Adaptive Re-Weighting Homotopy Algorithms Applied to Beamforming

Fernando G. Almeida Neto, *Student Member, IEEE*, Rodrigo C. de Lamare, *Senior Member, IEEE*, Vítor H. Nascimento, *Senior Member, IEEE*, Yuriy V. Zakharov, *Senior Member, IEEE*

Abstract

We develop adaptive beamforming algorithms which are robust against sensor failure and ill- conditioning in the autocorrelation matrix (common in low-rank interference scenarios). Both goals are achieved simultaneously through the use of ℓ_1 regularization. The algorithms are based on the complex adaptive re-weighting homotopy technique. We also develop iterative versions of the algorithms, that take advantage of properties of homotopy ℓ_1 solvers and dichotomous coordinate iterations to reduce considerably the computational complexity, compared with other regularization methods.

Index Terms

Adaptive re-weighting homotopy, adaptive beamforming, dichotomous coordinate descent

I. Introduction

Adaptive beamforming techniques are used in sensor arrays to enhance the reception of a signal of interest and suppress interference [1]. They implement techniques such as the minimum variance distortionless response (MVDR) beamformer [1] using data collected from sensors, since the second-order statistics required to compute the MVDR beamformer, in general, are not available.

Fernando G. Almeida Neto and Vítor H. Nascimento are with the Department of Electronic Systems, Escola Politécnica, University of São Paulo, São Paulo, SP, Brazil. e-mails: {fganeto, vitor}@lps.usp.br.

Rodrigo C. de Lamare is with the Centre for Telecommunications Research (CETUC), Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, RJ, Brazil. e-mail:delamare@cetuc.puc-rio.br

Yuriy V. Zakharov is with the Department of Electronics, University of York, York, UK. e-mail: yury.zakharov@york.ac.uk

This work is partly funded by CNPq and Fapesp-York grants.

Although many fields apply adaptive beamforming techniques, such as radar, sonar and wireless communications [1], it is challenging to implement traditional approaches in large arrays. Techniques such as the least mean-square (LMS), conjugate gradient (CG) and recursive least-squares (RLS) algorithms [1]–[3] have their convergence and tracking performances affected by the size and/or the eigenvalue spread of the input correlation matrix [2]. This performance can also degrade due to mismatch and modelling errors. Therefore, beamformers with many parameters may require many snapshots to converge, which can be incompatible with the requirements of some applications (for instance, space-time adaptive processing for airborne radar [4]–[7], where the amount of data involved requires a high computational cost to compute the beamformer).

As an alternative to the traditional methods, robust adaptive beamformers were proposed to reduce the performance degradation caused by steering vector uncertainties and also enhance interference cancellation. Recent advances also include techniques such as [8], [9] and [10], which use a distributed approach to compute the beamformer. Techniques such as adding a diagonal loading to the correlation matrix [11], [12], the robust Capon beamformer of [13], [14] (RCB), which uses the eigenvalue decomposition of the correlation matrix to compute the beamformer, and techniques based on worst-case performance optimization [14], [15] are some examples of robust beamformers. Many others can be found in the literature, (see [14] and references therein, for instance). However, most of these techniques are costly to compute (for instance, the beamformer of [13] has cubic computational complexity in the number of sensors in the array), which make them difficult to implement.

In this paper, we consider arrays for which the number of signal sources is much less than the number of sensors, such that the correlation matrix can become ill-conditioned when the ratio of the source to the noise power is high. We propose ℓ_1-norm regularized algorithms to regularize the matrix, and we show that this approach enhances the SINR performance at a low computational cost. The use of ℓ_1 regularization has the additional advantage of making the algorithm robust against sensor failure. For this purpose, we employ a modified version of the homotopy algorithm [16], which is an ℓ_1-norm regularized technique used in many applications, such as recovery of sparse signals from noisy measurements [17] and channel estimation [18]. Homotopy is generally applied to solve sparse systems of equations, and it helps the selection of the minimum amount of regularization required to compute the solution, reducing the bias. In the approach used in this paper, the homotopy strategy is not applied to sparse systems of equations, but to regularize the correlation matrix in a low-cost way. Our algorithms are extensions of the adaptive re-weighting homotopy (ARH) of [17] to the complex domain (C-ARH). We also develop new low-cost iterative versions of the C-ARH algorithms suitable for adaptive beamforming, and show that the proposed methods are able to compute the beamformer with a quadratic cost in the number of sensors.

We note that preliminary results on the C-ARH algorithms were reported in [19]. In this work, we describe in further detail the C-ARH and the multi-candidate (MC)-C-ARH algorithms, and introduce modifications to develop iterative versions of these methods. We show that the iterative approach is doubly advantageous, since it improves the signal-to-noise plus interference ratio (SINR) performance and also reduces the computational complexity.

Our contributions are summarized as follows.

1) The C-ARH and multi-candidate C-ARH algorithms, proposed in [19], are presented here in further detail.
2) We devise the iterative C-ARH (It-C-ARH) and the iterative MC-C-ARH (It-MC-C-ARH) algorithms to further improve the SINR performance with a reduced computational cost. We show that the iterative approaches outperform their non-iterative counterparts, improving the steady-state SINR.
3) We show how the dichotomous coordinate descent technique (DCD) [20] can be used to further reduce the computational complexity. The DCD is used to solve the systems of equations which appear in the C-ARH algorithm, in order to reduce the computational cost and to develop algorithms suitable for hardware implementations.
4) An analysis of properties of the proposed algorithms is presented along with an assessment of their computational complexity.
5) We present a simulation study comparing the proposed algorithms to existing robust techniques. We show that the iterative algorithms using the DCD present a small SINR performance degradation when compared to the RCB of [13], but the iterative algorithms require less computations, and are also robust against sensor failure, a property that RCB does not have.

This paper is organized as follows: Section II presents the system model and the problems considered in this paper. In Section III, we present the C-ARH and MC-C-ARH algorithms. In Section IV, we propose the iterative versions of C-ARH and MC-C-ARH, while in Section V we use the DCD algorithm to obtain low-cost algorithms. Section VI presents the analyses of the algorithms, and Section VII shows simulation results. We conclude the paper in Section VIII.

Notation: Lower case is used for scalar quantities (e.g.: a) and bold lower case for column vectors (e.g.: $\mathbf{b}$). Bold capital letters represent matrices (e.g.: $\mathbf{A}$). $\mathbf{a}_k$ stands for the k-th column of $\mathbf{A}$, while a_{kl} denotes the entry of $\mathbf{A}$ in the k-th row and in the l-th column. For a vector $\mathbf{b}$, we denote b_k as its k-th element. $(\cdot)^T$ stands for transposition, while $(\cdot)^H$ is the Hermitian of a matrix or vector. The operations $\mathcal{I}m\{\cdot\}$ and $\mathcal{R}e\{\cdot\}$ take only the imaginary and real parts of a complex number, and diag$(\cdot)$ defines a diagonal matrix. $||\cdot||_p$ is the ℓ_p-norm, and $E\{\cdot\}$ is the expectation operator. $\mathbf{I}_K$ represents a $K \times K$ identity matrix, and $\mathbf{0}_{K\times M}$ represents a $K \times M$ matrix of zeros.

II. System Model and Problem Statement

Consider a uniform linear array (ULA) with N sensors, and assume S signals, where one arrives from the desired direction of arrival θ_d, and the other $S-1$ signals are interferers. Additionally, assume that the signal of interest and the interferers are uncorrelated and that θ_d is known. Define the $N \times S$ matrix $\mathbf{B}$, where each column $\mathbf{b}_k$ corresponds to a steering vector [1] as given by

$$\mathbf{b}_k = [1 \;\; e^{-j\pi \sin(\theta_k)} \;\; \ldots \;\; e^{-j\pi(N-1)\sin(\theta_k)}]^T, \quad 1 \leq k \leq S.$$

At snapshot n, the sensor array data are modeled as

$$\mathbf{u}(n) = \mathbf{B}\,\mathbf{s}(n) + \boldsymbol{\eta}(n), \tag{1}$$

where $\mathbf{s}(n)$ contains signals produced by the S sources. $\boldsymbol{\eta}(n)$ is a vector of zero-mean, independent and identically distributed (i.i.d.) Gaussian noise with variance σ_η^2. The noise in each sensor is also assumed independent from the noise in other array elements. Without loss of generality, define $\theta_\text{d} = \theta_1$ and $\mathbf{b}_\text{d} = \mathbf{b}_1$. The coefficients of the MVDR beamformer [1] are given by

$$\mathbf{h}_\text{MVDR} = \mathbf{x}_\text{MVDR} / \mathbf{b}_\text{d}^H \mathbf{x}_\text{MVDR}, \tag{2}$$

and $\mathbf{x}_\text{MVDR}$ is the solution to

$$\mathbf{R}_\text{t} \mathbf{x}_\text{MVDR} = \mathbf{b}_\text{d}. \tag{3}$$

$\mathbf{R}_\text{t}$ is the theoretical $N \times N$ correlation matrix [1], which is defined as

$$\mathbf{R}_\text{t} = E\{\mathbf{u}(n)\mathbf{u}^H(n)\} = \mathbf{R}_\text{dI} + \mathbf{R}_\eta, \tag{4}$$

where

$$\mathbf{R}_\text{dI} = \mathbf{B} E\{\mathbf{s}(n)\mathbf{s}^H(n)\}\mathbf{B}^H \tag{5}$$

and

$$\mathbf{R}_\eta = \sigma_\eta^2 \mathbf{I}_N. \tag{6}$$

Express (1) explicitly in terms of the direction of interest (subscript d) and the interference (subscript I), i.e,

$$\mathbf{u}(n) = \mathbf{b}_\text{d}\, s_\text{d}(n) + \mathbf{B}_\text{I}\, \mathbf{s}_\text{I}(n) + \boldsymbol{\eta}(n), \tag{7}$$

to write $\mathbf{R}_{\text{dI}}$ as

$$\mathbf{R}_{\text{dI}} = \mathbf{R}_{\text{d}} + \mathbf{R}_{\text{I}}, \tag{8}$$

where $\mathbf{R}_{\text{d}} = \sigma_{\text{d}}^2 \mathbf{b}_{\text{d}} \mathbf{b}_{\text{d}}^H$, $\mathbf{R}_{\text{I}} = \mathbf{B}_{\text{I}} E\{\mathbf{s}_{\text{I}}(n)\mathbf{s}_{\text{I}}^H(n)\}\mathbf{B}_{\text{I}}^H$ and σ_{d}^2 is the variance of the signal of interest.

Note that the computation of the beamformer requires the solution of a linear system of equations. When the number of sources is less than the number of sensors, $\mathbf{R}_{\text{t}}$ can become ill-conditioned, requiring some form of regularization to compute $\mathbf{x}_{\text{MVDR}}$. In addition, if the measurements of some sensors are not available (if a sensor fails), a rank-reduction of the system of equations can be made before the introduction of the regularization, reducing the computations to obtain the solution. We consider both situations and show that using the ℓ_1-norm regularized algorithms presented in this paper, one can estimate the beamformer and improve the SINR performance with only $O(N^2)$ computations, while techniques such as [13] require $O(N^3)$ computations. The proposed ℓ_1 algorithms are also shown to be robust against errors in estimating faulty sensors.

A. Small number of interference sources

Consider that the number of interferers S is smaller than the number of sensors N. Assume that the interference sources are uncorrelated among each other, such that $\text{rank}(E\{\mathbf{s}\mathbf{s}^H\}) = S$, and assume that the angles θ_k are selected such that $\text{rank}(\mathbf{B}) = S$. Using properties of the rank of matrices (see [21]) one can show that[1]

$$\text{rank}(\mathbf{R}_{\text{dI}}) = \text{rank}(\mathbf{B}E\{\mathbf{s}\mathbf{s}^H\}\mathbf{B}^H) = S. \tag{9}$$

Since $\text{rank}(\mathbf{R}_{\text{dI}}) = S < N$, the eigenvalue decomposition of $\mathbf{R}_{\text{dI}}$ is given by

$$\mathbf{R}_{\text{dI}} = \mathbf{V} \begin{bmatrix} \mathbf{D}_0 & \mathbf{0}_{S\times(N-S)} \\ \mathbf{0}_{(N-S)\times S} & \mathbf{0}_{(N-S)\times(N-S)} \end{bmatrix} \mathbf{V}^H, \tag{10}$$

where the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{R}_{\text{dI}}$, and $\mathbf{D}_0$ is a diagonal matrix containing the S non-zero eigenvalues of $\mathbf{R}_{\text{dI}}$. Recalling that $\mathbf{V}\mathbf{V}^H = \mathbf{I}_N$, and using (6) and (10) in (4), we obtain

$$\mathbf{R}_{\text{t}} = \mathbf{V} \begin{bmatrix} \mathbf{D}_0 + \sigma_\eta^2 \boldsymbol{I}_S & \mathbf{0}_{S\times(N-S)} \\ \mathbf{0}_{(N-S)\times S} & \sigma_\eta^2 \boldsymbol{I}_{(N-S)} \end{bmatrix} \mathbf{V}^H. \tag{11}$$

Using eq. (11), it is easy to see that the condition number [21] of $\mathbf{R}_{\text{t}}$ is given by

$$\kappa(\mathbf{R}_{\text{t}}) = (d_{0_{\text{MAX}}} + \sigma_\eta^2)/\sigma_\eta^2, \tag{12}$$

[1]Note that we drop here time indices to simplify notation.

where $d_{0_{\text{MAX}}}$ stands for the maximum eigenvalue of $\mathbf{R}_{\text{dI}}$. Equation (12) shows that $\mathbf{R}_{\text{t}}$ becomes ill-conditioned if the noise power is much smaller than $d_{0_{\text{MAX}}}$. In this case, a regularization can be added to $\mathbf{R}_{\text{t}}$ to improve the computation of $\mathbf{x}_{\text{MVDR}}$ in (2). While this is usually done using ℓ_2-norm regularization (diagonal loading) [11], we will show that our ℓ_1-norm regularized algorithms also reduce the effects of the ill-conditioned $\mathbf{R}_{\text{t}}$, improving the computation of the beamformer and leading to low-cost algorithms. In addition, the use of homotopy allows our algorithms to choose just the right amount of regularization, reducing bias.

1) Reducing the system of equations when there are faulty sensors in the array: When a sensor j is not working properly, its measurements should be discarded. This information can be incorporated into the model with a modification of eq. (7), by introducing an $N \times N$ diagonal matrix $\mathbf{E}$, i.e.,

$$\mathbf{u}(n) = \mathbf{E}\left(\mathbf{b}_{\text{d}}\, \mathrm{s}_{\text{d}}(n) + \mathbf{B}_{\text{I}}\, \mathbf{s}_{\text{I}}(n) + \boldsymbol{\eta}(n)\right), \tag{13}$$

where the diagonal entries of $\mathbf{E}$ are equal to 0 for faulty sensors that do not contribute to beamforming, and 1 otherwise. When $e_{jj} = 0$, we zero the j-th element of all steering vectors, which eliminates the signal produced by sensor j. Using (13) to compute the correlation matrix, we obtain

$$\mathbf{R}_t = \mathbf{E}\left(\mathbf{R}_{\text{dI}} + \mathbf{R}_{\eta}\right)\mathbf{E}. \tag{14}$$

Assuming that the array has $F < N$ faulty sensors (but that there are still more working sensors than sources, i.e, $N - F > S$), and that these sensors are grouped such that $\mathbf{E}$ has the last F diagonal elements equal to 0, $\mathbf{R}_t$ is given by

$$\mathbf{R}_t = \begin{bmatrix} \mathbf{R}_{\text{R}} & \mathbf{0}_{(N-F)\times F} \\ \mathbf{0}_{F\times(N-F)} & \mathbf{0}_F \end{bmatrix}, \tag{15}$$

where $\mathbf{R}_{\text{R}} = \tilde{\mathbf{R}}_{\text{dI}} + \sigma_\eta^2 \mathbf{I}_{N-F}$ and $\tilde{\mathbf{R}}_{\text{dI}}$ is a matrix obtained from the first $N - F$ columns and the first $N - F$ rows of $\mathbf{R}_{\text{dI}}$. Ideally, if we know the matrix $\mathbf{E}$, we can define $\mathbf{b}_{\text{R}}$ as the first $N - F$ entries of $\mathbf{b}_{\text{d}}$, and solve the lower-dimension system of equations

$$\mathbf{R}_{\text{R}}\mathbf{x} = \mathbf{b}_{\text{R}}, \tag{16}$$

where $\mathbf{x}$ contains the $N - F$ non-zero entries of $\mathbf{x}_{\text{MVDR}}$. Matrix $\mathbf{R}_{\text{R}}$ will still be ill-conditioned when the eigenvalues of $\tilde{\mathbf{R}}_{\text{dI}}$ are much higher than the noise power, and regularization might be necessary to reduce the condition number and improve the computation of $\mathbf{x}$.

From (15) and (16), we see that the matrix $\mathbf{E}$ is required to obtain $\mathbf{R}_{\text{R}}$. In general, $\mathbf{E}$ is unknown and has to be estimated beforehand. In this paper, we use the energy detection method [22] to estimate the faulty sensors. However, we show through simulations that our ℓ_1-regularized algorithms are robust against errors in detecting faulty sensors, so that one might choose not to check for sensor failure.

Notice that in [23] a related problem is addressed and a lower bound to the error-variance in the estimation of the sources' direction of arrival (DOA) is obtained, when random sensor-breakdown occurs. In this paper, we do not address the problem of estimating the DOA, which is assumed known. We also do not consider the influence of errors in the estimation of the DOA to the computation of the beamformer. We consider only the effect of sensor failure on the beamformer itself. We show that using ℓ_1-norm regularized algorithms based on the homotopy algorithm, we obtain low-cost methods to introduce regularization and robustness against sensor failure, helping to improve the SINR performance.

III. Proposed complex homotopy algorithms

The complex homotopy algorithm (CH) was proposed in [18] as an extension of the real-valued homotopy technique [16] to the complex field. For both cases, the algorithm solves the optimization problem[2]

$$\underset{\mathbf{x}}{\text{minimize}} ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2/2 + w||\mathbf{x}||_1, \tag{17}$$

where $\mathbf{x}$ is a column vector with M entries, $\mathbf{A}$ is an $P \times M$ matrix, $\mathbf{y}$ is an $P \times 1$ vector, and w is a regularization parameter. The CH algorithm iteratively solves (17) using a support set Γ that is updated at every iteration. For each homotopy iteration, $\mathbf{x}$ must satisfy the following optimality conditions [18]

$$\begin{array}{l} \mathbf{a}_i^H(\mathbf{A}\mathbf{x} - \mathbf{y}) = -wz_i, \text{ for all } i \in \Gamma \\ |\mathbf{a}_i^H(\mathbf{A}\mathbf{x} - \mathbf{y})| < w, \text{ for all } i \in \Gamma_C \end{array}, \tag{18}$$

where Γ_C is the complement of Γ, and $\mathbf{z}$ denotes a vector obtained by applying the sign function elementwise on $\mathbf{x}$. For a large enough w, the solution of (17) will be $\mathbf{x} = \mathbf{0}$. The algorithm starts by computing $\max_i(|\mathbf{a}_i^H(\mathbf{y} - \mathbf{A}\mathbf{x})|)$, used to initialize w with the largest value for which Γ is non-empty. At each iteration, one element is added or removed from Γ, and w is moved to $w - \epsilon$, where ϵ is chosen so that $w - \epsilon$ is a breakpoint, i.e., the first point for which the new solution to (18) will need to add or remove an index in Γ. Denoting this new solution by $\mathbf{x} + \epsilon\partial\mathbf{x}$, (18) becomes[3]

$$\begin{array}{l} \mathbf{A}_\Gamma^H(\mathbf{A}\mathbf{x} - \mathbf{y}) + \epsilon\mathbf{A}_\Gamma^H\mathbf{A}\partial\mathbf{x} = -w\mathbf{z}_\Gamma + \epsilon\mathbf{z}_\Gamma \\ |\mathbf{a}_i^H(\mathbf{A}\mathbf{x} - \mathbf{y}) + \epsilon\mathbf{a}_i^H\mathbf{A}\partial\mathbf{x}| < w - \epsilon, i \in \Gamma_C \end{array}. \tag{19}$$

[2]Note that we introduce the algorithms for a general $P \times M$ matrix $\mathbf{A}$. For our beamforming approach, $M = P = (N - F)$ (the number of working sensors), $\mathbf{A}$ is an estimated version of $\mathbf{R}_\mathrm{R}$ and $\mathbf{y}$ corresponds to $\mathbf{b}_\mathrm{R}$.

[3]The subscript Γ is used to identify quantities related to the support set.

From (19), we take only the terms multiplied by ϵ and define a set of linear equations, which is used to compute $\partial\mathbf{x}$. Substituting $\partial\mathbf{x}$ in (19) we find ϵ, and update w with $w \leftarrow w - \epsilon$. The last step is the support update for the next iteration. The algorithm continues until $w = 0$ or some stopping criterion is met. References [16] and [18] present a detailed description of the algorithm.

A. The Complex Adaptive Re-Weighting Homotopy Algorithm

In [17] the real-valued homotopy algorithm was modified to solve the ℓ_1-weighted optimization problem

$$\underset{\mathbf{x}}{\text{minimize}} ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2/2 + \sum_{i=1}^{M} w_i |x_i|, \tag{20}$$

where w_i are positive weights. The motivation to modify the optimization problem and solve (20) instead of (17) was the possibility to adjust different weights to penalize the solution coefficients, which could be applied to enhance the level of sparsity of the solution and improve the performance [17].

The ARH algorithm applies a re-weighting approach to quickly compute $\mathbf{x}$, when the column vector $\mathbf{w}$ which contains the weights of (20) is replaced by a re-weighting vector $\tilde{\mathbf{w}}$. The idea behind the algorithm is that the solution moves to $\mathbf{x} + \delta\partial\mathbf{x}$ when $\mathbf{w}$ moves towards $\tilde{\mathbf{w}}$ along a straight line $(1-\delta)\mathbf{w} + \delta\tilde{\mathbf{w}}$, for $\delta \in [0, 1]$, where $\partial\mathbf{x}$ does not depend on δ. Simulation results in [17] have shown that ARH yields better performance and reconstruction accuracy than ℓ_1-based solvers (YALL1 [24], SpaRSA [25], SPGL1 [26]) used for recovering sparse signals from noisy measurements, while it requires lower computational complexity.

Based on the ARH algorithm of [17] and on the CH algorithm of [18], in [19] we developed an extension of ARH to complex-valued systems of equations. The technique was named the complex ARH algorithm and preliminary results presented in [19] showed that beamforming algorithms using C-ARH could lead to SINR performance gains. The technique is now presented in further detail, giving emphasis to beamforming applications.

To present the C-ARH technique, and later introduce the multi-candidate C-ARH algorithm, assume that $\mathbf{A}$, $\mathbf{x}$ and $\mathbf{y}$ are complex entities. For convenience, also assume that $\delta \in [0, \tilde{\delta}]$. In Section III-B, we use different values of $\tilde{\delta}$ to construct a diverse set of possible solutions, which is exploited by the multi-candidate algorithm to improve the performance. Considering these assumptions, the optimality conditions of C-ARH can be derived from (20), i.e.,

$$\begin{array}{l} \mathbf{A}_\Gamma^H(\mathbf{A}\mathbf{x} - \mathbf{y}) = -\mathbf{W}\mathbf{z}_\Gamma \\ |\mathbf{a}_i^H(\mathbf{A}\mathbf{x} - \mathbf{y})| < w_i \in \Gamma_C \end{array}, \tag{21}$$

where $\mathbf{W} = \text{diag}(\mathbf{w}_\Gamma)$, and $\mathbf{z}$ is a vector whose entries are the sign of the corresponding entries in $\mathbf{x}$. When $\mathbf{w}$ moves to $(1-\delta)\mathbf{w} + \delta\tilde{\mathbf{w}}$, (21) changes to

$$\begin{array}{l} \mathbf{A}_\Gamma^H(\mathbf{A}\mathbf{x}-\mathbf{y}) + \delta\mathbf{A}_\Gamma^H\mathbf{A}\partial\mathbf{x} = -\mathbf{W}\mathbf{z}_\Gamma + \delta(\mathbf{W}-\tilde{\mathbf{W}})\mathbf{z}_\Gamma \\ |\mathbf{a}_i^H(\mathbf{A}\mathbf{x}-\mathbf{y}) + \delta\mathbf{a}_i^H\mathbf{A}\partial\mathbf{x}| < w_i + \delta(\tilde{w}_i - w_i), i \in \Gamma_C \end{array},$$

and $\tilde{\mathbf{W}} = \text{diag}(\tilde{\mathbf{w}}_\Gamma)$. To compute $\partial\mathbf{x}$, we solve the system of equations

$$(\mathbf{A}_\Gamma^H\mathbf{A}_\Gamma)\partial\mathbf{x}_\Gamma = (\mathbf{W}-\tilde{\mathbf{W}})\mathbf{z}_\Gamma, \qquad (22)$$

where $z_i = \mathbf{a}_i^H(\mathbf{A}\mathbf{x}-\mathbf{y})/w_i, i \in \Gamma$, and the elements of $\partial\mathbf{x}$ outside the support are set to zero.

We have to check if a breakpoint occurs to update the support. A breakpoint occurs in two situations: when an element of $\mathbf{x} \in \Gamma$ changes sign, or when one inequality becomes an equality in (III-A). When an element changes sign, it must be removed from Γ. Recall that $\mathbf{x}$ is updated as $\mathbf{x} = \mathbf{x} + \delta\partial\mathbf{x}$. An element of $\mathbf{x}$ crosses zero when

$$\delta = -x_i/\partial x_i, \text{ for some } i \in \Gamma. \qquad (23)$$

Define $\mathbf{x}_R = \mathcal{R}e\{\mathbf{x}\}$, $\mathbf{x}_I = \mathcal{I}m\{\mathbf{x}\}$, $\mathbf{d}_R = \mathcal{R}e\{\partial\mathbf{x}\}$ and $\mathbf{d}_I = \mathcal{I}m\{\partial\mathbf{x}\}$, and recall that δ must be a real number in the interval $[0, \tilde{\delta}]$. The parameter δ has a real value in eq. (23), only if

$$x_{R_i}/d_{R_i} = x_{I_i}/d_{I_i}, i \in \Gamma. \qquad (24)$$

An element γ^- is removed from Γ if (24) holds for some i, and if (23) is in $[0, \tilde{\delta}]$, for the same breakpoint. If two or more breakpoints fulfill the restrictions, the smallest one is removed, which can be defined using

$$g = \min{}_+(-x_{R_i}/d_{R_i}), \text{ for all } x_{R_i}/d_{R_i} = x_{I_i}/d_{I_i}, i \in \Gamma$$

where $\min_+(\cdot)$ returns the smallest positive value in the argument. When g is empty, no term is removed, and C-ARH proceeds by choosing an element γ^+ that must be added to Γ. In this case, γ^+ is chosen by

$$\gamma^+ = \arg\max_{i \in \Gamma_C} |\mathbf{a}_i^H(\mathbf{A}\mathbf{x}-\mathbf{y})|, \qquad (25)$$

and δ is updated with the value of $\tilde{\delta}$. The last step is the update of $w_i \in \Gamma_C$, which is given by

$$w_i \leftarrow \max_j |\mathbf{a}_j^H(\mathbf{A}\mathbf{x}-\mathbf{y})|, \text{ for all } i \in \Gamma_C. \qquad (26)$$

The algorithm stops when the maximum $w_i \in \Gamma$ is smaller or equal to a pre-defined parameter τ. We summarize the algorithm in Table I.

1) Re-weighting choice: For this paper, we compute the re-weighting with

$$\tilde{w}_i = \min\left(\zeta, \zeta/\beta|x_i|\right), \text{ for all } i \in \Gamma, \tag{27}$$

where $\zeta = 2\sigma_\eta^2$ and $\beta = N||\mathbf{x}||_2^2/||\mathbf{x}||_1^2$. Note that this re-weighting is based on an approach proposed in [17]. We tried different re-weightings in the simulations presented in Section VII, and we selected the re-weighting that provided the best SINR performance for our beamforming scenario. Other re-weightings can also be applied.

2) Computational cost: The main contribution to the computational cost of each C-ARH iteration comes from computing

$$\mathbf{A}^H\left(\mathbf{y} - \mathbf{A}\mathbf{x}\right) = \mathbf{A}^H\mathbf{y} - \mathbf{A}^H\mathbf{A}\mathbf{x} \tag{28}$$

and $\partial\mathbf{x}$. The term (28) does not explicitly appear in Table I. However, if we consider the steps 2 and 9 (or 2 and 10, if the "else" condition of step 9 does not occur), we see that at every iteration, step 2 uses the elements of (28) computed with $\mathbf{a}_i$, $i \in \Gamma$, while step 9 (or 10) uses the elements which are calculated with the remaining $\mathbf{a}_i$. Recalling that steps 2 and 9 (or 2 and 10) occurs every iteration, we notice that (28) is computed every C-ARH step. This computation is costly in general, but it can be done with lower cost if some *a priori* information about $\mathbf{A}$ is available.

In a general approach, $\mathbf{A}$ can vary during the algorithm computations, requiring $\mathbf{A}^H\mathbf{A}$ and $\mathbf{A}^H\mathbf{y}$ in (28) to be re-computed at every iteration. Using the fact that $\mathbf{A}^H\mathbf{A}$ is symmetric, both terms are computed with $P(2M^2 + 6M)$ additions and $P(2M^2 + 6M) - (M^2 + 3M)$ multiplications. On the other hand, when $\mathbf{A}$ is invariant through the iterations, pre-computation of $\mathbf{A}^H\mathbf{y}$ and $\mathbf{A}^H\mathbf{A}$ can be used to reduce the number of operations. In this case, $\mathbf{A}^H\left(\mathbf{y} - \mathbf{A}\mathbf{x}\right)$ uses a matrix-vector product and the addition of two vectors, achieving a lower cost proportional to $|\Gamma|M$ per iteration (where $|\Gamma|$ is defined as the cardinality of Γ at a given iteration). The maximum cost per C-ARH iteration corresponds to $4|\Gamma|M + 5|\Gamma|$ multiplications, $4|\Gamma|M + 5|\Gamma|$ additions and $3|\Gamma|$ divisions, plus the computation of $\partial\mathbf{x}$ (which is the solution to a $|\Gamma| \times |\Gamma|$ system of equations), and the cost to obtain $\tilde{\mathbf{w}}$. The re-weighting applied in this paper uses an additional cost of $3|\Gamma|+2$ multiplications, $3|\Gamma|-2$ additions, $|\Gamma|+1$ divisions and $|\Gamma|$ square-roots per iteration.

To compute the solution to (20), C-ARH executes a number of iterations, where it adds or removes elements to the support of $\mathbf{x}$. Assuming that K is the total number of non-zero entries in the solution, the minimum number of iterations required to compute $\mathbf{x}$ is K. In this case, the algorithm only adds elements to the support, and the solution is obtained with $(K^2 + K)(2M + 4) - 2K$ additions, $(K^2 + K)(2M+4)+2K$ multiplications, $2K^2+3K$ divisions, $(K^2+K)/2$ square-roots, plus the solution of K

systems of equations with dimension $p \times p$, where p starts at 1 and linearly increases up to K during the iterations. Note that when the algorithm removes elements from Γ, the number of iterations increases. To check how frequently elements are removed from the support, we used C-ARH to solve a large number of examples with different values to $\mathbf{A}$ and $\mathbf{y}$. We compared the number of elements removed with the total amount of iterations used to compute the sparse vector $\mathbf{x}$, and we noted that these events rarely occur (in less than 1% of the iterations). Therefore, we assume that the minimum complexity obtained can be used as a reasonable approximation to the number of computations used by C-ARH.

3) C-ARH applied to beamforming: The algorithm presented in Table I can be applied to sparse systems of equations in general. For the purpose of this paper, we use the C-ARH to obtain a low-cost method to regularize eq. (3) and compute the MVDR beamformer.

Let $\mathbf{R}(n)$ be an approximation of $\mathbf{R}_t(n)$ and $\mathbf{R}_\mathrm{R}(n)$ be an approximation of $\mathbf{R}_\mathrm{R}$ at snapshot n. For beamforming, we solve eq. (20) using $\mathbf{A} = \mathbf{R}_\mathrm{R}(n)$ and $\mathbf{y} = \mathbf{b}_\mathrm{R}$, resulting in the following system of equations

$$\mathbf{R}_\mathrm{R}(n)\mathbf{x}(n) = \mathbf{b}_\mathrm{R}, \tag{29}$$

where the solution $\mathbf{x}(n)$ is used in

$$\mathbf{h}_\mathrm{R}(n) = \mathbf{x}(n)/\mathbf{b}_\mathrm{R}^H\mathbf{x}(n) \tag{30}$$

to compute the beamformer.

To solve (29), we first need to obtain $\mathbf{R}(n)$ and $\mathbf{E}$, so that the diagonal entries of $\mathbf{E}$ can be used to access the contribution of the working sensors, allowing us to obtain $\mathbf{R}_\mathrm{R}(n)$ and $\mathbf{b}_\mathrm{R}$. In this paper, we assume that $\mathbf{R}(n)$ is iteratively updated with

$$\mathbf{R}(n) = \nu\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}^H(n), \tag{31}$$

where $0 \le \nu < 1$ is the forgetting factor. Notice that (31) can be written as

$$\mathbf{R}(n) = \sum_{j=1}^{n} \nu^{n-j}\mathbf{u}(j)\mathbf{u}^H(j) + \nu^{n-j}\xi\mathbf{I}, \tag{32}$$

where $\xi\mathbf{I}$ is the initial regularization. Taking the expectation and recalling that $E\{\mathbf{u}(j)\mathbf{u}^H(j)\} = \mathbf{R}_t$, one can show that [2]

$$E\{\mathbf{R}(n)\} = \mathbf{R}_t/(1-\nu), \text{ when } n \to \infty. \tag{33}$$

Due to normalization in (30), the constant $(1-\nu)$ does not affect the computation of the beamforming solution, allowing the use of $\mathbf{R}(n)$ in (29).

The estimate of $\mathbf{E}$ can be obtained from the diagonal elements of $\mathbf{R}(n)$, using a technique based on the energy detection method [22]. To explain this approach, recall equations (15) and (33). It is easy to see that there are only two possibilities for the diagonal elements of $\mathbf{R}(n)$, when $n \to \infty$, i.e.,

$$r_{jj}(n \to \infty) \approx \begin{cases} \sigma_\eta^2/(1-\nu), & \text{if the } j\text{-th sensor is faulty} \\ (\tilde{r}_{\mathrm{dI}_{jj}}^2 + \sigma_\eta^2)/(1-\nu), & \text{otherwise.} \end{cases}$$

Assuming that the sources are uncorrelated, $\tilde{r}_{\mathrm{dI}_{jj}}^2$ is given by

$$\tilde{r}_{\mathrm{dI}_{jj}}^2 = \sum_{i=1}^{S} \sigma_{s_i}^2, \tag{34}$$

where each $\sigma_{s_i}^2$ corresponds to the variance of the i-th source, and $\sigma_{s_1}^2 = \sigma_{\mathrm{d}}^2$. In this case, one expects that the diagonal entries related to faulty sensors must have smaller variance, since they only measure noise. We exploit this fact to estimate $\mathbf{E}$.

To define if a sensor is faulty, consider that after a few snapshots it is possible to perceive that some diagonal entries of $\mathbf{R}(n)$ have higher values than others. The faulty sensors are identified using a threshold, based on the maximum $r_{jj}(n)$. We assume that if an element $r_{jj}(n)$ is at least 6dB smaller than the maximum entry, then the j-th sensor is faulty. The threshold is computed with

$$\mathrm{Thr} = 10^{-0.6}\mathrm{max}_j(r_{jj}(n)), \tag{35}$$

and all the diagonal entries of $\mathbf{R}(n)$ are compared to (35). If $r_{jj}(n)$ is smaller than Thr for some j, then the j-th sensor is faulty, and e_{jj} is set to 0. Otherwise, we set $e_{jj} = 1$. This technique is easy to implement (it only requires one multiplication and N comparisons) and is efficient for finding the faulty sensors, as can be seen in our simulation results. To further reduce the complexity, we can apply (35) only for the first t snapshots, turn the estimation of $\mathbf{E}$ off, and then use the last estimated matrix $\mathbf{E}$ for the remaining snapshots. Using this approach, we assume that t snapshots are sufficient to obtain a good estimate of the faulty sensors.

The C-ARH algorithm applied to beamforming is summarized in Table II. Recalling that $\mathbf{R}_{\mathrm{R}}(n)$ and $\mathbf{R}(n)$ are $M \times M$ and $N \times N$, respectively, the computational complexity corresponds to the cost of the algorithm in Table I, plus the cost to update $\mathbf{R}(n)$ and to compute $\mathbf{E}$ and $\mathbf{h}(n)$ – steps 1, 2 and 4 in Table II. These steps require the additional cost of $3M^2 + 5M + 3N + 5$ multiplications, $2M^2 + 4M + 2N - 1$ additions and 1 division.

B. *Multi-Candidate Complex Adaptive Re-Weighting Homotopy*

In the C-ARH algorithm, when we choose $\tilde{\delta} = 1$, $\mathbf{w}$ moves towards $\tilde{\mathbf{w}}$. However, we can choose a different $\tilde{\delta}$ and define a re-weighting that is a linear combination of $\mathbf{w}$ and $\tilde{\mathbf{w}}$. Since in general there

TABLE I

C-ARH ALGORITHM

```
Input: $\mathbf{A}$, $\mathbf{y}$, $\tau$, $\tilde{\delta}$          Output: $\mathbf{x}$, $\Gamma$
Initialize: $\partial\mathbf{x} = \mathbf{0}$, $\mathbf{x} = \mathbf{0}$, $w_i \leftarrow \max_i |\mathbf{a}_i^H \mathbf{y}|$ for all $i$, $\Gamma \leftarrow \arg\max_i |\mathbf{a}_i^H \mathbf{y}|$
     Repeat:
1        Select $\tilde{\mathbf{w}}$
2        For all $i \in \Gamma$, compute $z_i = \mathbf{a}_i^H (\mathbf{y} - \mathbf{A}\mathbf{x}) / w_i$
3        Solve $(\mathbf{A}_\Gamma^H \mathbf{A}_\Gamma)\partial\mathbf{x}_\Gamma = \text{diag}(\mathbf{w}_\Gamma - \tilde{\mathbf{w}}_\Gamma)\mathbf{z}_\Gamma$
4        Compute $\mathbf{x}_R = \mathcal{Re}\{\mathbf{x}_\Gamma\}$, $\mathbf{x}_I = \mathcal{Im}\{\mathbf{x}_\Gamma\}$, $\mathbf{d}_R = \mathcal{Re}\{\partial\mathbf{x}_\Gamma\}$ and $\mathbf{d}_I = \mathcal{Im}\{\partial\mathbf{x}_\Gamma\}$
5        $g = \text{min}_+(-x_{R_i}/d_{R_i})$, for all $x_{R_i}/d_{R_i} = x_{I_i}/d_{I_i}$
6        $\delta = \text{min}(g,\ \tilde{\delta})$
7        $\mathbf{x} = \mathbf{x} + \delta\partial\mathbf{x}$
8        $\mathbf{w}_\Gamma = \mathbf{w}_\Gamma + \delta(\tilde{\mathbf{w}}_\Gamma - \mathbf{w}_\Gamma)$
9      if $\delta < \tilde{\delta}$
           $\Gamma \leftarrow \Gamma \setminus \gamma^-$          ▷ Remove an element from $\Gamma$
       else
           $\gamma^+ = \arg\max_{i \in \Gamma_C} |\mathbf{a}_i^H (\mathbf{A}\mathbf{x} - \mathbf{y})|$
           $\Gamma \leftarrow \Gamma \cup \gamma^+$          ▷ Add a new element to $\Gamma$
       end
10     $w_i \leftarrow \max_j |\mathbf{a}_j^H (\mathbf{A}\mathbf{x} - \mathbf{y})|$,  for all $i \in \Gamma_C$
     until $\max_i(w_i) \leq \tau$
```

TABLE II

C-ARH ALGORITHM APPLIED TO BEAMFORMING

```
Input: $\mathbf{b}_\text{d}$, $\xi$, $\mathbf{u}(n)$, $\nu$, $\tilde{\delta}$, $\tau$, $t$          Output: $\mathbf{h}(n)$
Initialize: $\mathbf{R}(0) = \xi\mathbf{I}$, $\mathbf{x}(0) = \mathbf{0}$
    for $n = 1, 2, \cdots$
1      $\mathbf{R}(n) = \nu\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}(n)^H$
2      if $n < t$ then compute Thr $= 10^{-0.6}\max_j(r_{jj}(n))$ and estimate $\mathbf{E}$ to obtain $\mathbf{R}_\text{R}(n)$ and $\mathbf{b}_\text{R}$
3      Use $\tilde{\delta}$ and $\tau$ in C-ARH to solve $\mathbf{R}_\text{R}(n)\mathbf{x}(n) = \mathbf{b}_\text{R} \Rightarrow \mathbf{x}(n)$ (see Table I)
4      $\mathbf{h}(n) = \mathbf{x}(n)/\mathbf{b}_\text{R}^H\mathbf{x}(n)$
    end for
```

is no information about the weighting vector that generates the most accurate $\mathbf{x}$, the combination of the two weighting vectors can be a better option than only $\tilde{\mathbf{w}}$. In this context, MC-C-ARH is proposed to exploit multiple weight choices.

We start with the definition of a set $\mathbf{\Lambda}$ of N_C candidates for $\tilde{\delta}$. For each $\tilde{\delta} \in \mathbf{\Lambda} = \{\lambda_1, \lambda_2 \ldots \lambda_{N_C}\}$ the algorithm computes the corresponding solution $\mathbf{x}_i(n)$. A comparison criterion (e.g., the mean-square error (MSE) or SINR) is used to define the best solution computed for each snapshot. The candidate with the best figure of merit is selected. In Table III we summarize the algorithm, applied to beamforming.

In general, $\mathbf{R}_\text{d}$, $\mathbf{R}_\text{I}$ and $\mathbf{R}_\eta$ are not available, and an indirect method is required to select the candidate which provides the highest SINR. Define $\mathbf{R}_{\text{I}\eta} = \mathbf{R}_\text{I} + \mathbf{R}_\eta$ and recall that $\mathbf{R}_\text{d} = \sigma_\text{d}^2 \mathbf{b}_\text{d} \mathbf{b}_\text{d}^H$ and $\mathbf{b}_\text{d}^H \mathbf{h}_i(n) = 1$. The SINR for the i-th candidate is given by

$$\text{SINR}_i(n) = \mathbf{h}_i^H(n)\mathbf{R}_\text{d}\mathbf{h}_i(n)/\mathbf{h}_i^H(n)\mathbf{R}_{\text{I}\eta}\mathbf{h}_i(n) = \sigma_\text{d}^2/\mathbf{h}_i^H(n)\mathbf{R}_{\text{I}\eta}\mathbf{h}_i(n), \tag{36}$$

and it is maximized when $\mathbf{h}_i^H(n)\mathbf{R}_{\text{I}\eta}\mathbf{h}_i(n)$ is minimum. Since $\mathbf{R}_{\text{I}\eta}$ is unknown, (36) cannot be directly minimized. As an alternative, we note that the minimization of

$$\mathbf{h}_i^H(n)\mathbf{R}_\text{t}\mathbf{h}_i(n) = \sigma_\text{d}^2 + \mathbf{h}_i^H(n)\mathbf{R}_{\text{I}\eta}\mathbf{h}_i(n), \tag{37}$$

also maximizes the SINR, and an estimate $\mathbf{R}(n)$ of $\mathbf{R}_\text{t}$ can be used to compute (37). However, the computation of (37) is costly, proportional to $O(N^2)$. To reduce the number of computations, we propose a simpler method, with cost $O(N)$.

Defining $\mathbf{R}_{\text{I}\eta} = \mathbf{D}_{\text{I}\eta} + \mathbf{G}$, where $\mathbf{D}_{\text{I}\eta} = \sigma_{\text{I}\eta}^2\mathbf{I}$ has the diagonal entries of $\mathbf{R}_{\text{I}\eta}$, and $\mathbf{G}$ contains the other elements, we can write

$$\text{SINR}_i(n) = 1/[(\sigma_{\text{I}\eta}^2/\sigma_\text{d}^2)||\mathbf{h}_i(n)||_2^2 + (1/\sigma_\text{d}^2)\mathbf{h}_i^H(n)\mathbf{G}\mathbf{h}_i(n)]. \tag{38}$$

If we consider only two candidates, and that $\text{SINR}_1(n) > \text{SINR}_2(n)$, then we obtain

$$||\mathbf{h}_1(n)||_2^2 < ||\mathbf{h}_2(n)||_2^2 + [\mathbf{h}_2^H(n)\mathbf{G}\mathbf{h}_2(n) - \mathbf{h}_1^H(n)\mathbf{G}\mathbf{h}_1(n)]/\sigma_{\text{I}\eta}^2. \tag{39}$$

Assuming that $\mathbf{G}$ is small compared to $\sigma_{\text{I}\eta}^2\mathbf{I}$, then the second term in the right-hand side of (39) can be neglected. Extending the idea to N_c candidates, we obtain the proposed selection algorithm

$$\mathbf{h}_\text{MAX}(n) = \mathbf{h}_k(n) \ \text{ when } \ k = \text{arg}\ \min_i(||\mathbf{h}_i(n)||_2^2), \ i = 1, 2, \ldots, N_c. \tag{40}$$

Our simulations show that MC-C-ARH improves the SINR performance, when compared to C-ARH.

1) Computational cost: MC-C-ARH starts with the update of $\mathbf{R}(n)$ (step 1, Table III) and the computation of $\mathbf{E}$ (step 2, Table III). Then, the algorithm computes the C-ARH solution $\mathbf{x}_i(n)$ (step 3, Table III), $\mathbf{h}_i(n)$ (step 4, Table III) and $||\mathbf{h}_i(n)||^2$ (step 5, Table III) for each candidate. Recalling that $N \times N$ is the dimension of $\mathbf{R}(n)$, while $M \times M$ is the dimension of $\mathbf{R}_\text{R}(n)$, $\mathbf{R}(n)$ is updated with $3M^2 - 3M + 3N$ multiplications and $2M^2 - 2M + 2N$ additions. The cost to compute $\mathbf{h}_i(n)$

is $8M + 4$ multiplications, $6M - 1$ additions and 1 division, while the computation of $||\mathbf{h}_i(n)||^2$ uses $2M$ multiplications and $2M - 1$ additions for each candidate. The total computational cost is $3M^2 + M(10N_c - 3) + 3N + 4N_c + 1$ multiplications, $2M^2 + M(8N_c - 2) + 2N - 2N_c$ additions, N_c divisions, plus the cost to solve the C-ARH algorithm N_c times.

TABLE III

MC-C-ARH ALGORITHM APPLIED TO BEAMFORMING

```
Input: $\mathbf{b}_\mathrm{d}$, $\xi$, $\mathbf{u}(n)$, $\nu$, $\mathbf{\Lambda}$, $\tau$, $t$          Output: $\mathbf{h}_\mathrm{MAX}(n)$
Initialize: $\mathbf{R}(0) = \xi\mathbf{I}, \mathbf{x}_i(0) = \mathbf{0}$, $\forall\lambda_i \in \mathbf{\Lambda}$
   for $n = 1, 2, \cdots$
1    $\mathbf{R}(n) = \nu\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}(n)^H$
2    if $n < t$ then compute Thr $= 10^{-0.6}\max_j(r_{jj}(n))$ and estimate $\mathbf{E}$ to obtain $\mathbf{R}_\mathrm{R}(n)$ and $\mathbf{b}_\mathrm{R}$
     for all $\lambda_i \in \mathbf{\Lambda}$ :
3       Use C-ARH with $\tilde{\delta} = \lambda_i$ and $\tau$ to solve $\mathbf{R}_\mathrm{R}(n)\mathbf{x}_i(n) = \mathbf{b}_\mathrm{R} \Rightarrow \mathbf{x}_i(n)$ (see Table I)
4       Compute $\mathbf{h}_i(n) = \mathbf{x}(n)/\mathbf{b}_\mathrm{R}^H\mathbf{x}_i(n)$
5       Compute $||\mathbf{h}_i(n)||_2^2$ for all $i$
     end for
6    $\mathrm{m} = \arg\min_i\{||\mathbf{h}_i(n)||^2\}$          $\triangleright$ Find the best candidate
7    $\mathbf{h}_\mathrm{MAX}(n) = \mathbf{h}_\mathrm{m}(n)$
   end for
```

IV. ITERATIVE ALGORITHMS USING COMPLEX HOMOTOPY TECHNIQUES

In this section, we propose iterative algorithms based on the C-ARH technique. The idea behind these approaches is that we can compute the solution at snapshot n by adding an update term to the solution obtained at snapshot $n-1$, reducing the computations to obtain $\mathbf{x}(n)$. For this purpose, consider that the solution to (29) at snapshot n is given by

$$\mathbf{x}(n) = \mathbf{x}(n-1) + \Delta\mathbf{x}(n), \tag{41}$$

where $\Delta\mathbf{x}(n)$ is the updating term. Since $\mathbf{x}(n-1)$ is known at snapshot n, we use (41) in (29), to obtain

$$\mathbf{R}_\mathrm{R}(n)\Delta\mathbf{x}(n) = \boldsymbol{\beta}(n), \tag{42}$$

where we define

$$\boldsymbol{\beta}(n) = \mathbf{b}_\mathrm{R} - \mathbf{R}_\mathrm{R}(n)\mathbf{x}(n-1). \tag{43}$$

In this case, we can write the minimization problem

$$\underset{\Delta\mathbf{x}}{\text{minimize}}||\mathbf{R}_{\mathrm{R}}(n)\Delta\mathbf{x}(n)-\boldsymbol{\beta}(n)||_2^2/2+\sum_{i=1}^{M}w_i|\Delta x_i(n)|, \tag{44}$$

which is similar to the minimization problem solved by C-ARH in (20). Therefore, we can define $\mathbf{A}=\mathbf{R}_{\mathrm{R}}(n)$, $\mathbf{y}=\boldsymbol{\beta}_{\mathrm{R}}(n)$ and $\mathbf{x}=\Delta\mathbf{x}(n)$, and use C-ARH to compute $\Delta\mathbf{x}(n)$. The result is then applied in (41), to compute $\mathbf{x}(n)$. We call this approach the iterative C-ARH (It-C-ARH) algorithm.

The It-C-ARH algorithm computes $\boldsymbol{\beta}(n)$ at every snapshot, which requires the computation of a complex matrix-vector product and the addition of two complex vectors. Using this approach, the total number of operations corresponds to $4M^2$ multiplications and $4M^2$ additions. However, the computation of $\boldsymbol{\beta}(n)$ at snapshot n can be implemented less costly, using quantities computed in the previous snapshot. For this purpose, assume that $\mathbf{R}(n)$ is updated as presented in eq. (31), and use it in (43) to write

$$\boldsymbol{\beta}(n)=\mathbf{b}_{\mathrm{R}}-\left[\nu\mathbf{R}_{\mathrm{R}}(n-1)+\mathbf{u}_{\mathrm{R}}(n)\mathbf{u}_{\mathrm{R}}^H(n)\right]\mathbf{x}(n-1)=(1-\nu)\mathbf{b}_{\mathrm{R}}+\nu\boldsymbol{\zeta}(n-1)-\mathbf{u}_{\mathrm{R}}(n)z^*(n) \tag{45}$$

where $\mathbf{u}_{\mathrm{R}}(n)$ contains only signals obtained from sensors working properly. We define the residue

$$\boldsymbol{\zeta}(n-1)=\mathbf{b}_{\mathrm{R}}-\mathbf{R}_{\mathrm{R}}(n-1)\mathbf{x}(n-1) \tag{46}$$

and

$$z(n)=\mathbf{x}^H(n-1)\mathbf{u}_{\mathrm{R}}(n). \tag{47}$$

Using (45) and (41), $\boldsymbol{\zeta}(n)$ can be written in terms of the $\Delta\mathbf{x}(n)$, i.e.,

$$\boldsymbol{\zeta}(n)=\mathbf{b}_{\mathrm{R}}-\mathbf{R}_{\mathrm{R}}(n)(\mathbf{x}(n-1)+\Delta\mathbf{x}(n))=\boldsymbol{\beta}(n)-\mathbf{R}_{\mathrm{R}}(n)\Delta\mathbf{x}(n), \tag{48}$$

which can be efficiently computed to reduce the computational cost. When C-ARH computes $\Delta\mathbf{x}(n)$, it computes only the K entries in the support of $\Delta\mathbf{x}(n)$. In this case, C-ARH gives us perfect knowledge of the K non-zero entries of $\Delta\mathbf{x}(n)$. With this information, we can exclude the columns of $\mathbf{R}_{\mathrm{R}}(n)$ that are multiplied by the zero entries of $\Delta\mathbf{x}(n)$ in (48), such that the residue can be computed with $4KM$ multiplications and $4KM$ additions. Using this result to compute (45), the computational cost to calculate $\boldsymbol{\beta}(n)$ corresponds to $4KM+6M$ multiplications and $4KM+6M$ additions.

In Table IV the It-C-ARH algorithm is presented, and in Table V we describe the iterative MC-C-ARH algorithm, introducing multiple candidates for $\tilde{\delta}$. Notice that we use the same criterion applied by MC-C-ARH to select the best candidate in the iterative multi-candidate technique.

Computational cost of the It-C-ARH algorithm: Compared to C-ARH (see Table II), the iterative technique requires the additional computation of $z(n)$, $\boldsymbol{\beta}(n)$, $\mathbf{x}(n)$ and $\boldsymbol{\zeta}(n)$ (respectively steps 3, 4, 6

TABLE IV

IT-C-ARH ALGORITHM APPLIED TO BEAMFORMING

```
Input: $\mathbf{b}_{\mathrm{d}}$, $\xi$, $\mathbf{u}$, $\nu$, $\tilde{\delta}$, $\tau$, $t$        Output: $\mathbf{h}(n)$
Initialize: $\mathbf{R}(0) = \xi\mathbf{I}, \mathbf{x}(0) = \mathbf{0}$, $\boldsymbol{\zeta}(0) = \mathbf{0}$
   for $n = 1, 2, \cdots$
1    $\mathbf{R}(n) = \nu\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}(n)^H$
2    if $n < t$ then compute Thr $= 10^{-0.6}\max_j(r_{jj}(n))$ and estimate $\mathbf{E}$ to obtain $\mathbf{R}_{\mathrm{R}}(n)$ and $\mathbf{b}_{\mathrm{R}}$
3    $z(n) = \mathbf{x}^H(n-1)\mathbf{u}_{\mathrm{R}}(n)$
4    $\boldsymbol{\beta}(n) = \nu\boldsymbol{\zeta}(n-1) - \mathbf{u}_{\mathrm{R}}(n)z^*(n) + (1-\nu)\mathbf{b}_{\mathrm{R}}$
5    Use C-ARH with $\tilde{\delta}$ and $\tau$ to solve $\mathbf{R}_{\mathrm{R}}(n)\Delta\mathbf{x}(n) = \boldsymbol{\beta}(n) \Rightarrow \Delta\mathbf{x}(n)$, $\Gamma$ (see Table I)
6    Compute $\mathbf{x}(n) = \mathbf{x}(n-1) + \Delta\mathbf{x}(n)$
7    Compute $\boldsymbol{\zeta}(n) = \boldsymbol{\beta}(n) - \mathbf{R}_{\mathrm{R}}(n)\Delta\mathbf{x}(n)$
8    Compute $\mathbf{h}(n) = \mathbf{x}(n)/\mathbf{b}_{\mathrm{R}}{}^H\mathbf{x}(n)$
   end for
```

and 7 in Table IV). The term $(1-\nu)\mathbf{b}_{\mathrm{R}}$ does not change through the snapshots and can be pre-computed to reduce the computations. The total implementation cost per snapshot is $3M^2 + (15 + 4K)M + 3N + 5$ multiplications, $2M^2 + (14 + 4K)M + 2N + 2K - 3$ additions, 1 division, plus the cost to compute the C-ARH algorithm in Table I.

Computational cost of the It-MC-C-ARH algorithm: The computation of the iterative multi-candidate algorithm differs from the MC-C-ARH algorithm in the addition of steps 3, 4, 6 and 7 in Table V. With the additional steps, the complexity cost increases and is given by $3M^2+(4KN_c+20N_c-3)M+3N+4N_c+1$ multiplications, $2M^2 + (4KN_c + 18N_c - 2)M + 2N - 2KN_c - 4N_c$ additions, N_c divisions and the computation of C-ARH (Table I) N_c times.

V. THE HOMOTOPY ALGORITHMS USING DCD ITERATIONS

The C-ARH algorithm solves a linear system of equations (step 3, Table I), which is costly to compute. In this case, an efficient method to compute the solution is very important to keep the complexity low. Prior work report the use of DCD iterations [20] to solve systems of equations with a reduced number of operations. Since it avoids multiplications and divisions, which are costly to implement, many applications of this technique can be found in the literature. In [27] and [28], for instance, DCD iterations are applied to obtain low-complexity RLS and affine projection (AP) algorithms. To obtain the first, the RLS problem is expressed in terms of auxiliary equations with respect to increments of the filter weights. DCD iterations are then used to solve the auxiliary equations, resulting in a complexity reduction. The low-complexity AP

TABLE V

IT-MC-C-ARH ALGORITHM APPLIED TO BEAMFORMING

```
Input: $\mathbf{b}_\text{d}$, $\xi$, $\mathbf{u}$, $\nu$, $\mathbf{\Lambda}$, $\tau$, $t$        Output: $\mathbf{h}_\text{MAX}(n)$
Initialize: $\mathbf{R}(0) = \xi\mathbf{I}, \mathbf{x}_i(0) = \mathbf{0}$, $\boldsymbol{\zeta}_i(0) = \mathbf{0}$, $\forall\lambda_i \in \mathbf{\Lambda}$
     for $n = 1, 2, \cdots$
1      $\mathbf{R}(n) = \nu\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}(n)^H$
2      if $n < t$ then compute $\text{Thr} = 10^{-0.6}\max_j(r_{jj}(n))$ and estimate $\mathbf{E}$ to obtain $\mathbf{R}_\text{R}(n)$ and $\mathbf{b}_\text{R}$
       for all $\lambda_i \in \mathbf{\Lambda}$:
3        $z_i(n) = \mathbf{x}_i^H(n-1)\mathbf{u}_\text{R}(n)$
4        $\boldsymbol{\beta}_i(n) = \nu\boldsymbol{\zeta}_i(n-1) - \mathbf{u}_\text{R}(n)z_i^*(n) + (1-\nu)\mathbf{b}_\text{R}$
5        Use C-ARH with $\lambda_i$ and $\tau$ to solve $\mathbf{R}_\text{R}(n)\Delta\mathbf{x}_i(n) = \boldsymbol{\beta}_i(n) \Rightarrow \Delta\mathbf{x}_i(n)$, $\Gamma$ (see Table I)
6        Compute $\mathbf{x}_i(n) = \mathbf{x}_i(n-1) + \Delta\mathbf{x}_i(n)$
7        Compute $\boldsymbol{\zeta}_i(n) = \boldsymbol{\beta}(n) - \mathbf{R}_\text{R}(n)\Delta\mathbf{x}_i(n)$
8        Compute $\mathbf{h}_i(n) = \mathbf{x}_i/{\mathbf{b}_\text{R}}^H\mathbf{x}_i$
9        Compute $||\mathbf{h}_i(n)||_2^2$ for all i
       end for
10     $\text{m} = \arg\min_i\{||\mathbf{h}_i(n)||_2^2\}$          ▷ Find the best candidate
11     $\mathbf{h}_\text{MAX}(n) = \mathbf{h}_\text{m}(n)$
     end for
```

algorithm is obtained with a modification of the AP technique, which incorporates the DCD to update the filter weights, resulting in a method less costly to implement than NLMS [2]. In [29], the DCD is used to implement the MVDR beamformer in an FPGA. The estimate of the correlation matrix $\mathbf{R}(n)$ is updated with a rectangular window, and DCD is applied to compute a low-cost solution to $\mathbf{R}(n)\mathbf{x}(n) = \mathbf{b}_\text{d}$. $\mathbf{x}(n)$ is then used to compute the beamformer with eq. (30). In [30], a DCD-homotopy technique based on the algorithm of [16] is also proposed. The algorithm is obtained with a modification of the DCD, and is applied to recover sparse signals, in system identification problems.

Although there are different uses, DCD-based approaches in general lead to low-cost techniques, suitable for hardware implementation. This fact motivated us to employ DCD iterations to compute step 3 in Table I. For this purpose, we apply a leading element version of DCD [31], which is briefly described in Table VI. The algorithm solves $\mathbf{A}\mathbf{x} = \mathbf{y}$, where $\mathbf{A}$ is an $L \times L$ matrix, and $\mathbf{x}$ and $\mathbf{y}$ are column vectors with L entries. The step-size α depends on the value of parameter H, which determines the range of the elements of $\mathbf{x}$ (which is assumed to be represented with M_b bits in fixed-point format). The algorithm starts updating the most significant bits of the elements of $\mathbf{x}$, moving down to the least significant bits. The number of iterations is restricted to be less than a small value N_u, which helps to keep the number of computations low. Selecting a power-of-two step size, the algorithm is implemented with further complexity reduction, since divisions and multiplications can be replaced by bit-shifts. The maximum computational cost is $(4L+1)N_u+M_b$ additions, but this is the worst case, when the maximum number of iterations N_u is used.

In this paper, the use of DCD is particularly interesting for the iterative algorithms, since the previous solution can be employed as an initial condition to DCD, allowing the use of a small N_u. The non-iterative algorithms cannot use this initial condition, and would require a large number of DCD iterations N_u to compute the solution. For this reason, we focus now on the iterative techniques. For It-C-ARH and It-MC-C-ARH, DCD is used to solve (22), where $\mathbf{A} = \mathbf{A}_\Gamma^H \mathbf{A}_\Gamma$ and $\mathbf{y} = (\mathbf{W} - \tilde{\mathbf{W}})\mathbf{z}_\Gamma$. To further save operations, we can also select the forgetting factor as $\nu = 1 - 2^{-l}$, where l is a positive integer. This choice of ν substitutes multiplications by additions and bit-shifts in eqs. (31) and (45), reducing the complexity to update $\mathbf{R}_\mathrm{R}(n)$ and $\boldsymbol{\beta}(n)$. As it is shown in Section VII, the algorithms using DCD outperform our previous approaches, with further reduction in the number of computations.

TABLE VI

DCD WITH LEADING ELEMENT

```
Input: $\mathbf{y}$, $\mathbf{A}$, $M_b$, $H$, $N_u$          Output: $\boldsymbol{\zeta}_{\mathrm{DCD}}$, $\mathbf{x}$
Initialize: $m = 0$, $\alpha = H$, $\boldsymbol{\zeta}_{\mathrm{DCD}} = \mathbf{y}$
   for $n = 1, 2, \cdots, N_u$
1    $[k, s] = \arg\max_{p=1,\cdots,K}\{|\mathcal{R}e(\zeta_{\mathrm{DCD}_p})|, |\mathcal{I}m(\zeta_{\mathrm{DCD}_p})|\} \Rightarrow$ go to step 4
2    $m = m + 1$
3    $\alpha = \alpha/2$
4    if $m > M_b$, the algorithm stops
5    if $s = 1$
       $\zeta_{\mathrm{DCD}_{tmp}} = \mathcal{R}e(\zeta_{\mathrm{DCD}_k})$
     else
       $\zeta_{\mathrm{DCD}_{tmp}} = \mathcal{I}m(\zeta_{\mathrm{DCD}_k})$
     end
6    if $|\zeta_{\mathrm{DCD}_{tmp}}| \leq (\alpha/2)\mathbf{A}_{k,k}$, then go to step 1
7    $x_k = x_k + \mathrm{sign}(\zeta_{\mathrm{DCD}_{tmp}})s\alpha$
8    $\boldsymbol{\zeta}_{\mathrm{DCD}} = \boldsymbol{\zeta}_{\mathrm{DCD}} - \mathrm{sign}(\zeta_{\mathrm{DCD}_{tmp}})s\alpha\mathbf{a}_k$
9  end for
```

VI. Analysis

In this section, we summarize some analysis results that explain and give insight into the algorithms' behavior. We show why the C-ARH algorithm provides SINR performance gains even when the system of equations is not sparse, and we also define the range of values for $\tilde{\delta}$ for the multi-candidate algorithms. In addition, we compare the iterative and non-iterative algorithms to give some insights into the performance differences observed in the simulations. We then dedicate a section to summarize the computational complexity, and to provide some implementation tools to achieve low-cost algorithms.

A. *C-ARH applied to regularize sparse and non-sparse problems*

At the k-th homotopy iteration[4], the C-ARH algorithm (see Table I) uses

$$\mathbf{A}_\Gamma^H(\mathbf{A}\mathbf{x}(k-1)-\mathbf{y})+\delta\mathbf{A}_\Gamma^H\mathbf{A}\partial\mathbf{x}(k)=-\mathbf{W}\mathbf{z}_\Gamma(k)+\delta(\mathbf{W}-\tilde{\mathbf{W}})\mathbf{z}_\Gamma(k) \tag{49}$$

to compute the solution $\mathbf{x}(k)=\mathbf{x}(k-1)+\partial\mathbf{x}(k)$. Assume $\delta=1$, and recall that $\mathbf{z}_{\Gamma_i}(k)=\mathbf{x}_i(k)/|\mathbf{x}_i(k)|$. Equation (49) can be written as

$$\mathbf{A}_\Gamma^H\mathbf{A}_\Gamma\mathbf{x}(k)-\mathbf{A}_\Gamma^H\mathbf{y}=-\tilde{\mathbf{W}}\mathbf{z}_\Gamma(k), \tag{50}$$

where $\tilde{\mathbf{W}}\mathbf{z}_\Gamma(k)=\mathbf{D}(k)\mathbf{x}(k)$, and $\mathbf{D}(k)$ is a diagonal matrix with diagonal entries

$$d_{ii}(k)=\tilde{\mathbf{w}}_i/|x_i(k)|,\quad i\in\Gamma. \tag{51}$$

Using $\mathbf{D}(k)$ in eq. (50), we obtain

$$\left(\mathbf{A}_\Gamma^H\mathbf{A}_\Gamma+\mathbf{D}(k)\right)\mathbf{x}(k)=\mathbf{A}_\Gamma^H\mathbf{y}, \tag{52}$$

showing that a diagonal regularization is introduced, which will be beneficial when (50) is ill-conditioned. As shown in Section II-A, when the number of interferers is small, (50) is ill-conditioned, and the regularization provided by C-ARH helps to improve the estimates.

B. *Selection of the regularization parameter $\tilde{\delta}$*

The values of the $\tilde{\delta}$ must all be in the interval $[0,1]$ to avoid negative weights. In fact, the weight update is

$$\mathbf{w}\leftarrow(1-\delta)\mathbf{w}+\delta\tilde{\mathbf{w}}\in\Gamma, \tag{53}$$

where we just reorganized step 8 in Table I to make clear the contribution of $\mathbf{w}$ and $\tilde{\mathbf{w}}$ to the right-hand side of the equation. It is easy to note in eq. (53) that δ must be in $[0,1]$ to guarantee that $\mathbf{w}$ is always positive.

[4]Note that we use k to denote different homotopy iterations and avoid confusion with index n, which denotes snapshots.

C. Differences between the iterative and non-iterative algorithms

The C-ARH algorithm computes an iterative solution to the minimization problem presented in eq. (20) (where $\mathbf{A} = \mathbf{R}_{\mathrm{R}}(n)$ and $\mathbf{y} = \mathbf{b}_{\mathrm{R}}$ for beamforming). The algorithm uses an ℓ_1-norm regularization to $\mathbf{x}(n)$, helping ill-conditioned systems of equations and also favoring sparse solutions . The iterative approaches, on the other hand, solve a modified minimization problem (see (44)), where the ℓ_1-norm regularization is applied to the solution update, $\Delta\mathbf{x}(n)$. Since the two minimization problems are essentially different, we expect the techniques to perform distinctly under the same conditions. The simulations presented in Section VII corroborate this fact, showing that recursive techniques may outperform C-ARH for beamforming. In addition, recall that the iterative algorithms use C-ARH to compute the entries in the support of $\mathbf{x}(n)$. Since It-C-ARH uses the solution of the previous snapshot as an initial condition to compute $\mathbf{x}(n)$, one can expect this previous solution to be closer to the ideal solution than the zero-initial condition used by the non-iterative techniques. In this case, we expect to use less updates to obtain the It-C-ARH solution, reducing the number computations. In Section VII, we present simulations to support these observations.

D. Computational complexity

In this section, we summarize the computational complexity of the proposed algorithms. Tables VII - X are arranged in a nested structure, each table adding a layer needed for the solution of the beamforming problem.

In Table VII, we show the computational cost per iteration of the C-ARH algorithm (as described in Table I). The number of operations is presented as a function of the support size $|\Gamma|$ at each homotopy iteration. At every iteration, the algorithm computes the solution of a linear system of equations, with the dimension $|\Gamma| \times |\Gamma|$. Different approaches can be used to solve the system of equations, such as an LU factorization, which is an $O(|\Gamma|^3)$ technique, and the DCD algorithm, which is the low-cost alternative used in this paper. The total cost to compute $\mathbf{x}(n)$ (assuming that the maximum support size is given by K) is presented in Table VIII. The cost to implement C-ARH using the DCD with a leading element is also detailed.

Table IX shows the complexity of the algorithms proposed for beamforming. In our simulations comparing the iterative and non-iterative techniques (see Section VII), we notice that It-C-ARH and It-MC-ARH use a very low number of homotopy iterations per snapshot, which makes their computational cost much lower than the cost of C-ARH and MC-C-ARH. The simulations also indicate that the iterative algorithms have better SINR performance. When we use the DCD in the C-ARH algorithm, further reduction of the computational cost is obtained, since multiplications are replaced by additions and bit-shifts, and the parameter N_u can be designed to be a small number. Table X summarizes the number of computations used by the recursive algorithms using DCD iterations.

The dominant terms in the computational complexity of all the homotopy-based techniques are K^2M (see Table VIII) and M^2 (see Tables IX and X). Since the maximum number of elements in the support K can be close to the value of M, one would expect the cost to implement the proposed techniques to be cubic in the value of M. However, as it is shown in our simulations in Section VII, the iterative algorithms require a very low number of homotopy iterations per snapshot, such that $K \ll M$ and the complexity is reduced to $O(M^2)$.

Note that we use (27) for all techniques presented in this paper. Selecting a different re-weighting, the required number of computations may change, modifying the values presented in Tables VII and VIII.

TABLE VII

COMPUTATIONAL COMPLEXITY OF C-ARH PER HOMOTOPY ITERATION

Algorithm	$+$	$\times$	$\div$	$\sqrt{\cdot}$
C-ARH	$\lvert\Gamma\rvert(4M+8)-2$	$\lvert\Gamma\rvert(4M+8)+2$	$4\lvert\Gamma\rvert+1$	$\lvert\Gamma\rvert$
	Plus the solution of a $\lvert\Gamma\rvert \times \lvert\Gamma\rvert$ linear system of equations			
C-ARH (DCD)	$4\lvert\Gamma\rvert(M+N_u+2)+N_u+M_b-2$	$\lvert\Gamma\rvert(4M+8)+2$	$4\lvert\Gamma\rvert+1$	$\lvert\Gamma\rvert$

TABLE VIII

MINIMUM COMPUTATIONAL COMPLEXITY OF C-ARH PER SNAPSHOT (WHEN THE TOTAL SUPPORT SIZE IS K)

Algorithm	$+$	$\times$	$\div$	$\sqrt{\cdot}$
C-ARH	$K^2(2M+4)+K(2M+2)$	$K^2(2M+4)+K(2M+6)$	$2K^2+3K$	$(K^2+K)/2$
	Plus the solution of K systems of equations with dimension $p \times p$, where $p = 1, 2, \ldots, K$			
C-ARH (DCD)	$K^2(2M+6)+K(2M+2+6N_u+M_b)$	$K^2(2M+4)+K(2M+6)$	$2K^2+3K$	$(K^2+K)/2$

VII. SIMULATIONS

In our simulations, we compare the SINR performance of the proposed algorithms and techniques from the literature. We consider a 64-sensor ULA and assume one direction of interest at an angle of 20^o, and 4 interferers with angles 30^o, 45^o, 53^o and 60^o. We perform two groups of simulations. In the first scenario, we assume that there are 5 faulty sensors in the array, and we use a faulty-sensor detector

TABLE IX

COMPUTATIONAL COMPLEXITY PER SNAPSHOT OF THE ALGORITHMS APPLIED TO BEAMFORMING

Algorithm	$+$	$\times$	$\div$	C-ARH (Tab. VIII)
C-ARH (beamf.)	$2M^2+4M+2N-1$	$3M^2+5M+3N+5$	1	1
MC-C-ARH	$2M^2+(8N_c-2)M+2N-2N_c$	$3M^2+(10N_c-3)M+3N+4N_c+1$	N_c	N_c
It-C-ARH	$2M^2+(4K+14)M+2N+2K-3$	$3M^2+(4K+15)M+3N+5$	1	1
It-MC-C-ARH	$2M^2+(4KN_c+18N_c-2)M$ $+2N+2KN_c-4N_c$	$3M^2+(4KN_c+20N_c-3)M$ $+3N+4N_C+1$	N_c	N_c

TABLE X

TOTAL COMPUTATIONAL COMPLEXITY OF THE ALGORITHMS USING DCD ITERATIONS

Algorithm	$+$	$\times$	$\div$	$\sqrt{\cdot}$
DCD-It-C-ARH	$3(M^2+5M+N-1)+K^2(2M+6)$ $+K(6M+6N_u+M_b+4)$	$2M^2+14M+2N+5$ $+K^2(2M+6)+K(6M+6)$	$2K^2+3K+1$	$(K^2+K)/2$
DCD-It-MC-C-ARH	$3(M^2-M+N)$ $+N_c(K^2(2M+6)+20M-4)$ $+N_cK(6M+6N_u+M_b+4K)$	$2(M^2-M+N)+1$ $+N_c(K^2(2M+4)$ $+K(6M+24)+4)$	$N_c(2K^2+3K+1)$	$N_c(K^2+K)/2$

to identify them. The faulty sensors are randomly selected, and all the techniques considered in the simulation use $\mathbf{R}_{\mathrm{R}}(n)$ to compute the beamformer. In the second simulation, we use the same conditions as before, but we assume that we are unable to identify the faulty elements. For such situation, we show that the proposed techniques are robust to errors in the detection of faulty elements.

We present the SINR performance of C-ARH, MC-C-ARH, It-C-ARH, It-MC-C-ARH, DCD-It-C-ARH and DCD-It-MC-C-ARH algorithms, and we compare these techniques to the $O(M^3)$ RCB of [13], the RLS algorithm [1] without the addition of regularization, and to a method using a diagonal loading (DL) added to $\mathbf{R}_{\mathrm{R}}(n)$. We use the approach of [11] to regularize $\mathbf{R}_{\mathrm{R}}(n)$ by adding to the matrix the diagonal loading $10\sigma_\eta^2\mathbf{I}$, assuming exact knowledge of σ_η. The solution to $(\mathbf{R}_{\mathrm{R}}(n)+10\sigma_\eta^2\mathbf{I})\mathbf{x}(n)=\mathbf{b}_{\mathrm{R}}$ is then used to compute the beamformer, which can be implemented, for instance, using RLS, at cost of $O(M^2)$. We present the SINR performance and the number of homotopy steps used by the proposed techniques, which expresses the maximum dimension of the system of equations solved by the homotopy-based algorithms. To determine the faulty sensors in the first simulation, we use the energy detection method, as presented in Section III-A3. Matrix $\mathbf{E}$ is estimated only during the first 100 snapshots. Afterwards, we freeze the last estimated value of $\mathbf{E}$ for the remaining snapshots.

The power of each interferer is 10 times the power of the signal of interest, $\sigma_{\mathrm{d}}^2 = 1$. The SNR is 8dB, and the noise is a zero-mean Gaussian i.i.d. sequence. The signals produced by the sources are zero-mean binary sequences of -1 and 1, and we use eq. (31) to iteratively estimate $\mathbf{R}_t$. The forgetting factor is given by $\nu = 1 - 2^{-6} \simeq 0.9844$ and $\mathbf{R}(0) = 10^{-3}\mathbf{I}$. The algorithms are adjusted in the first simulation to achieve the maximum SINR in the steady-state. We keep the same values for the parameters in the second simulation to study the effect of the wrong identification of faulty sensors to the beamformer computation. The figures are obtained with the mean of 200 realizations.

A. Regularization when the faulty-sensor detector is applied

For the simulation presented in this section, we assume that there are 5 faulty sensors, and that the faulty-sensor detector is applied to identify and exclude them from the computation of the beamformer. To adjust C-ARH and MC-C-ARH, we use $\tau = 0.32$ and define three candidates for the MC-C-ARH algorithm, given by $\mathbf{\Lambda} = [0.6\ 0.8\ 1]$. The iterative algorithms It-C-ARH and It-MC-C-ARH use a stopping parameter $\tau_{\mathrm{It}} = 2$, while DCD-It-C-ARH and DCD-It-MC-C-ARH use $\tau_{\mathrm{DCD}} = 5$, $N_u = 8$, $M_b = 16$ bits and $H = 2$. The multi-candidate iterative algorithms use the same set of candidates as selected for MC-C-ARH, and the constraint of the RCB is given by $\epsilon = 0.1$. Figure 1 presents the SINR performance and the number of homotopy iterations required by the proposed techniques to compute the MVDR beamformer.

From Figure 1, one can see that the C-ARH algorithm and the DL approach have almost the same SINR performance after 250 snapshots, and that both algorithms are outperformed by the MC-C-ARH algorithm after 200 snapshots. The proposed iterative algorithms outperform the other homotopy-based techniques, and the highest SINR is achieved by the DCD-based algorithms. The iterative algorithms outperform their non-iterative counterparts with a lower number of homotopy iterations, which reduces the number of computations. For instance, consider the average number of iterations used by C-ARH (which is 56), It-C-ARH (3) and DCD-It-C-ARH (1.5), after they achieve the steady-state. Using these values of K in the equations presented in Tables VIII, IX and X, we compare the number of multiplications used by these methods. The C-ARH algorithm requires 400471 multiplications plus the cost to solve 56 systems of equations of sizes from 1×1 to 56×56. The It-C-ARH algorithm uses 26211 multiplications and solves only 3 systems of equations. The DCD-It-C-ARH algorithm, the least costly of them, uses only 8386 multiplications. No additional multiplications are required to solve the systems of equations, since

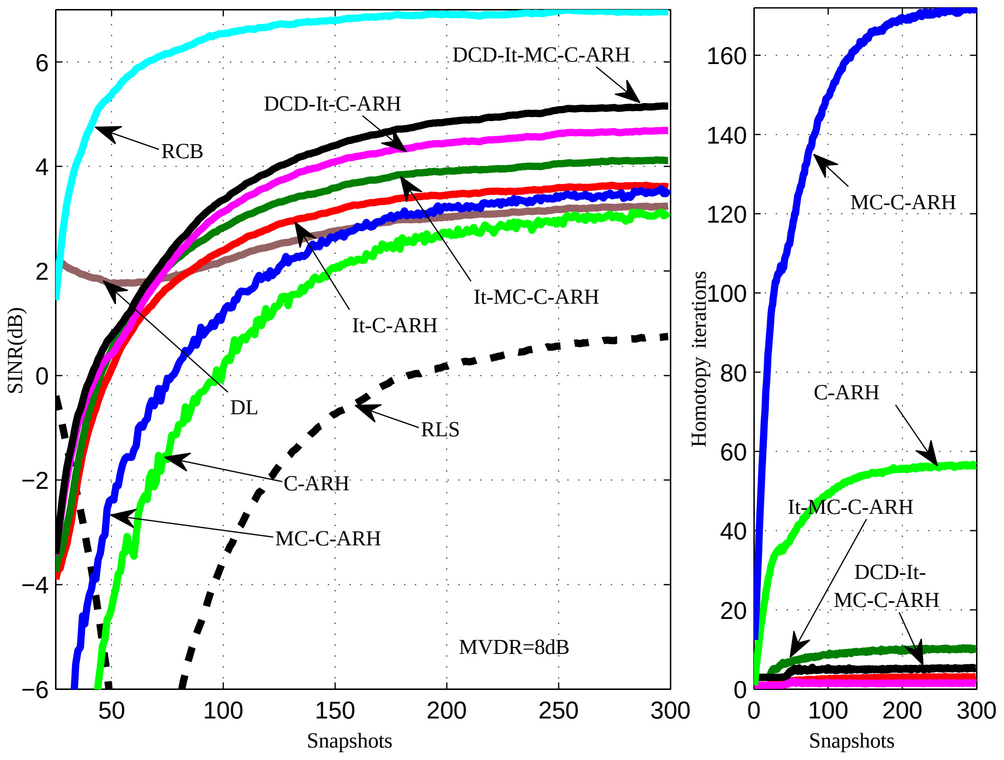

Fig. 1. SINR performance (left) and average number of homotopy iterations per snapshot (right), when there are 5 faulty sensors. Mean of 200 realizations. The red and magenta curves in the right figure are not identified to avoid overlap of arrows. The red curve is the average number of homotopy iterations required by DCD-It-C-ARH, and it is equal to 1.5 iterations per snapshot. The magenta curve is the average number of iterations required by It-C-ARH, and it corresponds to 2.5 iterations per snapshot.

the DCD algorithm does not require multiplications.

The DCD-based iterative algorithms are only outperformed by the RCB, which achieves an SINR performance 1.5dB higher than the DCD-It-MC-C-ARH. Notice that the RCB has a computational cost of $O(M^3)$, while the DCD techniques are $O(M^2)$. Checking Table X, one can verify that the dominant terms in computational complexity of DCD-It-C-ARH are K^2M and M^2. Since, a priori, the maximum number of homotopy iterations could be as high as M, the cost to implement the techniques would be cubic. However, as we can see in the simulation, the iterative algorithms require a much lower number of homotopy iterations (3 homotopy iterations for It-C-ARH and 1.5 iterations for DCD-It-C-ARH), and $K \ll M$. For this situation, the iterative algorithms require a number of computations which is quadratic in M. The trade-off for the less costly approach is a performance loss of about 1.5dB in this scenario.

B. Robustness to errors in the identification of faulty sensors in the array

In this simulation, we show that the iterative techniques are robust when the faulty sensors cannot be identified and excluded from the array. We assume that the array has 5 faulty sensors, but the detector is unable to identify them. For this situation, we cannot reduce the dimension of the correlation matrix, and the SINR performance of the algorithms might be affected. We maintain the value of the parameters used in Section VII-A, and we show that the iterative DCD-based algorithms presents almost the same performance. Figure 2 show the results.

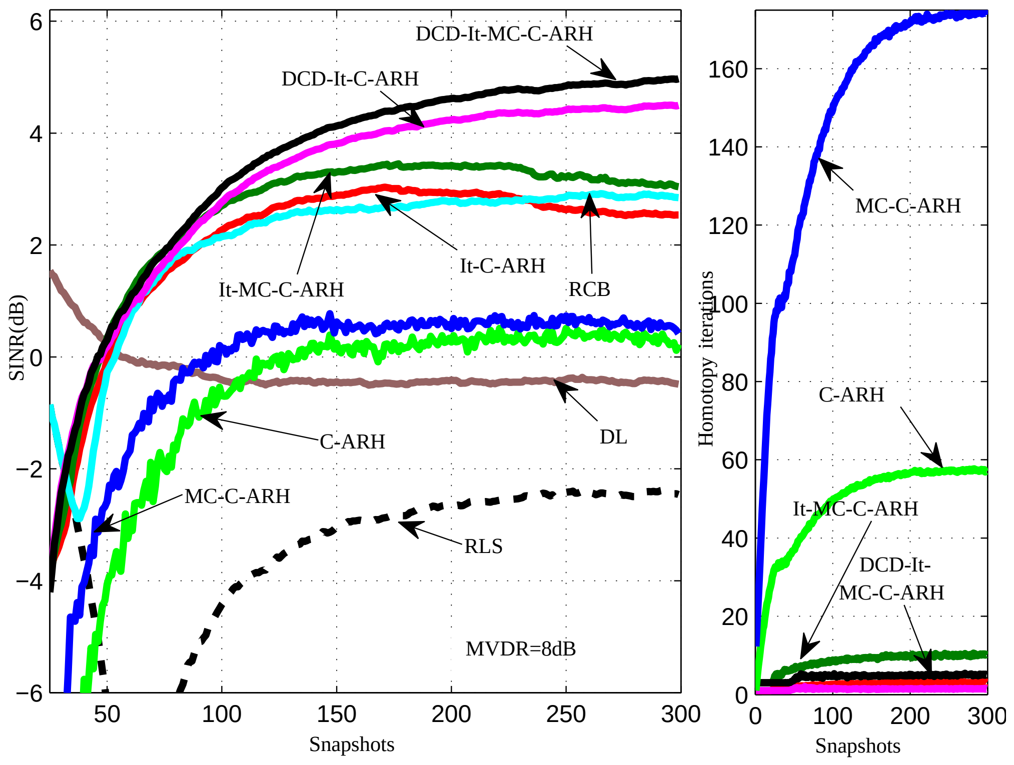

Fig. 2. SINR performance (left) and average number of homotopy iterations per snapshot (right), when there are 5 faulty sensors, but they are not identified by the detector. Mean of 200 realizations. The red and magenta curves in the right figure are not identified to avoid overlap of arrows. The red curve is the average number of homotopy iterations required by DCD-It-C-ARH, and it is equal to 1.5 iterations per snapshot. The magenta curve is the average number of iterations required by It-C-ARH, and it corresponds to 2.8 iterations per snapshot.

In this scenario, the performance degrades for all the algorithms, but the techniques using the DCD are less affected by the faulty sensors. The better performance of the DCD algorithms is explained by the very low number of homotopy iterations used by these techniques. To understand this, recall the algorithm presented in Table I. To add an element to the support set, the algorithm selects the columns of the matrix which are most correlated to the residue. When the algorithm starts, it first adds to the support the highly correlated features of the matrix, for which the selection is not affected by the noise introduced by the faulty sensors. However, after a few steps of the C-ARH, the columns less correlated to the residue and the columns introduced by the faulty sensors can be mistaken, and the algorithm can add to the support wrong elements, leading to poor performance. Since the DCD iterative algorithms use a very low number of iterations, they only add to the support the most correlated elements, easier to identify, reducing the error and improving the SINR performance.

VIII. Conclusion

In this paper, we have presented new beamforming algorithms based on the ℓ_1-norm regularized C-ARH technique. Our iterative algorithms have low computational complexity and are robust against ill-condition in the input autocorrelation matrix, which arises when the number of interferers is small compared to the number of sensors. We compared our methods with the diagonal loading approach of [11] and the RCB of [13], robust beamformers, in two situations: one in which faulty sensors are removed from the equations, and another in which faulty sensors are not correctly detected. In the first situation, our iterative algorithms perform better than C-ARH, MC-C-ARH and the diagonal loading approach of [11], but they are outperformed by the RCB of [13]. In the second scenario, when the contribution of the faulty sensors is not removed from the sample correlation matrix and the algorithms keep the same parameters as in the first simulation, the DCD-based iterative methods are robust and outperform the other techniques. We have also shown that the iterative methods can be implemented with a reduced cost when compared to the RCB. The trade-off between SINR performance and computational complexity favors the iterative techniques.

We have presented the computational complexity of the proposed techniques, as a function of the number of sensors and the support size of C-ARH. Since the iterative algorithms require fewer homotopy iterations per snapshot than the C-ARH and MC-C-ARH algorithms, they are less costly to implement. For the DCD-based techniques, the number of computations is much lower, since many multiplications are replaced by bit-shifts and additions. Since the number of homotopy iterations is also low, the iterative algorithms can be implemented with a reduced cost, proportional to M^2, where M is the number of sensors in the array, while the RCB is cubic in M.

References

[1] H.L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation and Modulation Theory*, Wiley, 2002.

[2] S.S. Haykin, *Adaptive Filter Theory, 4ed.*, Prentice Hall, 2002.

[3] V.H. Nascimento and M.T.M. Silva, “Adaptive filters,” in *Academic Press Library in Signal Processing:*, Rama Chellappa and Sergios Theodoridis, Eds., vol. 1, Signal Processing Theory and Machine Learning, pp. 619—761. Chennai: Academic Press, 2014.

[4] Z. Yang, R.C. de Lamare, and X. Li, “L1-regularized STAP algorithms with a generalized sidelobe canceler architecture for airborne radar,” *IEEE Trans. Signal Processing*, vol. 60, no. 2, pp. 674–686, 2012.

[5] Z. Yang, R.C. de Lamare, and X. Li, “Sparsity-aware space-time adaptive processing algorithms with L1-norm regularisation for airborne radar,” *IET Signal Processing*, vol. 6, no. 5, pp. 413–423, 2012.

[6] Z. Yang, R.C. de Lamare, and X. Li, “L1-regularized STAP algorithm with a generalized sidelobe canceler architecture for airborne radar,” in *IEEE Statistical Signal Process. Workshop (SSP)*, 2011, pp. 329–332.

[7] J.R. Guerci and J.S. Bergin, “Principal components, covariance matrix tapers, and the subspace leakage problem,” *IEEE Trans. on Aerospace and Electronic Systems*, vol. 38, no. 1, pp. 152–162, Jan 2002.

[8] J.A. Bucklew and W.A. Sethares, “Convergence of a Class of Decentralized Beamforming Algorithms,” *IEEE Trans. on Signal Processing*, vol. 56, no. 6, pp. 2280–2288, June 2008.

[9] C. Lin, V.V. Veeravalli, and S.P. Meyn, “A Random Search Framework for Convergence Analysis of Distributed Beamforming With Feedback,” *IEEE Trans. on Information Theory*, vol. 56, no. 12, pp. 6133–6141, Dec 2010.

[10] S. Song, J.S. Thompson, P.-J. Chung, and P. M. Grant, “Exploiting Negative Feedback Information for One-Bit Feedback Beamforming Algorithm,” *IEEE Trans. on Wireless Communications*, vol. 11, no. 2, pp. 516–525, Feb. 2012.

[11] H. Cox, R.M. Zeskind, and M.M. Owen, “Robust Adaptive Beamforming,” *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 35, no. 10, pp. 1365–1376, Oct 1987.

[12] L. Du, J. Li, and P. Stoica, “Fully Automatic Computation of Diagonal Loading Levels for Robust Adaptive Beamforming,” *IEEE Trans.on Aerospace and Electronic Systems*, vol. 46, no. 1, pp. 449–458, Jan 2010.

[13] J. Li, P. Stoica, and Z. Wang, “On robust Capon beamforming and diagonal loading,” *IEEE Trans. on Signal Processing*, vol. 51, no. 7, pp. 1702–1715, July 2003.

[14] J. Li and P. Stoica, *Robust adaptive beamforming*, Wiley Online Library, 2006.

[15] S.A. Vorobyov, A.B. Gershman, and Z.-Q. Luo, “Robust adaptive beamforming using worst-case performance optimization: a solution to the signal mismatch problem,” *IEEE Trans. on Signal Processing*, vol. 51, no. 2, pp. 313–324, Feb 2003.

[16] D.L. Donoho and Y. Tsaig, “Fast solution of L1-norm minimization problems when the solution may be sparse,” *IEEE Trans. Inform. Theory*, vol. 54, no. 11, pp. 4789–4812, 2008.

[17] M.S. Asif and J. Romberg, “Fast and accurate algorithms for re-weighted L1-norm minimization,” *IEEE Trans. Signal Processing*, vol. 61, no. 23, pp. 5905–5916, Dec 2013.

[18] C. Qi, X. Wang, and L. Wu, “Underwater acoustic channel estimation based on sparse recovery algorithms,” *IET Signal Processing*, vol. 5, no. 8, pp. 739–747, Dec. 2011.

[19] F.G. Almeida Neto, V.H. Nascimento, Y.V. Zakharov, and R.C. de Lamare, “Adaptive re-weighting homotopy for sparse beamforming,” in *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, Sept. 2014, pp. 1–5.

[20] Y.V. Zakharov and T.C. Tozer, “Multiplication-free iterative algorithm for LS problem,” *Electronics Letters*, vol. 40, no. 9, pp. 567–569, 2004.

[21] C.D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, USA, 2000.

[22] S.M. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*, Prentice Hall, 1998.

[23] K.T. Wong, Y.I. Wu, Y.-S. Hsu, and Y. Song, "A Lower Bound of DOA-Estimates by an Array Randomly Subject to Sensor-Breakdown," *IEEE Sensors Journal*, vol. 12, no. 5, pp. 911–913, May 2012.

[24] J. Yang and Y. Zhang, "Alternating direction algorithms for L1-problems in compressive sensing," *SIAM Journal on Scientific Computing*, vol. 33, no. 1, pp. 250–278, 2011.

[25] S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Trans. Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[26] E. Van Den Berg and M.P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[27] Y.V. Zakharov, G.P. White, and J. Liu, "Low-complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Trans. on Signal Processing*, vol. 56, no. 7, pp. 3150–3161, 2008.

[28] Y.V. Zakharov, "Low-complexity implementation of the affine projection algorithm," *IEEE Signal Processing Letters*, vol. 15, pp. 557–560, 2008.

[29] J. Liu, B. Weaver, Y. Zakharov, and G. White, "An FPGA-based MVDR beamformer using dichotomous coordinate descent iterations," in *IEEE International Conference on Communications*, 2007, pp. 2551–2556.

[30] Y. Zakharov and V.H. Nascimento, "Homotopy algorithm using dichotomous coordinate descent iterations for sparse recovery," in *Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2012, pp. 820–824.

[31] J. Liu, Y.V. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 56, no. 11, pp. 2425–2438, 2009.