38th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, Nov. 7-10, 2004, Vol. 1, pp. 983–988.

Convolutive Blind Signal Separation in Acoustics by Joint Approximate Diagonalization of Spatiotemporal Correlation Matrices

Marcel Joho

Phonak Inc., Champaign, IL, USA, (joho@ieee.org)

Abstract

We present an efficient algorithm for the blind signal separation (BSS) problem with convolutive signal mixtures, as it usually appears in Acoustics, e.g., in the cocktail party problem. Since acoustical signals are typically non-stationary and non-white, we make use of these two statistical properties in the formulation of the blind cost function. In order to achieve true signal separation, the algorithm aims at finding a single polynomial matrix, the convolutive separation matrix, that jointly diagonalizes a set of measured spatiotemporal correlation matrices. Minimizing the cost function turns out to be mathematically equivalent to a convolutive joint approximate diagonalization problem (CJAD).

In order to increase the initial convergence rate, a Armijo line search is incorporated into the update. The final algorithm operates primarily in the frequency domain (fast convolution techniques) even though the cost function and gradient are formulated in the time domain. This approach reduces the computational complexity and avoids the so-called permutation problem.

1 Introduction

1.1 Problem formulation

1.1.1 Signal mixing

The system setup is depicted in Fig. 1. M_s unknown mutually uncorrelated *source signals* s_m are filtered and mixed by an unknown time-invariant finite-length causal *convolutive mixing system* $\mathbf{A}^{M_\text{x}\times M_\text{s}} \triangleq \{\mathbf{A}_n\}_{n=0}^{N_\text{a}}$ resulting in M_x measurable sensor signals x_m. The source- and sensor-signals are stacked in vectors, $\mathbf{s}$ and $\mathbf{x}$, respectively. For simplicity we neglect any additive noise components. Hence, the convolutive mixing process is described as $\mathbf{x}=\mathbf{A}*\mathbf{s}$ where

$$\mathbf{x}(t) = (\mathbf{A}*\mathbf{s})(t) = \sum\nolimits_{n=0}^{N_\text{a}} \mathbf{A}_n\,\mathbf{s}(t-n) \tag{1}$$

or, written in the z-domain,

$$\mathbf{x}(z) \triangleq \sum\nolimits_t \mathbf{x}(t)\,z^{-t} = \mathbf{A}(z)\,\mathbf{s}(z)\,. \tag{2}$$

1.1.2 Signal separation

The M_x sensor signals x_m are mixed and filtered with a finite-length non-causal convolutive separation system $\mathbf{W}^{M_\text{u}\times M_\text{x}} \triangleq \{\mathbf{W}_n\}_{n=-N_\text{w}}^{N_\text{w}}$ resulting in M_u output signals u_m. The separation process is described as $\mathbf{u}=\mathbf{W}*\mathbf{x}=\mathbf{W}*\mathbf{A}*\mathbf{s}$, or written in the z-domain

$$\mathbf{u}(z) = \mathbf{W}(z)\,\mathbf{x}(z) = \mathbf{W}(z)\mathbf{A}(z)\,\mathbf{s}(z)\,. \tag{3}$$

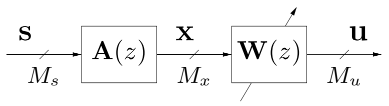

Figure 1: Setup of convolutive mixing / demixing system.

The objective of the blind-source-separation problem for the convolutive mixing case is to find a $\mathbf{W}(z)$ such that the global system can be written as

$$\mathbf{G}(z) = \mathbf{W}(z)\,\mathbf{A}(z) = \mathbf{P}\,\mathbf{D}(z) \tag{4}$$

where $\mathbf{D}(z)$ is a diagonal polynomial matrix and $\mathbf{P}$ is a permutation matrix. In the following we assume that $M_\text{u} = M_\text{s} \le M_\text{x}$, and that the source signals and mixing system can be complex valued. Depending on $\mathbf{A}(z)$, M_x, and M_s perfect separation is possible for a finite N_w.

1.2 Mathematical preliminaries

1.2.1 Basic notation

The notation used throughout this paper is the following: Vectors are written in lower case, matrices in upper case. Matrix and vector transpose, complex conjugation and Hermitian transpose are denoted by $(.)^T$, $(.)^*$, and $(.)^H \triangleq ((.)^*)^T$, respectively. The sample index is denoted by t. The identity matrix is denoted by $\mathbf{I}$, a vector or a matrix containing only zeros by $\mathbf{0}$. $E\{.\}$ denotes the expectation operator. The Frobenius norm and the trace of a matrix are denoted by $\|.\|_F$ and $\text{tr}\{.\}$, respectively. $\text{diag}(\,\mathbf{A}\,)$ zeros the off-diagonal elements of $\mathbf{A}$ and

$$\text{off}(\,\mathbf{A}\,) \triangleq \mathbf{A} - \text{diag}(\,\mathbf{A}\,) \tag{5}$$

zeros the diagonal elements of $\mathbf{A}$. The extension for polynomial matrices is defined straightforwardly as $\text{off}(\,\mathbf{A}(z)\,) \triangleq \mathbf{A}(z) - \text{diag}(\,\mathbf{A}(z)\,) = \sum_n \text{off}(\,\mathbf{A}_n\,)z^{-n}$. Linear convolution between two sequences is denoted by $*$. Furthermore, let $\mathbf{A}(z) \triangleq \sum_n \mathbf{A}_n z^{-n}$. Then we define

$$\mathbf{A}^\dagger(z) \triangleq \mathbf{A}^H(1/z^*) = \sum\nolimits_n \mathbf{A}_n^H z^{+n}\,. \tag{6}$$

where $\mathbf{A}^\dagger(z)$ is said to be the paraconjugate of $\mathbf{A}(z)$, see [1].

1.2.2 Inner products and induced norms

A commonly chosen *inner product* between two matrices $\mathbf{A}$ and $\mathbf{B}$ is $\langle \mathbf{A}, \mathbf{B} \rangle \triangleq \mathrm{tr}\{\mathbf{A}\mathbf{B}^H\}$. The corresponding *induced norm* is the *Frobenius norm* $\|\mathbf{A}\|_F \triangleq \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle}$.

The Frobenius norm can also be extended to polynomial matrices. Let $\mathbf{A}(z) \triangleq \sum_n \mathbf{A}_n z^{-n}$ and $\mathbf{B}(z) \triangleq \sum_n \mathbf{B}_n z^{-n}$ be two polynomial matrices with *finite energy*, i.e., $\sum_n \|\mathbf{A}_n\|_F^2 < \infty$ or $\sum_n \|\mathbf{B}_n\|_F^2 < \infty$. Then an *inner product* can be defined as [2,3]

$$\langle \mathbf{A}(z), \mathbf{B}(z) \rangle_{\mathcal{F}} \triangleq \sum\nolimits_n \langle \mathbf{A}_n, \mathbf{B}_n \rangle = \sum\nolimits_n \mathrm{tr}\{\mathbf{A}_n \, \mathbf{B}_n^H\} . \quad (7)$$

The induced norm becomes

$$\|\mathbf{A}(z)\|_{\mathcal{F}} \triangleq \sqrt{\langle \mathbf{A}(z), \mathbf{A}(z) \rangle_{\mathcal{F}}} = \sqrt{\sum\nolimits_n \|\mathbf{A}_n\|_F^2} \quad (8)$$

which is the extention of the Frobenius norm to polynomial matrices. Norms provide a convenient way to measure a distance between two matrices or polynomial matrices, as they induce a *metric* [4]. For example, we can measure the “distance” between two polynomial matrices $\mathbf{A}(z)$ and $\mathbf{B}(z)$ as $d(\mathbf{A}(z), \mathbf{B}(z)) \triangleq \|\mathbf{A}(z) - \mathbf{B}(z)\|_{\mathcal{F}}$. In our case, we will use $d(.,.)$ to measure the distance between two spatiotemporal correlation matrices. Interestingly, the paraconjugate operator defined in (6) is the Hilbert-adjoint operator [4] for polynomial matrices, i.e., $\langle \mathbf{T}(z)\,\mathbf{A}(z)\,, \mathbf{B}(z) \rangle_{\mathcal{F}} = \langle \mathbf{A}(z)\,, \mathbf{T}^\dagger(z)\,\mathbf{B}(z) \rangle_{\mathcal{F}}$.

We can also define an inner product for the case where $\mathbf{A}(z)$ and $\mathbf{B}(z)$ are transformed into the frequency domain, i.e., $\mathbf{A}(e^{j\omega})$ and $\mathbf{B}(e^{j\omega})$, respectively, as

$$\left\langle \mathbf{A}(e^{j\omega}), \mathbf{B}(e^{j\omega}) \right\rangle_{\mathcal{F}} \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} \left\langle \mathbf{A}(e^{j\omega}), \mathbf{B}(e^{j\omega}) \right\rangle d\omega \quad (9)$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathrm{tr}\{\mathbf{A}(e^{j\omega})\mathbf{B}^H(e^{j\omega})\} d\omega \, . \quad (10)$$

1.2.3 First-order Taylor series approximation in matrix-polynomial form

Taylor-series approximation is an important tool in optimization problems. In multivariate optimization the parameters are usually arranged in vector form. However, in some applications, such as the joint diagonalization problem, the natural arrangement of the parameters is in matrix form [5]. For this case, an elegant form of the Taylor series approximation is given by Manton [6]

$$\mathcal{J}(\mathbf{W} + \delta\,\mathbf{Z}) = \mathcal{J}(\mathbf{W}) + \delta\,\mathrm{Re}\{\langle \mathbf{D}_\mathbf{W}, \mathbf{Z} \rangle\} + O(\delta^2) \quad (11)$$

where $\mathbf{D}_\mathbf{W}$ denotes the derivative or gradient of the cost function $\mathcal{J}$ at $\mathbf{W}$ and $\langle \mathbf{D}_\mathbf{W}, \mathbf{Z} \rangle \triangleq \mathrm{tr}\{\mathbf{Z}^H \mathbf{D}_\mathbf{W}\}$.

For the convolutive joint diagonalization problem, the natural arrangement of the parameters is given in polynomial-matrix form, i.e., $\mathbf{W}(z)$. In this case, we can extend the formulation of the Taylor series in (11) to

$$\mathcal{J}(\mathbf{W}(z) + \delta\,\mathbf{Z}(z)) \approx \mathcal{J}(\mathbf{W}(z)) + \delta\,\mathrm{Re}\left\{\langle \mathbf{D}_\mathbf{W}(z), \mathbf{Z}(z) \rangle_{\mathcal{F}}\right\} \quad (12)$$

where $\langle .,. \rangle_{\mathcal{F}}$ is defined in (7) and $\mathbf{D}_\mathbf{W}(z)$ is referred to as the gradient of $\mathcal{J}$ at $\mathbf{W}(z)$.

1.2.4 Signals and spatiotemporal correlations

We use the following notation: $x_m(t)$ denotes the value of the signal x_m at discrete time t and $x_m \triangleq \{x_m(t)\}$ denotes the time series of signal x_m. Furthermore, we define $\mathbf{x}(t) \triangleq (x_1(1), \ldots, x_{M_x}(t))^T$ and $\mathbf{x} \triangleq (x_1, \ldots, x_{M_x})^T = \{\mathbf{x}(t)\}$. The spatiotemporal correlation matrix between two signal vectors $\mathbf{u}$ and $\mathbf{x}$, and the corresponding z-transform of the correlation sequence are defined as

$$\mathbf{R}_{\mathbf{ux}}(\tau;t) \triangleq E\{\mathbf{u}(t)\,\mathbf{x}^H(t-\tau)\} \quad (13)$$

$$\mathbf{R}_{\mathbf{ux}}(z;t) \triangleq \sum\nolimits_{\tau=-\infty}^{\infty} \mathbf{R}_{\mathbf{ux}}(\tau;t) z^{-\tau} \, , \quad (14)$$

respectively. For stationary signals we have $\mathbf{R}_{\mathbf{ux}}(\tau;t) = \mathbf{R}_{\mathbf{ux}}(\tau)$ and, hence, $\mathbf{R}_{\mathbf{ux}}(z;t) = \mathbf{R}_{\mathbf{ux}}(z)$.

2 A joint-diagonalization approach

2.1 Blind cost function

A well-designed cost function is essential in optimization problem. In our case, we choose the same *blind cost function* as was used in [3]:

$$\mathcal{J}_1(\mathbf{W}(z)) \triangleq \sum_{p=1}^{P} \|\ \mathrm{off}\ (\mathbf{R}_{\mathbf{uu}}(z;t_p))\ \|_{\mathcal{F}}^2 \quad (15)$$

$$= \sum_p \|\ \mathrm{off}\ (\mathbf{W}(z)\mathbf{R}_{\mathbf{xx}}(z;t_p)\mathbf{W}^\dagger(z))\ \|_{\mathcal{F}}^2 . \quad (16)$$

where

$$\mathbf{R}_{\mathbf{xx}}(z;t_p) = \mathbf{A}(z)\mathbf{R}_{\mathbf{ss}}(z;t_p)\mathbf{A}^\dagger(z) \quad (17)$$

is the spatiotemporal correlation matrix of the sensor signals at time t_p and t_p denotes the center of the pth *snapshot*. The cost function (16) takes $P > 1$ snapshots into account as we assume that the source signals s_m are non-stationary. To be more precise, we assume that the source signals are stationary within the time-frame of a single snapshot, but are considered non-stationary over the time-frame of all P snapshots. Furthermore, as we assume that the source signals are mutually uncorrelated for all t, $\mathbf{R}_{\mathbf{ss}}(z;t_p)$ is modeled to have diagonal structure for every snapshot. We do not require that the source signals need to be white. The unknown mixing system $\mathbf{A}(z)$ is assumed to be time-invariant, hence, $\mathbf{W}(z)$ is chosen to be time-invariant as well.

The cost function (16) is designed to separate, but not deconvolve the unknown source signals. Furthermore, an optimal solution $\mathbf{W}(z)$, which minimizes (16), does not suffer the so-called permutation problem, as opposed to other commonly used cost functions for the same problem, e.g., [7] (See [8] for more details on how to avoid the permutation problem). In order to prevent the trivial solution $\mathbf{W}(z) \equiv \mathbf{0}$, which obviously minimizes (16), we need to impose some additional constraints on $\mathbf{W}(z)$, see [3] for more details. The optimization problem defined in (16) subject to some constraints, is referred to as a *joint-diagonalization problem*, as we wish to find a single polynomial matrix $\mathbf{W}(z)$ that jointly diagonalizes all products $\mathbf{W}(z)\mathbf{R}_{\mathbf{xx}}(z;t_p)\mathbf{W}^\dagger(z)$. In practical applications usually some of the assumptions are violated, such that the optimum $\mathbf{W}(z)$ will only approximately jointly diagonalize all products.

2.2 Gradient and update equation

Referring to the Taylor series representation (12), the gradient of the cost function (16) is

$$\mathbf{D}_{\mathbf{W}}(z) = \sum_{r=-N_\mathrm{w}}^{N_\mathrm{w}} \nabla_{\mathbf{W}_r} \mathcal{J}_1\big(\mathbf{W}(z)\big)\, z^{-r} \qquad (18)$$

where

$$\nabla_{\mathbf{W}_r} \mathcal{J}_1 = 4 \sum_{p=1}^{P} \sum_{\tau=-\tau_\mathrm{e}}^{\tau_\mathrm{e}} \mathrm{off}\,\big(\mathbf{R}_{\mathbf{uu}}(\tau; t_p)\big)\, \mathbf{R}_{\mathbf{ux}}(r-\tau; t_p) \qquad (19)$$

is the gradient of $\mathcal{J}_1$ at $\mathbf{W}(z)$ with respect to $\mathbf{W}_r$, see [3]. Furthermore,

$$\mathbf{R}_{\mathbf{ux}}(\tau; t_p) = \sum_m \mathbf{W}_m\, \mathbf{R}_{\mathbf{xx}}(\tau - m; t_p) \qquad (20)$$

$$\mathbf{R}_{\mathbf{uu}}(\tau; t_p) = \sum_m \sum_n \mathbf{W}_m\, \mathbf{R}_{\mathbf{xx}}(\tau - m + n; t_p)\, \mathbf{W}_n^H\,. \qquad (21)$$

Note that we only included a finite interval $\tau \in [-\tau_\mathrm{e}, \tau_\mathrm{e}]$ in the update (19). The update equation of the corresponding steepest descent algorithm is

$$\mathbf{W}(z)[k+1] = \mathbf{W}(z)[k] - \mu\, \mathbf{D}_{\mathbf{W}}(z)[k] \qquad (22)$$

where μ denotes the step size of the update.

2.3 Modified update equation

In [3] the update (22) was used with a constant step size μ. In order to prevent the algorithm to become unstable for every situation, the step size μ needed to be chosen fairly small. Whether the cost function $\mathcal{J}_1$ is poorly conditioned or not depends primarily on the power and spatiotemporal correlation properties of the input signals in all snapshots. Therefore, we aim at finding an update strategy that makes the convergence behavior more robust against the input signal properties. [1] To this end, we make a slight modification to the update equation (22), namely

$$\mathbf{W}(z)[k+1] = \mathbf{W}(z)[k] + \mu_k\, \mathbf{S}(z)[k]\,. \qquad (23)$$

The difference between (22) and (23) is mainly, that the update does not necessarily need to go in the direction of the negative gradient and, more importantly, that we can choose a different step size in every iteration. The motivation will become apparent once we introduce the concept of line-search methods in Section 3.

Note that (22) and (23) correspond, in fact, to a time-domain update equation, as the time-domain filter coefficients $\mathbf{W}_r$ are updated, i.e., $\mathbf{W}_r[k+1] = \mathbf{W}_r[k] + \mu_k\, \mathbf{S}_r[k]$ for $r \in [-N_\mathrm{w}, N_\mathrm{w}]$.

3 Line-search methods

3.1 General optimization techniques

Typically quadratic cost functions are preferred for a given optimization problem for several reasons. However, this is here not the case as the free parameters in the cost function (16), i.e., the elements of $\mathbf{W}(z)$, appear up to the fourth power. In previous work on convolutive BSS, most algorithms use a steepest-descent method with a constant step size μ for every update. The step size μ needs to be chosen small enough, such that the algorithm does not diverge for any type of input signal, but should also be chosen large enough, such that still a decent convergence rate can be achieved. The same is true for stochastic algorithms, such as the LMS or NLMS, which are commonly used in non-blind adaptive signal processing, e.g., adaptive beamforming or echo cancellation.

Taking a closer look at the cost function (16), we discover, that once the spatiotemporal correlation matrices $\{\mathbf{R}_{\mathbf{xx}}(z; t_p)\}$ are estimated, finding a minimum of (16) becomes a pure off-line optimization problem. In order to find a fast converging algorithm, we should make use of existing (non-stochastic) optimization techniques. Applying a full Newton method seems to be infeasible nowadays, as the dimension of the Hessian becomes extremely large. Moreover, a Newton method would show its largest benefit over a steepest-descent method in the vicinity of a local minima, as the cost function becomes there approximately quadratic.

Incorporating a *line-search method* into the optimization algorithm seems to be more promising, as this does not increase the computational complexity in the same order as a Newton method does. Moreover, a line-search method can improve the initial convergence in an ill-conditioned situation considerably compared to a fixed-step-size method. Usually inexact line-search methods are used in practice, whereas exact line-search methods are mostly used just for a convergence analysis. In the following, we will use an Armijo-line-search method [9] in order to improve the convergence rate of the algorithm.

3.2 Constraint optimization

As already mentioned in Section 2.1, we need to constrain $\mathbf{W}(z)$ somehow to prevent the algorithm of converging to the trivial solution $\mathbf{W}(z) \equiv \mathbf{0}$. We will choose to constrain $\mathrm{diag}(\,\mathbf{W}(z)\,) \equiv \mathbf{I}$, which is sometimes referred to as the *minimum distortion principle* [10]. To this end, we will initialize $\mathbf{W}(z)[0] = \mathbf{I}$ and then simply do not update the diagonal filters of $\mathbf{W}(z)$. This can also be achieved by always choosing a search direction who's diagonal filters are set to zero. In the following, we will choose the *search direction* $\mathbf{S}(z)$ in the kth iteration as

$$\mathbf{S}(z)[k] = -\,\mathrm{off}\,\big(\mathbf{D}_{\mathbf{W}}(z)[k]\big) \qquad (24)$$

which corresponds to a projection operation of the negative gradient.

3.3 Armijo line search

Among the different inexact line search methods, we decide to use the Armijo line-search method [9]. The Armijo line-search method will give a reasonable step size μ_k in every iteration that will neither be too large (instability) or very small (slow convergence). Furthermore, if at the kth iteration the search direction $\mathbf{S}_k(z)$ points along a feasible direction, then the Armijo line-search method will find a μ_k such that the cost function will

[1]This is somehow analog to an LMS algorithm that is modified to either an NLMS algorithm (to make the convergence independent of the input signal power), or even an RLS algorithm (to also make the convergence independent of the input signal correlation properties).

Armijo line-search routine

```
Input parameters:            $\mathbf{W}_k$, $\mathbf{S}_k$, $\mathbf{D}_{\mathbf{W}\,k}$
Output parameters:           $\mu_k$
Constants:                   $\mu_0$, $\eta$, $\gamma$, $\epsilon$
Early termination criteria:  $\mu\,\|\mathbf{S}_k\| < \epsilon$
Initialization:              $\mu := \mu_0$

if $\mathrm{Re}\{\langle \mathbf{D}_{\mathbf{W}\,k}, \mathbf{S}_k \rangle\} > 0$
    then $\mu := -\mu$
while $\mathcal{J}\big(\mathbf{W}_k\big) - \mathcal{J}\big(\mathbf{W}_k + \gamma^{-1}\mu\,\mathbf{S}_k\big)$
        $\geq -\eta\,\gamma^{-1}\,\mu\,\mathrm{Re}\{\langle \mathbf{D}_{\mathbf{W}\,k}, \mathbf{S}_k \rangle\}$
    do $\mu := \gamma^{-1} \cdot \mu$
while $\mathcal{J}\big(\mathbf{W}_k\big) - \mathcal{J}\big(\mathbf{W}_k + \mu\,\mathbf{S}_k\big)$
        $< -\eta\,\mu\,\mathrm{Re}\{\langle \mathbf{D}_{\mathbf{W}\,k}, \mathbf{S}_k \rangle\}$
    do $\mu := \gamma \cdot \mu$
return $\mu_k := \mu$
```

Figure 2: Armijo line search method for the matrix form. Here $\|.\| \triangleq \langle . \, , . \rangle^{1/2}$ denotes the induced norm. The extension to the polynomial-matrix form is straightforward by replacing the inner product $\langle . \, , . \rangle$ with $\langle . \, , . \rangle_{\mathcal{F}}$. In case the search direction $\mathbf{S}_k$ is pointing in a ascent direction, μ_k will become negative, such that the resulting update $\mu_k \mathbf{S}_k$ will be in a decent direction. The inner product needs to be computed only once within an Armijo line search.

decrease, i.e., $\mathcal{J}\big(\mathbf{W}_k(z) + \mu_k\,\mathbf{S}_k(z)\big) \leq \mathcal{J}\big(\mathbf{W}_k(z)\big)$. In Appendix A the Armijo rule is described in vector, matrix, and polynomial matrix form. A routine to compute the Armijo step size for an arbitrary search direction is given in Fig. 2.

4 Efficient implementation in the frequency domain

The proposed algorithm, including the Armijo-line search method can be implemented efficiently in the frequency domain. To this end, we compute the linear (multichannel) convolutions via fast convolution techniques, where the linear convolutions are embedded in circular convolutions. The circular convolutions themselves are computed efficiently via FFT / IFFT and element-wise multiplications in the DFT domain. This strategy, if properly applied, does not change the original cost function nor does is modify the direction of the gradient, even though we transform some of the signals into the frequency domain. Hence the final algorithm will not suffer any permutation problem. The proposed algorithm is given in Fig. 7 and works also for complex source signals and complex filter coefficients. Furthermore, $\mathbf{W}(z)$ can be a non-square matrix ($M_{\mathrm{u}} \leq M_{\mathrm{x}}$). The notation of the vectors and the derivation are based on the same concepts as described in [2, Chapter 3 & Appendix F].

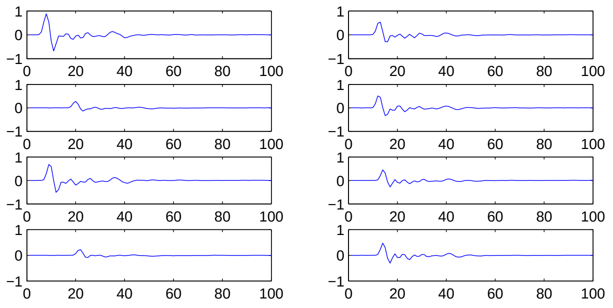

Figure 3: Impulse responses of the 4×2 mixing filter $\mathbf{A}(z)$

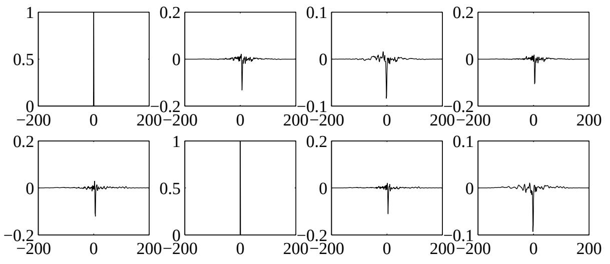

Figure 4: Impulse responses of the 2×4 demixing filter $\mathbf{W}(z)$

The proposed algorithm differs primarely from the algorithm in [3] by the additional line search incorporated into update stategy. Note that we can also perform the Armijo line search method in the frequency domain, under two conditions: (i) the search direction in the frequency domain is derived from the time-domain gradient, and (ii) the line search is carried out in all frequencies jointly (the same μ for all frequency bins). Generally speaking, the inner products $\langle . \, , . \rangle_{\mathcal{F}}$ can also be computed in the discrete Fourier domain, as long as the DFT length C is chosen large enough to avoid circular wrap around effects.

We also make use of another simplification. Since we choose the search direction $\mathbf{S}_k(z)$ according to (24), we have $\langle \mathbf{D}_{\mathbf{W}\,k}(z), \mathbf{S}_k(z) \rangle_{\mathcal{F}} = -\|\mathbf{S}_k(z)\|_{\mathcal{F}}^2$. Here we made use that the inner product between a diagonal matrix and an off-diagonal matrix is always zero, i.e., $\langle \mathrm{diag}(\,\mathbf{D}_{\mathbf{W}\,k}(z)\,), \mathbf{S}_k(z) \rangle_{\mathcal{F}} = 0$. The norm $\|\mathbf{S}_k(z)\|_{\mathcal{F}}^2$ can also be computed in the discrete Fourier domain, in the same way as $J[k]$ computes the norm of $\mathcal{J}(\mathbf{W}_k(z))$. Moreover, $\|\mathbf{S}_k(z)\|_{\mathcal{F}}$ can also serve to evaluate the termination condition of the Armijo line search, see Fig. 2.

The full implementation of the algorithm is not completely described in Fig. 7. Primarily the steps of the Armijo line search method in Fig. 2 are missing. However, the computational steps of how to compute $\mathcal{J}(\mathbf{W}_k(z) + \mu\mathbf{S}(z))$ are exactly the same as to compute $\mathcal{J}(\mathbf{W}_k(z))$. The only difference is that $\bar{\mathbf{w}}_{mn}[k]$ is replaced by $\bar{\mathbf{w}}_{mn}[k] + \mu \cdot \bar{\mathbf{s}}_{mn}[k]$ when computing $J[k]$. The IFFT / FFT operation carried out to obtain $\bar{\mathbf{s}}_{mn}[k]$ is required to prevent that the filter length of $\mathbf{W}(z)$ grows.

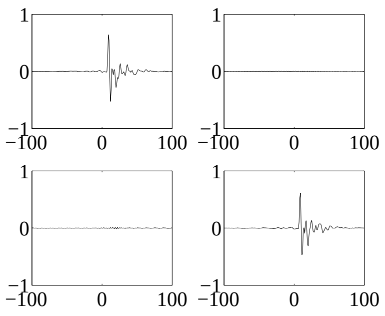

Figure 5: Impulse responses of the 2×2 global system $\mathbf{G}(z)$

5 Simulation example

To verify the performance of the proposed algorithm, we setup an artificial mixing system with $M_{\mathrm{s}} = 2$ source signals and $M_{\mathrm{x}} = 4$ sensors. The mixing system $\mathbf{A}(z)$ is extracted from real measured HRTFs (head-related transfer functions) and have length $N_{\mathrm{a}} = 100$. The correlation matrices $\mathbf{R}_{\mathbf{ss}}(z;t_p)$ of $P = 3$ snapshots are generated artificially to be diagonal matrices with diagonal elements $(\mathbf{R}_{\mathbf{ss}}(z;t_p))_{m,m} = b(z;t_p)\, b^{\dagger}(z;t_p)$ where $b(z;t_p)$ are randomly chosen filters of length 20. This setup simulates the case where the source signals are non-stationary *and* non-white. The input correlation matrices are computed as $\mathbf{R}_{\mathbf{xx}}(z;t_p) = \mathbf{A}(z)\mathbf{R}_{\mathbf{ss}}(z;t_p)\mathbf{A}^{\dagger}(z)$. This artificial generation of $\{\mathbf{R}_{\mathbf{ss}}(z;t_p)\}$ guarantees that the global minimum of $\mathcal{J}_1$ is, in fact, zero. The demixing system $\mathbf{W}(z)$ is a 2×4 matrix where each filter has length 199, ($N_{\mathrm{w}} = 99$). The impulse responses of $\mathbf{W}(z)$ and $\mathbf{G}(z)$ after convergence are shown in Fig. 4 and Fig. 5. From the vanishing off-diagonal impulse responses of $\mathbf{G}(z)$ it is clearly seen, that the proposed algorithm can perform almost perfect signal separation. Fig. 5 also indicates that the proposed algorithm does not suffer any permutation problem. The convergence behavior is shown in Fig. 6. It is clearly seen, that including the Armijo line search into the update can improve the convergance behavior considerably.

6 Conclusions

The presented algorithm is an extended version of the algorithm in [3]. In addition to the previous algorithm, we added an Armijo line search into the update equation. In contrast to Newton methods, which are very effective for quadratic cost functions or in the vicinity of a local minimum, the Armijo line search method helps also to improve the initial convergence. This statement is especially true for our case, as the optimization parameters appear up to the fourth power in the cost function. Moreover, it turns out that the new algorithm is also more robust against ill conditioned problems, and no additional normalization is required.

Line search methods are often used in optimization problems to improve the convergence behavior of a gradient based optimization algorithm. Incorporating line-search methods in the update strategy was previously suggested for the non-convolutive JAD problem. In [11] an Armijo line search method was used for the JAD problem with unitary matrices (Stiefel manifold) and in [12] an exact line search method was used for the JAD problem with non-unitary matrices. In order to apply an Armijo line search method to the convolutive JAD problem, we needed to extend the Armijo rule to the case of polynomial matrices.

Similarly to the previous algorithm, the cost function is formulated in the time domain, and also the derivation the gradient. By computing the linear convolutions via fast convolution techniques, most of the computation is carried out in the frequency domain. In doing so, we can elegantly avoid the so called permutation problem, which appears, if the cost function is formulated directly in the frequency domain on a bin-by-bin basis.

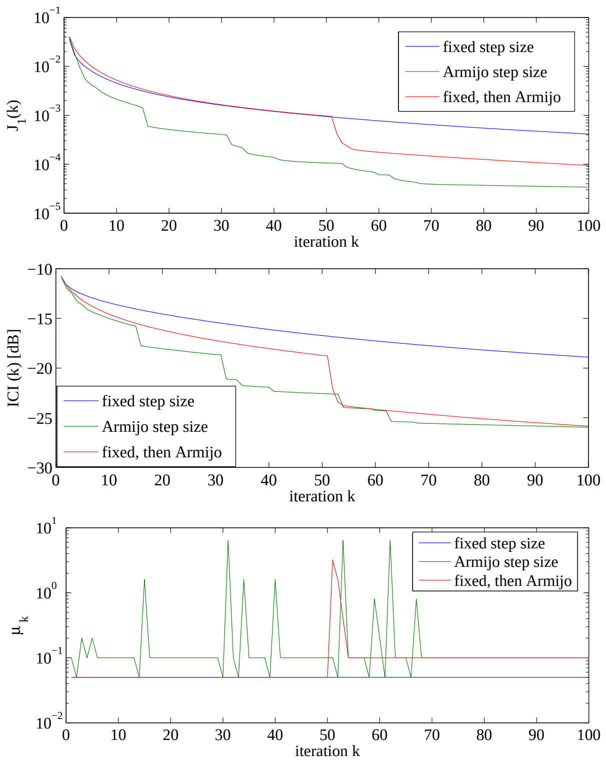

Figure 6: Comparison between (i) fixed step size, (ii) with Armijo line search, and (iii) fixed step size for the first 50 iterations, then with Armijo line search. The curves are for a typical single trial: (top) blind cost function $\mathcal{J}(\mathbf{W}(z)[k])$, (middle) interchannel interference ICI[k], and (bottom) step size μ_k.

References

[1] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Prentice-Hall, 1993.

[2] M. Joho, "A systematic approach to adaptive algorithms for multichannel system identification, inverse modeling, and blind identification," Ph.D. dissertation, ETH Zürich, Dec. 2000. http://e-collection.ethbib.ethz.ch/show?type=diss&nr=13783

[3] ——, "Blind signal separation of convolutive mixtures: A time-domain joint-diagonalization approach," in *Proc. ICA*, Granada, Spain, Sep. 22–24, 2004, pp. 577–584.

[4] E. Kreyszig, *Introductory Functional Analysis with Applications*. John Wiley & Sons, 1978.

[5] J.-F. Cardoso and A. Souloumiac, "Jacobi angles for simultaneous diagonalization," *SIAM J. Matrix Anal. and Appl.*, vol. 17, no. 1, pp. 161–164, Jan. 1996.

[6] J. H. Manton, “Optimisation algorithms exploiting unitary constraints,” *IEEE Trans. Signal Processing*, vol. 50, no. 3, pp. 635–650, Mar. 2002.

[7] L. Parra and C. Spence, “Convolutive blind separation of non-stationary sources,” *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 3, pp. 320–327, 2000.

[8] H. Buchner, R. Aichner, and W. Kellermann, “A generalization of blind source separation algorithms for convolutive mixtures based on second-order statistics,” *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 1, pp. 120–134, 2005.

[9] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[10] K. Matsuoka and S. Nakashima, “Minimal distortion principle for blind source separation,” in *Proc. ICA*, San Diego, CA, Dec. 9–12, 2001, pp. 927–932.

[11] M. Nikpour, J. H. Manton, and G. Hori, “Algorithms on the Stiefel manifold for joint diagonalisation,” in *Proc. ICASSP*, vol. 2, Orlando, FL, May 13–17, 2002, pp. 1481–1484.

[12] A. Yeredor, A. Ziehe, and K. R. Müller, “Approximate joint diagonalization using a natural-gradient approach,” in *Proc. ICA*, Granada, Spain, Sep. 22–24, 2004, pp. 89–96.

A Armijo Rule

A.1 Armijo rule in vector form

The Armijo rule is used in a line-search problem to find a reasonable step-size μ, see [9]. For fixed scalars $0 < \mu_0$, $0 < \eta < 1$, and $0 < \gamma < 1$, the Armijo rule chooses at the kth iteration the step size $\mu_k = \mu_0\,\gamma^m$ where m is the first nonnegative integer that fulfills

$$\mathcal{J}(\mathbf{w}_k) - \mathcal{J}(\mathbf{w}_k + \mu_0\,\gamma^m\,\mathbf{s}_k) \geq -\eta\,\mu_0\,\gamma^m\ \mathrm{Re}\{\langle \mathbf{d}_{\mathbf{w}k}, \mathbf{s}_k\rangle\} \tag{25}$$

The search direction and the gradient of $\mathcal{J}$ at $\mathbf{w}_k$ are denoted as $\mathbf{s}_k$ and $\mathbf{d}_{\mathbf{w}k}$, respectively, and $\langle \mathbf{d}_{\mathbf{w}k}, \mathbf{s}_k\rangle \triangleq \mathbf{s}_k^H \mathbf{d}_{\mathbf{w}k}$. Note that $\mathcal{J}(\mathbf{w}_k)$ is a real function, whereas the $\mathbf{w}_k$ can be complex.

A.2 Armijo rule in matrix form

Similarly to the vector form, Armijo rule chooses at the kth iteration the step size $\mu_k = \mu_0\,\gamma^m$ where m is the first nonnegative integer that fulfills

$$\mathcal{J}(\mathbf{W}_k) - \mathcal{J}(\mathbf{W}_k + \mu_0\,\gamma^m\,\mathbf{S}_k) \geq -\eta\,\mu_0\,\gamma^m\ \mathrm{Re}\{\langle \mathbf{D}_{\mathbf{W}k}, \mathbf{S}_k\rangle\} \tag{26}$$

The search direction and the gradient of $\mathcal{J}$ at $\mathbf{W}_k$ are denoted as $\mathbf{S}_k$ and $\mathbf{D}_{\mathbf{W}k}$, respectively, and $\langle \mathbf{D}_{\mathbf{W}k}, \mathbf{S}_k\rangle \triangleq \mathrm{tr}\{\mathbf{S}_k^H\,\mathbf{D}_{\mathbf{W}k}\}$.

A.3 Armijo rule in polynomial-matrix form

Similarly to the vector and matrix form, Armijo rule chooses at the kth iteration the step size $\mu_k = \mu_0\,\gamma^m$ where m is the first nonnegative integer that fulfills

$$\begin{aligned}\mathcal{J}\big(\mathbf{W}_k(z)\big) - \mathcal{J}\big(\mathbf{W}_k(z) + \mu_0\,\gamma^m\,\mathbf{S}_k(z)\big)\\ \geq -\eta\,\mu_0\,\gamma^m\ \mathrm{Re}\left\{\,\langle \mathbf{D}_{\mathbf{W}k}(z), \mathbf{S}_k(z)\rangle_{\mathcal{F}}\,\right\}\end{aligned} \tag{27}$$

The search direction and the gradient of $\mathcal{J}$ at $\mathbf{W}_k(z)$ are denoted as $\mathbf{S}_k(z)$ and $\mathbf{D}_{\mathbf{W}k}(z)$, respectively, and the inner product $\langle .,.\rangle_{\mathcal{F}}$ is defined in (7).

FCJD-ALS

Definitions:

$$\tilde{\mathbf{P}}_N \triangleq \begin{bmatrix} \mathbf{I}_{N+1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{C-2N-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_N \end{bmatrix}$$

$$\begin{aligned}\tilde{\mathbf{w}}_{mn}[k] \triangleq (&w_{mn,0}[k], \dots, w_{mn,N_\mathrm{w}}[k], 0, \dots,\\ &0, w_{mn,-N_\mathrm{w}}[k], \dots, w_{mn,-1}[k])^T\end{aligned}$$

Initialization ($\forall\, m, n, p$):

$$\tilde{\mathbf{w}}_{mn}[0] := \begin{cases} (1, 0, \dots, 0)^T & \text{for } m = n \\ (0, 0, \dots, 0)^T & \text{for } m \neq n \end{cases}$$

$$\bar{\mathbf{w}}_{mn}[0] := \mathrm{FFT}(\,\tilde{\mathbf{w}}_{mn}[0]\,)$$

Precompute spatiotemporal input correlations ($\forall\, m, n, p$):

$$r^{(p)}_{x_m x_n}(\tau) := E\{x_m(t_p)\,x_n^*(t_p - \tau)\} \quad \text{for } \tau \in \{-\tau_\mathrm{xx}, .., \tau_\mathrm{xx}\}$$

$$\begin{aligned}\tilde{\mathbf{r}}^{(p)}_{x_m x_n} := (&r^{(p)}_{x_m x_n}(0), \dots, r^{(p)}_{x_m x_n}(\tau_\mathrm{xx}), 0, \dots,\\ &0, r^{(p)}_{x_m x_n}(-\tau_\mathrm{xx}), \dots, r^{(p)}_{x_m x_n}(-1))^T\end{aligned}$$

$$\bar{\mathbf{r}}^{(p)}_{x_m x_n} := \mathrm{FFT}(\,\tilde{\mathbf{r}}^{(p)}_{x_m x_n}\,)$$

For each iteration k do ($\forall\, m, n, p$):

$$\bar{\mathbf{r}}^{(p)}_{u_m x_n}[k] := \sum\nolimits_{l=1}^{M_\mathrm{x}} \bar{\mathbf{w}}_{ml}[k] \odot \bar{\mathbf{r}}^{(p)}_{x_l x_n}$$

$$\bar{\mathbf{r}}^{(p)}_{u_m u_n}[k] := \sum\nolimits_{l=1}^{M_\mathrm{x}} \bar{\mathbf{w}}^*_{nl}[k] \odot \bar{\mathbf{r}}^{(p)}_{u_m x_l}[k]$$

$$\tilde{\mathbf{r}}^{(p)}_{u_m u_n}[k] := \mathrm{IFFT}(\,\bar{\mathbf{r}}^{(p)}_{u_m u_n}[k]\,)$$

$$\tilde{\mathbf{e}}^{(p)}_{mn}[k] := \begin{cases} \mathbf{0} & \text{for } m = n \\ \tilde{\mathbf{P}}_{\tau_\mathrm{e}}\,\tilde{\mathbf{r}}^{(p)}_{u_m u_n}[k] & \text{for } m \neq n \end{cases}$$

$$\bar{\mathbf{e}}^{(p)}_{mn}[k] := \mathrm{FFT}(\,\tilde{\mathbf{e}}^{(p)}_{mn}[k]\,)$$

$$J[k] := \sum\nolimits_{p=1}^{P} \sum\nolimits_{m=1}^{M_\mathrm{u}} \sum\nolimits_{n=1}^{M_\mathrm{u}} \left\|\bar{\mathbf{e}}^{(p)}_{mn}[k]\right\|_F^2$$

$$\vartriangle\bar{\mathbf{w}}_{mn}[k] := \sum\nolimits_{p=1}^{P} \sum\nolimits_{l=1}^{M_\mathrm{u}} \bar{\mathbf{e}}^{(p)}_{ml}[k] \odot \bar{\mathbf{r}}^{(p)}_{u_l x_n}[k]$$

compute search direction:

$$\bar{\mathbf{s}}_{mn}[k] := \begin{cases} \mathbf{0} & \text{for } m = n \\ -\,\mathrm{FFT}\Big(\tilde{\mathbf{P}}_{N_\mathrm{w}}\mathrm{IFFT}\,(\vartriangle\bar{\mathbf{w}}_{mn}[k])\Big) & m \neq n \end{cases}$$

$$\mu_k := \text{Armijo line search}\big(\{\bar{\mathbf{w}}_{mn}[k]\}, \{\bar{\mathbf{s}}_{mn}[k]\}\big)$$

$$\bar{\mathbf{w}}_{mn}[k+1] := \begin{cases} \bar{\mathbf{w}}_{mn}[k] & \text{for } m = n \\ \bar{\mathbf{w}}_{mn}[k] + \mu_k \cdot \bar{\mathbf{s}}_{mn}[k] & \text{for } m \neq n \end{cases}$$

Figure 7: FCJD-ALS: Fast convolutive joint diagonalization with Armijo line search. All vectors have length C, which is the FFT size. In order to prevent circular wrap-around effects affecting the updates, the FFT size needs to be chosen large enough. The notation and the concept behind the arrangement of the vector elements are taken from [2, Chapter 3 & Appendix F].