 1 'spOOL'

Track 2: Music source 2 'Peruvian Skies'

Track 3: $y_1(n)$ *and* $y_2(n)$ *for instantaneous mixing matrix with adaptive filter length 1*

Track 4: $y_1(n)$ *and* $y_2(n)$ *for instantaneous mixing matrix with adaptive filter length 512*

Track 5: $x_1(n)$ *and* $x_2(n)$ *with convolutive mixing matrix of length 512*

Track 6: $y_1(n)$ *and* $y_2(n)$ *for convolutive mixing and unmixing matrix of length 512 without normalisation*

Track 7: $y_1(n)$ *and* $y_2(n)$ *for convolutive mixing and unmixing matrix of length 512 with normalisation*

Track 8: $y_1(n)$ *and* $y_2(n)$ *as Track 7 only after 100 passes through the data*

Track 9: $y_1(n)$ *and* $y_2(n)$ *as Track 7 but with non-linear function (4.11)*

Track 10: $y_1(n)$ *and* $y_2(n)$ *as Track 7 but without TD-window in the update*

Track 11: $y_1(n)$ *and* $y_2(n)$ *as Track 7 but with normalisation placed inside update cycle*

Track 12: $x_1(n)$ *and* $x_2(n)$ *with mixing and unmixing matrix of length 512 for speech sources*

Track 13: $y_1(n)$ *and* $y_2(n)$ *with mixing and unmixing matrix of length 512 for speech sources*

Acknowledgements

I'd like to thank my supervisors Piet Sommen and Daniel Schobben for their support. For every day problems with Unix, Matlab and LaTeX, Daniel was always prepared to help me out. In our weekly meetings, Piet made sure that the structure of the project was maintained. Their comments and suggestions were very useful. During this project I've learned some valuable lessons for my further career, and that is exactly what a graduation project is meant for. Concerning the usual computer trouble, I also want to thank Jurgen van Engelen for being an expert. Further I'd like to thank my parents for letting me be who I am and supporting me under any condition. At last but certainly not least I'd like to thank my girlfriend Yvette for just being herself and putting up with me.

To everybody who supported me one way or another, Thanks!

Bibliography

[1] Cardoso, J.F., "Information maximisation and maximum likelihood for blind source separation", IEEE Signal Processing Letters, Vol.4, No.4, p.112–114,Apr 1997.

[2] Choi, Seungjin; Ruey-Wen Liu, "Two-stage neural network for blind sources separation", Proceedings of the 39th Midwest symp. on Circuits and Systems, Vol.2, p.827–830 Ames, USA, 18-21 Aug.1996, Ed. by G. Cameron et al., IEEE, New York.

[3] Amari, S.; A. Cichocki; H.H. Yang, "A new learning algorithm for blind signal separation", in Advances in Neural Information Processing 8, Proceedings of the 1995 Conference, p.757-763, Denver, USA, 27-30 Nov.1995, Ed. by D.S. Touretzky et al., MIT Press, Cambridge.

[4] Belouchrani, A.; A. Cichocki; K.A. Meraim, "A blind identification and separation technique via multi-layer neural networks", Progress in Neural Information Processing. Proceedings of the Int. Conf. on Neural Information Processing, Vol.2, p.1195-2000, Hong Kong, 24-27 Sept.1996, Ed. by S. Amari et al., Springer-Verlag, Singapore.

[5] Bell, A.J.; T.J. Sejnowski, "A non-linear information maximisation algorithm that performs blind separation", Advances in Neural Information Processing 7, p.467-474, Denver, USA, 28 Nov-3 Dec 1994, Ed. by G. Tesauro, MIT Press.

[6] Cichocki, A.; R. Unbehauen; L. Moszczynski; E. Rummert, "A new on-line adaptive learning algorithm for blind separation of source signals", 1994 Int. Symp. on Artificial Neural Networks, Proceedings, p.406-411, Tainan, Taiwan, 15-17 Dec. 1994, Nat. Cheng Kung Univ.

[7] Cichocki, A.; R. Unbehauen; E. Rummert, "Robust learning algorithm for blind separation of signals", Electronic Letters, Vol. 30, Iss. 17, p.1386-1387, 18 Aug 1994.

[8] Lambert, R.H., "Multichannel blind deconvolution: FIR matrix algebra and separation of multipath mixtures (Bussgang property. Wiener filtering", University of Southern California, 1996.

[9] Choi, Seung-Jin, "Adaptive blind sources separation based on unsupervised learning (neural networks)", University of Notre Dame, 1996.

[10] Comon, P.; C. Jutten; J. Herault, "Blind separation of sources: II. Problems statement", Signal Processing, Vol.24, Iss.1, p.11-20, July 1991.

[11] Yellin, D.; E. Weinstein, "Criteria for multichannel signal separation", IEEE Transactions on Signal Processing, Vol.42, Iss.8, p.2158-2168, Aug.1994.

[12] Karhunen, J.; J. Joutsensalo, "Representation and separation of signals using nonlinear PCA type learning", Neural Networks, Vol.7, Iss.1, p.113-127, 1994.

[13] Kawamoto, M.; K. Matsuoka; M. Oya, "Blind separation of sources using temporal correlation of the observed signals", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E80-A, Iss.4, p.695-704, april 1997.

[14] Puntonet, C.G.; A. Prieto, "Geometric approach for blind separation of signals", Electronic Letters, Vol.33, Iss.10, p.835-836, 8 May 1997.

[15] Burel, G., "Blind separation of sources - A nonlinear neural algorithm", Neural networks, Vol.5, Iss.6, p.937-947, 1992.

[16] Bell, A.J.; T.J. Sejnowski, "An information maximisation approach to blind separation and deconvolution", Neural Computation, Vol.7, Iss.6, p1129-1159, 1995.

[17] Matsuoka, K.; M. Ohya; M. Kawamoto, "A neural-net for blind separation of nonstationary signals", Neural Networks, Vol.8, Iss.3, p.411-419, 1995.

[18] Cichocki, A.; R. Unbehauen, "Robust neural networks with online learning for blind identification and blind separation of sources", IEEE Transactions on Circuits and Systems I. Fundamental theory and applications, Vol.43, Iss.11, p.894-906, 1996.

[19] Eom, K.S.; B.C. Lim; D.J. Park, "Blind separation algorithm without nonlinear functions", Electronics letters, Vol.32, Iss.3, p.165-166, 1996.

[20] Yellin, D.; E. Weinstein, "Multichannel signal separation - Methods and analysis", IEEE Transactions on Signal Processing, Vol.44, Iss.1, p.106-118,1996.

[21] Karhunen, J.; J. Joutsensalo, "Generalizations of Principal Component Analysis, optimisation problems and neural networks", Neural Networks, Vol.8, Iss.4, p.549-562, 1995.

[22] Karhunen, J.; E. Oja; L.Y. Wang; R. Vigario; J. Joutsensalo, "A class of neural networks for Independent Component Analysis", IEEE Transactions on Neural Networks, Vol.8, Iss.3, p.486-504, 1997.

[23] Pham, D.T.; P. Garat, "Blind separation of mixture of independent sources through a quasi-maximum likelihood approach", IEEE Transactions on Signal Processing, Vol.45, Iss.7, p.1712-1724, 1997.

[24] Cardoso, J.F.; B.H. Laheld, "Equivariant adaptive source separation", IEEE Transactions on Signal Processing, Vol.44, Iss.12, p.3017-3030, 1996.

[25] Wang, Chuan, "An information-theoretic perspective for learning systems with engineering applications (adaptive filters, neural networks)", University of Florida, 1996.

[26] Comon, P., "Independent component analysis, anew concept?", Signal Processing, Vol.36, p.287-314, 1994.

[27] Jutten, C.; J. Herault, "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture", Signal Processing, Vol.24, p.1-10, 1991.

[28] Li, S.; T.J. Sejnowski, "Adaptive separation of mixed broadband sound sources with delays by a beamforming Herault-Jutten network", IEEE Journal of Oceanic Engineering, Vol.20, Iss.1, p.73-79, 1994.

[29] Bellini, S., "Bussgang techniques for blind deconvolution and equalization", in Blind deconvolution, Ed. S. Haykin, p.8-52, Englewood Cliffs, New Jersey, Prentice Hall, 1994.

[30] Chan, D.B., "Blind signal separation", University of Cambridge, 19 Jan. 1997.

[31] Cichocki, A.; S. Amari; M. Adachi; W. Kasprzak, "Self-adaptive neural networks for blind separation of sources", 1996 IEEE Int. Symp. on Circuits and Systems. Circuits and Systems connecting the world, p.157-160, Vol.2, Atlanta, USA, 12-15 May 1996.

[32] Principe, J.C.; Chuang Wang; Hsiao-Chun Wu, "Temporal decorrelation using teacher forcing anti-Hebbian learning and its application in adaptive blind source separation", Neural Networks for Signal Processing IV, proceedings of the 1996 IEEE Signal Processing Society Workshop, p.413-422, Kyoto, Japan, 4-6 Sept 1996.

[33] Wang, J.; Z. He, "Blind identification and separation of convolutively mixed independent sources", IEEE Transactions on Aerospace.., 1997, Vol.33, No.3, p.997.

[34] Lambert, R.H.; A.J. Bell, "Blind separation of mutliple speakers in a multipath environment", 1997 Int. Conf. on Acoustics, Speech and Signal Processing, p.423-426, Vol.I, 21-24 Apr 1997, Munich, Germany.

[35] Amari, S.-I., "Differential-geometric methods in statistics", Lecture Notes in Statistics, Vol. 28, 1985, Springer.

[36] Smaragdis, P., "Blind separation of convolved mixtures in the frequency domain", to be published.

[37] Sommen, P.C.W., "Adaptive filtering methods: on methods to use a priori information in order to reduce complexity while maintaining convergence properties", PhD Thesis, Eindhoven Technical University of Technology, 1992.

[38] Smaragdis, P., "Information theoretic approaches to source separation", Massachusetts Institute of Technology, June 1997.

[39] Torkkola, K., "Blind separation of delayed sources based on information maximisation", Proc. IEEE ICASSP, Atlanta, May 1996.

[40] Torkkola, K., "Blind separation of convolved sources based on information maximisation", Proc. IEEE Workshop on Neural Networks and Signal Processing, Kyota, Japan, Sept. 1996.