AN ADAPTIVE VERSION OF THE ALGEBRAIC CONSTANT MODULUS ALGORITHM

Alle-Jan van der Veen

Delft Univ. of Technology, Dept. Electrical Eng./DIMES, Mekelweg 4, 2628 CD Delft, The Netherlands

The Algebraic Constant Modulus Algorithm (ACMA) is a very effective block-algorithm for blind source separation. A particular feature is that it finds beamformers for all sources simultaneously. However, in cases with continuously transmitting sources and varying environments, data-adaptive algorithms may be more appropriate. In this paper, we derive an adaptive version of the ACMA, with a reasonable computational complexity. Simulations indicate that it is reliably tracking beamformers to all sources even in a highly time-varying scenario.

1. INTRODUCTION

Constant modulus algorithms (CMAs) are used to separate multiple coinciding constant modulus sources impinging on an antenna array. A challenge is to find all independent beamformers, one for each source. Most existing CMAs [2] find only a single source. In the context of data-adaptive algorithms, the CM Array [3, 4] attempts to find all sources recursively, by using a CMA to find a source and an LMS to subtract it from the data. However, convergence may be slow and subsequent signals tend to be less accurate due to misadjustment; it is also possible that the same signal is found twice. The most reliable algorithms to date appear to be OCMA [3, 5] and the related MUK [6]. In these algorithms, a prewhitening step ensures that the transformed beamformers are approximately orthogonal. Separately running CMAs are used to track each source, and an orthogonalization step after each update keeps the beamformers linearly independent. It is performing well in stationary situations but can have tracking problems (swapping of outputs) when channels are time-varying, e.g., due to fading.

The Algebraic Constant Modulus Algorithm (ACMA) [7] is a non-iterative block-algorithm (acting on a batch of data) and finds all independent beamformers algebraically, as the solution of a joint diagonalization problem. The algorithm is quite effective even on small blocks of data. The question studied in this paper is whether this algorithm can be modified to become data-adaptive, so that it can be used for tracking.

2. DATA MODEL AND ASSUMPTIONS

We assume a scenario with d constant modulus (CM) sources impinging on $M \geq d$ receive antennas. Under the narrow-band assumption, this leads to the model

$$\mathbf{x}_k = \mathbf{A}\mathbf{s}_k + \mathbf{n}_k \qquad (1)$$

where the vector $\mathbf{x}_k$ is a stacking of the M antenna outputs $(x_i)_k$ at discrete time k, $\mathbf{s}_k$ is a stacking of the d source signals $(s_i)_k$, and $\mathbf{A}$ is the $M\times d$ array response matrix. The M-dimensional vector $\mathbf{n}_k$ is additive white Gaussian noise. We may collect the data in matrices:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N], \qquad \mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N],$$

and likewise for a noise matrix $\mathbf{N}$, so that the model becomes

$$\mathbf{X} = \mathbf{A}\mathbf{S} + \mathbf{N}.$$

Under the assumption that $\mathbf{A}$ has full column rank, the objective is to find a collection of beamformers $\{\mathbf{w}_i\}$, one for each source, such that $\mathbf{w}_i^{\mathrm{H}}\mathbf{x}_k = (s_i)_k$ is equal to one of the original sources, and such that all sources are retrieved (obviously their ordering cannot be established). We collect the beamformers in a matrix

$$\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_d], \qquad M\times d.$$

To ensure that independent beamformers correspond to independent signals, we will consider a prewhitening operation which also reduces the dimension of $\mathbf{X}$ from M to d rows. An underscore is used to denote prefiltered variables. Thus, let $\underline{\mathbf{X}} := \mathbf{F}^{\mathrm{H}}\mathbf{X}$ where $\mathbf{F} : M\times d$ is the prefilter. Let $\hat{\mathbf{R}}_{\mathbf{x}} = \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{H}}$ have eigenvalue decomposition $\hat{\mathbf{R}}_{\mathbf{x}} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}^2\hat{\mathbf{U}}^{\mathrm{H}}$, and let $\hat{\mathbf{U}}_s\hat{\mathbf{\Sigma}}_s^2\hat{\mathbf{U}}_s^{\mathrm{H}}$ be a rank-d truncated approximation ($\hat{\mathbf{U}}$ has d columns). We choose $\mathbf{F} = \hat{\mathbf{U}}_s\hat{\mathbf{\Sigma}}_s^{-1/2}$; in that case $\hat{\underline{\mathbf{R}}}_{\mathbf{x}} := \frac{1}{N}\underline{\mathbf{X}}\underline{\mathbf{X}}^{\mathrm{H}} = \mathbf{I}$, and at the same time it reduces the dimension of $\mathbf{X}$ from M rows to d. After prefiltering, we have the model

$$\underline{\mathbf{X}} = \underline{\mathbf{A}}\mathbf{S} + \underline{\mathbf{N}}, \qquad \text{where} \quad \underline{\mathbf{A}} := \mathbf{F}^{\mathrm{H}}\mathbf{A},\ \underline{\mathbf{N}} := \mathbf{F}^{\mathrm{H}}\mathbf{N}.$$

This is essentially the same model as before, except $\underline{\mathbf{X}}$ has only d channels and $\underline{\mathbf{A}}:\ d\times d$ is square. The original beamforming problem is now replaced by finding a matrix $\mathbf{T} : d\times d$ with columns $\mathbf{t}_i$, acting on $\underline{\mathbf{X}}$. After $\mathbf{T}$ has been found, the beamforming matrix on the original data will be $\mathbf{W} = \mathbf{F}\mathbf{T}$.

3. CMA, OCMA AND MUK

The CMA(2,2) algorithm [2] is given by the update equation

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu\mathbf{x}_k\bar{z}_k, \qquad z_k = (|y_k|^2 - 1)y_k$$

where $y_k = \mathbf{w}_k^{\mathrm{H}}\mathbf{x}_k$ is the output of the beamformer using its current estimate, and μ is a small step size. The "Orthogonal CMA" (OCMA) [3, 5] premultiplies the update term by $\hat{\mathbf{R}}_{\mathbf{x}}^{-1}$ to make the algorithm independent of the scaling of the data:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu\hat{\mathbf{R}}_{\mathbf{x}}^{-1}\mathbf{x}_k\bar{z}_k, \qquad z_k = (|y_k|^2 - 1)y_k. \qquad (2)$$

We can directly interpret this algorithm in terms of the prewhitening step. Indeed, define $\mathbf{t} = \hat{\mathbf{R}}_{\mathbf{x}}^{1/2}\mathbf{w}$ and $\underline{\mathbf{x}} = \hat{\mathbf{R}}_{\mathbf{x}}^{-1/2}\mathbf{x}$. Premultiplying (2) with $\hat{\mathbf{R}}_{\mathbf{x}}^{1/2}$ leads to

$$\mathbf{t}_{k+1} = \mathbf{t}_k - \mu\underline{\mathbf{x}}_k\bar{z}_k, \qquad z_k = (|y_k|^2 - 1)y_k$$

where $y_k = \mathbf{w}_k^{\mathrm{H}}\mathbf{x}_k = \mathbf{t}_k^{\mathrm{H}}\underline{\mathbf{x}}_k$. Therefore, OCMA is equal to the ordinary CMA, but in the whitened domain. The algorithm is easily modified to update d beamformers in parallel:

$$\mathbf{T}_{k+1} = \mathbf{T}_k - \mu\underline{\mathbf{x}}_k\mathbf{z}_k^{\mathrm{H}}, \qquad \mathbf{z}_k = (\mathbf{y}_k \odot \bar{\mathbf{y}}_k - \mathbf{1}) \odot \mathbf{y}_k$$

where $\mathbf{y}_k = \mathbf{W}_k^{\mathrm{H}}\mathbf{x}_k = \mathbf{T}_k^{\mathrm{H}}\underline{\mathbf{x}}_k$ and $\odot$ denotes the Schur-Hadamard product (entrywise multiplication). In spite of its appearance, the beamformers are updated independently, and there is no guarantee that they converge to independent solutions. However, since $\mathbf{T}$ is supposed to be almost orthogonal and therefore well-conditioned, it is straightforward to recondition $\mathbf{T}$ after each update.

This work was supported in part by the Dutch Min. Econ. Affairs under TSIT 1025 "Beyond-3G", and by NWO-STW under the VICI programme (DTC.5893). An extended version of this paper will appear as a book chapter [1].

A simple technique for this is to compute an SVD of $\mathbf{T}$ as $\mathbf{T} = \sum \sigma_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{H}}$, and to replace the singular values of $\mathbf{T}$ that are below some threshold (e.g., smaller than 0.5) by 1:

$$\mathbf{T}' = \mathrm{recond}(\mathbf{T}) := \sum \sigma'_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{H}} \quad \text{where } \sigma'_j = \begin{cases} 1, & \sigma_j < 0.5 \\ \sigma_j, & \sigma_j \geq 0.5 \end{cases} \quad (3)$$

A similar algorithm was proposed more recently, called the Multi-User Kurtosis Algorithm (MUK) [6]. MUK is not specifically targeted for CM signals, but aims to separate statistically independent non-Gaussian signals by maximizing the absolute value of the kurtosis of the output. For sources with a negative kurtosis, this leads to [6]

$$\mathbf{T}_{k+1} = \mathbf{T}_k - \mu \underline{\mathbf{x}}_k \mathbf{z}_k^{\mathrm{H}}, \qquad \mathbf{z}_k = \mathbf{y}_k \odot \bar{\mathbf{y}}_k \odot \mathbf{y}_k .$$

In [6], a condition that $\mathbf{T}$ should be orthogonal (rather than well-conditioned) is maintained. This can be formulated as an orthogonal Procrustes problem [8] of which the solution is

$$\mathbf{T} =: \sum_j \sigma_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{H}} \qquad \Rightarrow \qquad \mathbf{T}' = \mathrm{reorth}(\mathbf{T}) := \sum_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{H}} . \quad (4)$$

The complexity of these algorithms is about $4dM + d^3 + 5d^2$ per update step, where $4dM$ is due to adaptive prewhitening, and d^3 is due to the reconditioning step.

4. THE ANALYTICAL CMA

The Analytical CMA (ACMA) [7] is based on the CMA(2,2) cost function formulated as a Least Squares problem:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_k (|\hat{s}_k|^2 - 1)^2 , \qquad \hat{s}_k = \mathbf{w}^{\mathrm{H}} \mathbf{x}_k . \quad (5)$$

Using Kronecker product properties, we can write $|\hat{s}_k|^2$ as

$$|\hat{s}_k|^2 = \mathbf{w}^{\mathrm{H}} (\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}) \mathbf{w} = (\bar{\mathbf{x}}_k \otimes \mathbf{x}_k)^{\mathrm{H}} (\bar{\mathbf{w}} \otimes \mathbf{w}) .$$

We subsequently stack the rows $(\bar{\mathbf{x}}_k \otimes \mathbf{x}_k)^{\mathrm{H}}$ of the data into a matrix $\mathbf{P}$ (size $N \times M^2$), so that $\mathbf{P} = [\bar{\mathbf{X}} \circ \mathbf{X}]^{\mathrm{H}}$ where $\circ$ denotes the Khatri-Rao product. Also introducing $\mathbf{y} = \bar{\mathbf{w}} \otimes \mathbf{w}$ and $\mathbf{1} = [1, \cdots, 1]^{\mathrm{T}}$, we can write (5) as

$$\frac{1}{N} \sum_k (|\hat{s}_k|^2 - 1)^2 = \frac{1}{N} \| \mathbf{P}\mathbf{y} - \mathbf{1} \|^2 . \quad (6)$$

It is shown in [9] that this can be conveniently rewritten in the whitened domain as

$$\mathbf{t} = \arg\min_{\substack{\mathbf{y} = \bar{\mathbf{t}} \otimes \mathbf{t} \\ \|\mathbf{y}\| = 1}} \mathbf{y}^{\mathrm{H}} \hat{\underline{\mathbf{C}}} \mathbf{y} \quad (7)$$

where $\mathbf{w} = \mathbf{F}\mathbf{t}$, $\mathbf{F}$ is the prewhitening filter as specified before, and

$$\hat{\underline{\mathbf{C}}} := \tfrac{1}{N} \underline{\mathbf{P}}^{\mathrm{H}} \underline{\mathbf{P}} - \tfrac{1}{N} \underline{\mathbf{P}}^{\mathrm{H}} \mathbf{1} \cdot \tfrac{1}{N} \mathbf{1}^{\mathrm{H}} \underline{\mathbf{P}} \quad (8)$$
$$= \tfrac{1}{N} \textstyle\sum_k (\bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k)(\bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k)^{\mathrm{H}} - \left[\frac{1}{N} \sum_k \bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k \right] \left[\frac{1}{N} \sum_k \bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k \right]^{\mathrm{H}} .$$

To solve this problem, the ACMA approach [7] is to first find an orthonormal basis $\mathbf{V}_n$ for the the d-dimensional approximate nullspace of $\hat{\underline{\mathbf{C}}}$, i.e., the set of d least dominant eigenvectors of $\hat{\underline{\mathbf{C}}}$. We subsequently look for a set of d unit-norm vectors $\bar{\mathbf{t}}_i \otimes \mathbf{t}_i$ that best spans the same subspace,

$$\mathbf{T} = \arg\min_{\substack{\|\mathbf{t}_i\| = 1 \\ \mathbf{M} \text{ invertible}}} \| \mathbf{V}_n \mathbf{M} - (\bar{\mathbf{T}} \circ \mathbf{T}) \|_{\mathrm{F}}^2 . \quad (9)$$

where $\underline{\mathbf{T}} = [\mathbf{t}_1, \cdots, \mathbf{t}_d]$ is the set of beamformers in the whitened domain, $\mathbf{T} \circ \mathbf{T} := [\mathbf{t}_1 \otimes \mathbf{t}_1, \cdots, \mathbf{t}_d \otimes \mathbf{t}_d]$, and $\mathbf{M}$ is a full rank $d \times d$ matrix that relates the two bases of the subspace. This problem can be recognized as a joint diagonalization problem, which is related to an eigenvalue problem and gives exact form results in the noise-free case.

Given a block of data $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$

1. Compute $\hat{\mathbf{R}}_{\mathbf{x}} = \frac{1}{N} \sum_k \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}$ $(M^2 N)$
   Compute the EVD $\hat{\mathbf{R}}_{\mathbf{x}} = \hat{\mathbf{U}} \hat{\mathbf{\Sigma}}^2 \hat{\mathbf{U}}^{\mathrm{H}}$ (M^3)
   Set the prewhitening filter $\mathbf{F} = \hat{\mathbf{U}}_s \hat{\mathbf{\Sigma}}_s^{-1/2}$ (dM)
2. Prefilter the data, $\underline{\mathbf{x}}_k = \mathbf{F}^{\mathrm{H}} \mathbf{x}_k$ (dMN)
   $\hat{\underline{\mathbf{C}}} = \frac{1}{N} \sum_k (\bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k)(\bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k)^{\mathrm{H}} - \mathrm{vec}(\hat{\underline{\mathbf{R}}}_{\mathbf{x}}) \mathrm{vec}(\hat{\underline{\mathbf{R}}}_{\mathbf{x}})^{\mathrm{H}}$ $(d^4 N)$
3. Compute $\mathbf{V}_n$, the nullspace of $\hat{\underline{\mathbf{C}}}$ $(\leq d^6)$
4. Initialize $\mathbf{T} = \mathbf{I}$
   Until convergence:
   $\mathbf{M} = \mathbf{V}_n^{\mathrm{H}} (\mathbf{T} \circ \mathbf{T})$ (d^4)
   $\mathbf{M}' = \mathrm{recond}(\mathbf{M})$, equation (3) (d^3)
   $\mathbf{Y} = \mathbf{V}_n \mathbf{M}'$ (d^4)
   $\mathbf{t}_i = \pi_1(\mathbf{y}_i), \quad i = 1, \cdots, d$, equation (10) (d^4)
5. Set $\mathbf{W} = \mathbf{F}\mathbf{T}$ and $\hat{\mathbf{s}}_k = \mathbf{W}^{\mathrm{H}} \mathbf{x}_k$ (dMN)

Figure 1. ACMA using iterative implementation of the joint diagonalization.

As inspired by [10], the problem (9) can be solved iteratively using Alternating Least Squares: (*i*) given an initial value for $\mathbf{T}$, solve for $\mathbf{M}$; (*ii*) for the new value of $\mathbf{M}$, solve for $\mathbf{T}$. Both steps are simple to implement: for fixed $\mathbf{T}$, solving for $\mathbf{M}$ gives

$$\mathbf{M} = \mathbf{V}_n^{\mathrm{H}} (\bar{\mathbf{T}} \circ \mathbf{T}) ,$$

whereas for fixed $\mathbf{M}$, solving for $\mathbf{T}$ gives

$$\mathbf{T} = \arg\min_{\|\mathbf{t}_i\| = 1} \| \mathbf{V}_n \mathbf{M} - (\bar{\mathbf{T}} \circ \mathbf{T}) \|^2 = \arg\min_{\|\mathbf{t}_i\| = 1} \| \mathbf{Y} - (\bar{\mathbf{T}} \circ \mathbf{T}) \|^2 ,$$

where $\mathbf{Y} = \mathbf{V}_n \mathbf{M} = \mathbf{V}_n \mathbf{V}_n^{\mathrm{H}} (\bar{\mathbf{T}} \circ \mathbf{T})$ is the projection of the previous solution onto the estimated subspace. The problem decouples into solving for the individual columns of $\mathbf{T}$:

$$\mathbf{t}_i = \arg\min_{\|\mathbf{t}_i\| = 1} \| \mathbf{y}_i - (\bar{\mathbf{t}}_i \otimes \mathbf{t}_i) \|^2 = \arg\min_{\|\mathbf{t}_i\| = 1} \| \mathbf{Y}_i - \mathbf{t}_i \mathbf{t}_i^{\mathrm{H}} \|^2 ,$$

where $\mathbf{Y}_i = \mathrm{vec}^{-1}(\mathbf{y}_i)$. The solution is given by an SVD of $\mathbf{Y}_i$ and retaining the dominant component, i.e.,

$$\mathbf{Y}_i =: \sum_j \sigma_j \mathbf{u}_j \mathbf{u}_j^{\mathrm{H}}, \qquad \mathbf{t}_i = \mathbf{u}_1 =: \pi_1(\mathbf{y}_i) . \quad (10)$$

We will denote this "projection onto rank-1" by $\mathbf{t}_i = \pi_1(\mathbf{y}_i)$.

The columns of $\mathbf{T}$ are processed independently. To avoid that they converge to the same solution, we can apply a reconditioning step on $\mathbf{M}$, since it is expected to be close to unitary after prewhitening. The resulting iterative ACMA is summarized in figure 1. Also the computational complexity of each step is indicated. Most of the work is done in two steps: the construction of $\hat{\mathbf{R}}_{\mathbf{x}}$ which has a complexity of $M^2 N$, and the construction of $\hat{\underline{\mathbf{C}}}$, which has a complexity of $d^4 N$.

5. ADAPTIVE ACMA

To make the preceding block-algorithm adaptive, the following ingredients are needed:

1. Adaptive implementation of the prewhitening filter $\mathbf{F}$,
2. Adaptive tracking of the nullspace of $\hat{\underline{\mathbf{C}}}$,
3. Adaptive update of the joint diagonalization, or alternatively, of the rank-1 mapping of each subspace vector.

A suitable adaptive prewhitening algorithm appeared in [11]. This algorithm combines an adaptive Cholesky factorization and inversion with a subspace tracking algorithm based on PAST [12]. Its complexity is order $4dM + 3d^2$ per update step.

5.1. Adaptive tracking of $\hat{\underline{\mathbf{C}}}$

We first derive an update equation for $\hat{\underline{\mathbf{C}}}$. In the final algorithm we will avoid to construct and store $\hat{\underline{\mathbf{C}}}$, but will directly update its nullspace using the update vectors of $\hat{\underline{\mathbf{C}}}$.

So far, $\hat{\underline{\mathbf{C}}}$ was defined only in terms of a given batch of N samples, but we would like to convert this into an exponential window (λ-scaling). As before, let $\mathbf{P}$ be an $N \times d^2$ dimensional matrix with rows $\mathbf{y}_k^{\mathrm{H}} := (\bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k)^{\mathrm{H}}$, where $\underline{\mathbf{x}}_k$ is the received (whitened) sample at time k. For simplicity of notation, we will drop the underscores in this subsection from now on, since all data will be in the whitened domain. According to equation (8)

$$\hat{\mathbf{C}} := \frac{1}{N}\mathbf{P}^{\mathrm{H}}\mathbf{P} - \frac{1}{N^2}\mathbf{P}^{\mathrm{H}}\mathbf{1}\mathbf{1}^{\mathrm{H}}\mathbf{P}.$$

Comparing this matrix to

$$\mathbf{H} := \sum_{k=1}^{N} \begin{bmatrix} 1 \\ \mathbf{y}_k \end{bmatrix} [1 \quad \mathbf{y}_k^{\mathrm{H}}] = [\mathbf{1} \quad \mathbf{P}]^{\mathrm{H}} [\mathbf{1} \quad \mathbf{P}] = \begin{bmatrix} \mathbf{1}^{\mathrm{H}}\mathbf{1} & \mathbf{1}^{\mathrm{H}}\mathbf{P} \\ \mathbf{P}^{\mathrm{H}}\mathbf{1} & \mathbf{P}^{\mathrm{H}}\mathbf{P} \end{bmatrix} \qquad (11)$$

we see that $N\hat{\mathbf{C}}$ is equal to the Schur complement $\mathbf{H}_{2,2} - \mathbf{H}_{2,1}\mathbf{H}_{1,1}^{-1}\mathbf{H}_{1,2}$.

An adaptive update rule for $\hat{\mathbf{C}}$ can be derived from an adaptive update rule for $\mathbf{H}$, where we scale previous estimates with λ, with $0 < \lambda < 1$. Thus let $\mathbf{H}_k$ and $\mathbf{C}_k$ be defined as

$$\mathbf{H}_k := (1-\lambda) \sum_{i=1}^{k} \lambda^{k-i} \begin{bmatrix} 1 \\ \mathbf{y}_i \end{bmatrix} [1 \quad \mathbf{y}_i^{\mathrm{H}}] =: \begin{bmatrix} \alpha_k & \mathbf{p}_k^{\mathrm{H}} \\ \mathbf{p}_k & \mathbf{N}_k \end{bmatrix}, \quad \mathbf{C}_k := \mathbf{N}_k - \frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{H}}}{\alpha_k}$$

then $\mathbf{H}_k$ is an unbiased estimate of $\mathbf{H}$, due to multiplication with the factor $(1-\lambda)$, and $\mathbf{C}_k$ is an unbiased estimate of $\hat{\mathbf{C}}$. The update rule for $\mathbf{C}_k$ which follows from this equation can be shown to be

$$\begin{aligned} \mathbf{C}_k &= \lambda \mathbf{C}_{k-1} + \tfrac{\alpha_{k-1}}{\alpha_k} \lambda(1-\lambda) \cdot (\mathbf{y}_k - \mathbf{p}_{k-1}/\alpha_{k-1})(\mathbf{y}_k - \mathbf{p}_{k-1}/\alpha_{k-1})^{\mathrm{H}} \\ &=: \lambda \mathbf{C}_{k-1} + \beta_k \mathbf{c}_k \mathbf{c}_k^{\mathrm{H}}. \end{aligned} \qquad (12)$$

where $\mathbf{p}_{k-1}$ and α_{k-1} are updated as

$$\mathbf{p}_k = \lambda \mathbf{p}_{k-1} + (1-\lambda)\bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k, \qquad \alpha_k = \lambda\alpha_{k-1} + (1-\lambda). \qquad (13)$$

Therefore, the vector by which to update $\mathbf{C}_{k-1}$ is equal to a scaling of the modified data vector

$$\mathbf{c}_k := \bar{\underline{\mathbf{x}}}_k \otimes \underline{\mathbf{x}}_k - \mathbf{p}_{k-1}/\alpha_{k-1}. \qquad (14)$$

5.2. Adaptive tracking of the nullspace of $\hat{\underline{\mathbf{C}}}$

We have to track the nullspace of $\hat{\underline{\mathbf{C}}}$, a d-dimensional nullspace in a d^2-dimensional space. Currently the most promising and computationally efficient algorithm for this appears to be the "Normalized Orthogonal OJA" (NOOJA) [13]. It is of complexity $4dM$, and scaling-independent: if the input data $\mathbf{c}_k$ is multiplied by a scalar, then the resulting subspace estimate $\mathbf{V}_k$ is unchanged.

5.3. Adaptive update of the joint diagonalization

Using the preceding nullspace tracking algorithm in our application, we can update the basis $\mathbf{V}_n$ of the nullspace of $\hat{\underline{\mathbf{C}}}$ given the update vector $\mathbf{c}_k$ in (14). The final step is an efficient implementation of the joint diagonalization. After the subspace update, the iterative ACMA (figure 1) uses the previous estimate of the beamformers, $\mathbf{T}$, and continues with the following three steps:

$$\begin{aligned} \mathbf{M} &= \mathbf{V}_n^{\mathrm{H}}(\bar{\mathbf{T}} \circ \mathbf{T}) \\ \mathbf{Y} &= \mathbf{V}_n \mathbf{M} = \mathbf{V}_n \mathbf{V}_n^{\mathrm{H}}(\bar{\mathbf{T}} \circ \mathbf{T}) \\ \mathbf{t}_i &= \pi_1(\mathbf{y}_i), \quad i = 1, \cdots, d. \end{aligned}$$

Given data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots]$, compute $\hat{\mathbf{s}}_k = \mathbf{W}_k^{\mathrm{H}}\mathbf{x}_k$:

Initialize prewhitening filter $\mathbf{F}$; $\mathbf{T} = \mathbf{I}_{d\times d}$, $\mathbf{p} = \mathbf{0}, \alpha = 0$
for $k = 1, 2, \cdots$ do

1. Update $\mathbf{F}$ using $\mathbf{x}_k$ [11] $(4dM + 3d^2)$
   $\underline{\mathbf{x}} = \mathbf{F}^{\mathrm{H}}\mathbf{x}_k$, the prewhitened input vector
2. Compute the update vector $\mathbf{c}$ for $\hat{\underline{\mathbf{C}}}$: (d^2)
   $\mathbf{c} = \bar{\underline{\mathbf{x}}} \otimes \underline{\mathbf{x}} - \mathbf{p}/\alpha$
   $\mathbf{p} = \lambda\mathbf{p} + (1-\lambda)\bar{\underline{\mathbf{x}}} \otimes \underline{\mathbf{x}}$
   $\alpha = \lambda\alpha + (1-\lambda)$
   Compute $\mathbf{Y} = \bar{\mathbf{T}} \circ \mathbf{T}$ (d^3)
   Regard $\mathbf{Y}$ as a basis of the nullspace of $\hat{\underline{\mathbf{C}}}$, and update it using $\mathbf{c}$ [13] $(4d^3)$
3. for $i = 1, \cdots, d$ do (d^3)
   $\mathbf{Y}_i = \mathrm{vec}^{-1}(\mathbf{y}_i)$
   $\mathbf{t}_i = \mathbf{Y}_i\mathbf{t}_i$ (one step of a power iteration)
   $\mathbf{t}_i = \mathbf{t}_i/\|\mathbf{t}_i\|$
   end
4. $\mathbf{T} = \mathrm{recond}(\mathbf{T})$, equation (3) (d^3)
5. $\hat{\mathbf{s}}_k = \mathbf{T}^{\mathrm{H}}\underline{\mathbf{x}}$ (d^2)

$4dM + 6d^3$

Figure 2. Adaptive implementation of ACMA.

This projects $\bar{\mathbf{T}} \circ \mathbf{T}$ onto the estimated subspace, resulting in $\mathbf{Y}$, and subsequently maps the columns of $\mathbf{Y}$ back to the Kronecker-product structure, resulting in new estimates of the columns $\mathbf{t}_i$ of $\mathbf{T}$. However, the complexity of the projection is too high (order d^4 instead of d^3).

Therefore, the following modification is introduced: instead of updating the basis $\mathbf{V}_n$, we compute $\bar{\mathbf{T}} \circ \mathbf{T}$ and regard it as the current estimate of the subspace basis (i.e., set $\mathbf{V}_n = \bar{\mathbf{T}} \circ \mathbf{T}$). Using this basis, the subspace update is performed, giving $\mathbf{Y}$, and then the result is mapped back to the Kronecker-product structure. In this context, the update performed by the NOOJA algorithm is interpreted as a Householder reflection which tries to make $\bar{\mathbf{T}} \circ \mathbf{T}$ orthogonal to the current update vector $\mathbf{c}_k$.

The last step of the algorithm is the mapping of the columns $\mathbf{y}_i$ of $\mathbf{Y}$ to a Kronecker-product structure, $\mathbf{y}_i =: \mathbf{t}_i \otimes \mathbf{t}_i$, or equivalently,

$$\mathbf{Y}_i = \mathbf{t}_i \mathbf{t}_i^{\mathrm{H}}$$

where $\mathrm{vec}(\mathbf{Y}_i) = \mathbf{y}_i$. Instead of computing an SVD (complexity d^3 for $i = 1, \cdots, d$), we can apply a power iteration [8] for this, which takes the general form

$$\mathbf{v}_{k+1} = \mathbf{Y}\mathbf{v}_k, \qquad k = 1, 2, \cdots$$

The best choice of an initial point is the previous estimate for $\mathbf{t}_i$, and in this case, a single step of the iteration is sufficient to give a good improvement of the estimate. The complexity of one update step is d^2 per subspace vector, or d^3 in total.

The resulting algorithm is summarized in figure 2. As indicated, the complexity of the algorithm is of order $4dM + 6d^3$. This is comparable to the complexity of OCMA and MUK.

6. COMPARISON OF MUK WITH ADAPTIVE-ACMA

We consider a uniform linear array with $M = 4$ antennas, $d = 3$ constant-modulus sources with directions $[-10^\circ, 20^\circ, 30^\circ]$ and amplitudes $[1,\ .8,\ .9]$. To test the tracking behavior of the adaptive

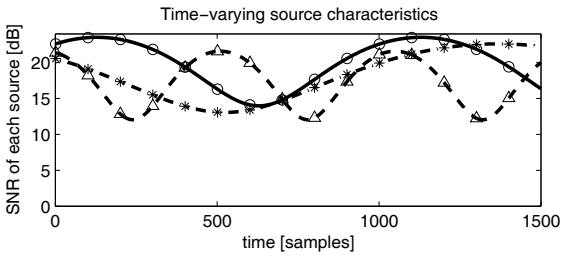

Figure 3. SNR of a time-varying channel (example).

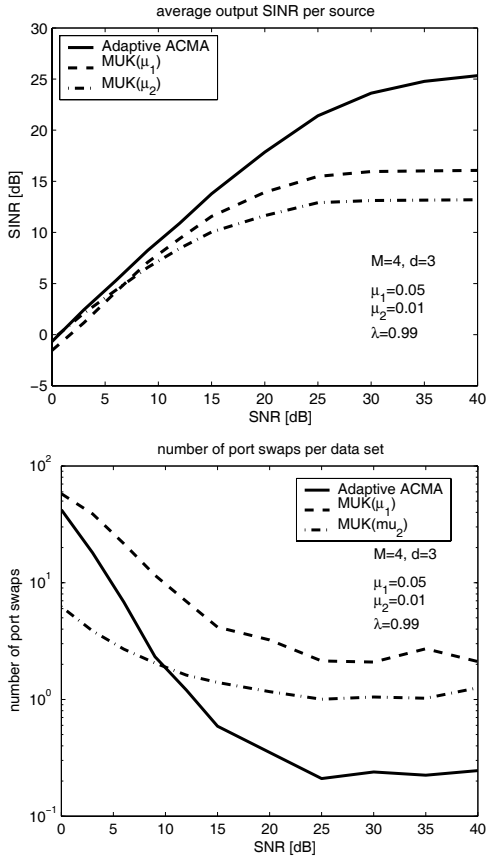

Figure 4. Performance of MUK and adaptive-ACMA in a time-varying scenario: (a) average output SINR, (b) average number of port swaps per interval (1500 samples).

algorithms, this scenario is made time-varying. Specifically, the source amplitudes are varied in sinusoidal patterns with randomly selected periods, with a maximum of 3 periods over the simulated time interval ($N = 1500$ samples). An example is shown in figure 3. Moreover, the direction vectors $\{\mathbf{a}_i\}$ of each source are not selected on an array manifold, instead each entry of each $\mathbf{a}_i$ is a unimodular complex number with an arbitrarily selected linear phase progression, with at most one cycle per interval.

Figure 4(a) shows the average output SINR of each source for MUK and Adaptive ACMA as a function of SNR, where the average is over time, over the three sources, and over 300 monte carlo runs, each with different randomly varying channels (there has been no attempt to detect and remove "failed cases" where not all independent sources are found). Since proper selection of the step size μ in MUK is a difficult and sensitive issue, the results for two different values are shown. Figure 4(b) shows the average number of times that a port swap occurred in a data run of N samples (i.e., cases where a beamformer suddenly starts to track a different source). For MUK, a larger μ gives faster tracking and better output SINR, but also more port swaps: usually at least once per data set. Both algorithms are performance limited at high SNR due to the time-variation in the observation window, but overall, Adaptive-ACMA is better in this particular scenario.

Since this is a new algorithm, it is unclear whether this is observation holds in general, although other simulations indicate that for higher SNRs, Adaptive-ACMA is always several dB better and is more reliable in recovering all weight vectors. Another experience is that the performance of Adaptive-ACMA is limited by the speed and accuracy of the nullspace tracking algorithm. NOOJA is not completely satisfactory since it tends to be jittery in steady state, and further improvements may be obtained by replacing NOOJA by a more accurate (and more complex) nullspace tracker.

References

[1] A.J. van der Veen and A. Leshem, "Constant modulus beamforming," in *Robust Beamforming* (P. Stoica e.a., ed.), ch. 8, Wiley Interscience, 2005.

[2] J.R. Treichler and B.G. Agee, "A new approach to multipath correction of constant modulus signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 31, pp. 459–471, Apr. 1983.

[3] R. Gooch and J. Lundell, "The CM array: An adaptive beamformer for constant modulus signals," in *Proc. IEEE ICASSP*, (Tokyo), pp. 2523–2526, 1986.

[4] A.V. Keerthi, A. Mathur, and J.J. Shynk, "Misadjustment and tracking analysis of the constant modulus array," *IEEE Trans. Signal Processing*, vol. 46, pp. 51–58, Jan. 1998.

[5] R. Pickholtz and K. Elbarbary, "The recursive constant modulus algorithm: A new approach for real-time array processing," in *27-th Asilomar Conf. Signals, Syst. Comp.*, pp. 627–632, IEEE, 1993.

[6] C. Papadias, "Globally convergent blind source separation based on a multiuser kurtosis maximization criterion," *IEEE Trans. Signal Processing*, vol. 48, pp. 3508–3519, Dec. 2000.

[7] A.J. van der Veen and A. Paulraj, "An analytical constant modulus algorithm," *IEEE Trans. Signal Processing*, vol. 44, pp. 1136–1155, May 1996.

[8] G. Golub and C.F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, 1989.

[9] A.J. van der Veen, "Asymptotic properties of the algebraic constant modulus algorithm," *IEEE Trans. Signal Processing*, vol. 49, pp. 1796–1807, Aug. 2001.

[10] N.D. Sidiropoulos, G.B. Giannakis, and R. Bro, "Parallel factor analysis in sensor array processing," *IEEE Trans. Signal Processing*, vol. 48, pp. 2377–2388, Aug. 2000.

[11] S.C. Douglas, "Combined subspace tracking, prewhitening, and contrast optimization for noisy blind signal separation," in *Proc. 2nd int. workshop Indept. Component Anal. Source Sep.*, (Helsinki, Finland)), pp. 579–584, June 2000.

[12] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Proc.*, vol. 43, pp. 95–107, Jan. 1995.

[13] S. Attallah and K. Abed-Meraim, "Fast algorithms for subspace tracking," *IEEE Signal Processing Letters*, vol. 8, pp. 203–206, July 2001.