A DISTRIBUTED ADAPTIVE ALGORITHM FOR NON-SMOOTH SPATIAL FILTERING PROBLEMS

Charles Hovine and *Alexander Bertrand*

KU Leuven, Department of Electrical Engineering (ESAT)
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics
KU Leuven Institute for Artificial Intelligence (Leuven.AI)
Leuven, Belgium
{charles.hovine, alexander.bertrand}@esat.kuleuven.be

ABSTRACT

Computing the optimal solution to a spatial filtering problems in a Wireless Sensor Network can incur large bandwidth and computational requirements if an approach relying on data centralization is used. The so-called distributed adaptive signal fusion (DASF) algorithm solves this problem by having the nodes collaboratively solve low-dimensional versions of the original optimization problem, relying solely on the exchange of compressed views of the sensor data between the nodes. However, the DASF algorithm has only been shown to converge for filtering problems that can be expressed as smooth optimization problems. In this paper, we explore an extension of the DASF algorithm to a family of non-smooth spatial filtering problems, allowing the addition of non-smooth regularizers to the optimization problem, which could for example be used to perform node selection, and eliminate nodes not contributing to the filter objective, therefore further reducing communication costs. We provide a convergence proof of the non-smooth DASF algorithm and validate its convergence via simulations in both a static and adaptive setting.

Index Terms— Adaptive spatial filtering, Wireless Sensor Networks, Non-smooth optimization, Distributed signal processing.

1. INTRODUCTION

A spatial filtering problem usually consists in finding the linear combination of a set of signals that is optimal with regards to some criterion, and can therefore be expressed as the solution of an optimization problem. Common examples include principal components analysis [1], canonical correlation analysis [2], MAX-SNR beamforming, multichannel Wiener filtering [3] and common spatial patterns [4].

In the case of Wireless Sensor Networks (WSNs), where several sensing nodes communicate via wireless links, signals are often only short-term stationary, with statistics drifting over time. Being able to *adaptively* compute filters therefore becomes an important requirement. The classical approach to computing spatial filters in WSNs consists in designating a particular node as the *fusion center* (FC), which will collect all the raw data and perform the filter computation centrally [5]. This approach is however not ideal, as the bandwidth and computational power required at the FC scales poorly with the number of both nodes and signals. Additionally, the FC constitutes a single point of failure, which can be problematic for many deployment scenarios. An alternative approach consists in solving the filtering problem in a distributed fashion, by sharing the work across the sensor nodes.

The DASF algorithm [6] is a framework for solving adaptive spatial filtering problems in a distributed fashion. Instead of sharing their raw observations, the nodes share efficiently crafted compressed views of their sensor data, which are then used to locally solve low-dimensional versions of the original optimization problem at each node. In addition, the sensor signals' statistics are allowed to change during the course of the algorithm, such that the optimal solution can be tracked adaptively.

The convergence and optimality of DASF in the case of filtering problem expressible as smooth optimization problems, has been studied in [7], but the applicability of the algorithm to non-smooth problems is still unknown. In this paper, we show the convergence and optimality of the algorithm for a family of non-smooth, and possibly non-convex optimization problems. In addition to allowing the algorithm to be applied to well-known non-smooth problems such as sparse signal recovery and compressed sensing [8,9], it allows the use of non-smooth sparsity-promoting regularizers. In the context of WSNs, such regularizers can for example be used to perform channel selection, and hence reduce both bandwidth and computational stress on the sensor nodes.

2. PROBLEM STATEMENT

We consider a network consisting of K sensor nodes, where each node k collects discrete observations of an M_k-channel signal $\boldsymbol{y}_k(t)$. We denote $\boldsymbol{y}(t) = [\boldsymbol{y}_1^T(t), \ldots, \boldsymbol{y}_K^T(t)]^T$ the network-wide multichannel sensor signal, where each observation is an element of $\mathbb{R}^M$ with $M = \sum_k M_k$.Our goal is to design a network-wide spatial filter $\boldsymbol{X} \in \mathbb{R}^{M \times Q}$ which fuses all the channels of $\boldsymbol{y}(t)$ into Q output channels that satisfy a certain optimality criterion, which in generic form can be written as

$$\begin{aligned} \boldsymbol{X}^\star(t) \in \underset{\boldsymbol{X}}{\operatorname{argmin}}\, & f(\boldsymbol{X}^T\boldsymbol{y}(t), \boldsymbol{X}^T\boldsymbol{B}) + g(\boldsymbol{X}^T\boldsymbol{\Gamma}) \\ \text{s.t.} \quad & [\boldsymbol{X}_k^T\boldsymbol{y}_k(t), \boldsymbol{X}_k^T\boldsymbol{B}_k] \in \mathcal{X}_k \quad \forall k. \end{aligned} \tag{1}$$

where each block $\boldsymbol{X}_k \in \mathbb{R}^{M_k \times Q}$ is defined according to the partitioning $\boldsymbol{X} = [\boldsymbol{X}_1^T, \cdots, \boldsymbol{X}_K^T]^T$. $\boldsymbol{\Gamma} = \text{BlkDiag}(\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_K)$,

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 802895), the FWO (Research Foundation Flanders) for project G081722N and from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

where BlkDiag$(\cdot)$ is the operator producing a block diagonal matrix whose blocks correspond to the operator's ordered arguments, with $\boldsymbol{\Gamma}_k \in \mathbb{R}^{M_k \times L_k}$ a constant (time-independent) data matrix, along with $\boldsymbol{B} = [\boldsymbol{B}_1^T, \ldots, \boldsymbol{B}_K]^T \in \mathbb{R}^{M \times D}$. The term involving f is a smooth function of $\boldsymbol{X}$ (i.e. differentiable with continuous gradient) and the term involving g is a convex, possibly non-smooth, function of $\boldsymbol{X}$. The other particular characteristics, of f, g and $\mathcal{X}_k$ are not immediately relevant and will thus be later described in Section 4, along with our convergence analysis. Additionally, we require that there exist some functions g_k, such that the non-smooth term g can be separated as

$$g(\boldsymbol{X}^T\boldsymbol{\Gamma}) = \sum_k g_k(\boldsymbol{X}_k^T\boldsymbol{\Gamma}_k). \tag{2}$$

which holds for, e.g., the l_1-norm. Typically, f is a cost function depending on the second order statistics of $\boldsymbol{y}(t)$ (e.g. the covariance matrix), g is a regularizing term promoting certain desirable properties of the solution (e.g. a sparsity inducing norm), and the constraint sets $\mathcal{X}_k$ encode some hard limits on the filter, such as limiting the maximum output power, or requiring the per-node filters to have uncorrelated outputs. Note that, as it is defined, g cannot be an indicator function, and that all the constraints must therefore be encoded in $\mathcal{X}_k$[1]. As an example, a possibility for f is

$$f(\boldsymbol{X}^T\boldsymbol{y}(t)) = \mathbb{E}\left\{\left\|\boldsymbol{X}^T\boldsymbol{y}(t) - \boldsymbol{d}(t)\right\|_2^2\right\}, \tag{3}$$

where $\mathbb{E}\{\cdot\}$ denotes the expectation operator, and both $\boldsymbol{y}(t)$ and $\boldsymbol{d}(t)$ are random signals. Finally, we emphasize that the per-node (i.e. per-block) constraints in (1) are stricter than in the original DASF problem setting in [6], which allowed for coupling constraints between the different blocks of variables, i.e. between the variables of different nodes. Note that although (1) is only allowed to depend on $\boldsymbol{y}(t)$, it does not preclude the existence of multiple sets of signals. Indeed, by imposing the proper structure on both f and $\boldsymbol{y}(t)$, we can describe problems depending on multiple sets of data. We may for example wish to solve problems of the form

$$\min_{\boldsymbol{X}_1,\boldsymbol{X}_2} f(\boldsymbol{X}_1^T\boldsymbol{u}(t), \boldsymbol{X}_2^T\boldsymbol{v}(t)) \tag{4}$$

which is possible in the framework of (1) by defining $\boldsymbol{X} = [\boldsymbol{X}_1^T, \boldsymbol{X}_2^T]^T$ and $\boldsymbol{y}(t) = \text{BlkDiag}(\boldsymbol{u}(t), \boldsymbol{v}(t))$.

For the rest of this paper, we will omit the time index t of $\boldsymbol{X}^\star$, as we assume for mathematical tractability that $\boldsymbol{y}(t)$ is short-term stationary, and hence that the set of optimal filters varies slowly with time (i.e. $\boldsymbol{X}^\star(t) \approx \boldsymbol{X}^\star(t+\tau)$ for small enough τ). Furthermore, we do not have access to the data-generating process $\boldsymbol{y}(t)$, but only to consecutive realizations of $\boldsymbol{y}(t)$, which, under the assumption of ergodicity, can be used to obtain an estimate of the statistics implicitly involved in (1). In an actual implementation, one would evaluate/optimize (1) based on estimated statistics of $\boldsymbol{y}(t)$, i.e. $\boldsymbol{y}(t)$ would need to be replaced by a matrix of discrete samples $\boldsymbol{Y}(t)$ centered around t and the expectation in (3) would have to be approximated with a sample average.

Our objective is to solve (1) in a bandwidth-efficient manner. The optimization procedure therefore cannot rely on a fusion center to collect samples of the full $\boldsymbol{y}(t)$ vector to estimate inter-channel statistics, as this would incur significant communication costs. Indeed, in an adaptive setting where the data is allowed to change at every iteration, every new sample would need to be collected by the FC. Instead, we propose a fully distributed procedure that relies on the nodes sharing linearly compressed views of their observations with one another, and locally solving lower dimensional versions of (1) at different times instance, depending only on the compressed observations received from other nodes. By exploiting the short-term stationarity of $\boldsymbol{y}(t)$, each iteration of the algorithm can be performed over a different time-window, thereby behaving like an adaptive filter in which the filter coefficients are adjusted every time a new (block of) sample(s) is collected.

3. NON-SMOOTH DASF

In order to ease the exposition of the algorithm, we limit our description to the specific case of fully-connected networks. A generalization to arbitrary topologies can be done in a similar fashion as for the original DASF algorithm [6,7]. Furthermore, without loss of generality, we ignore the deterministic argument $\boldsymbol{X}^T\boldsymbol{B}$ as it adds a lot of clutter in the equations, while it is largely treated in the same way as the $\boldsymbol{X}^T\boldsymbol{y}(t)$ argument (we again refer to [6,7] for further details).

In our algorithm, each node k is responsible for updating its own block $\boldsymbol{X}_k \in \mathbb{R}^{M_k \times Q}$ of $\boldsymbol{X} = [\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_K^T]^T$, corresponding to its own locally observed data $\boldsymbol{y}_k(t)$. Let us denote $\boldsymbol{X}^i$ the algorithm's estimate of the solution of (1) at iteration i. We emphasize that each iteration is performed on a different block of N samples of $\boldsymbol{y}(t)$, i.e., the update from $\boldsymbol{X}^i$ to $\boldsymbol{X}^{i+1}$ will be based on the observations of $\boldsymbol{y}(t)$ at sample times $t = (i-1)N, ..., iN-1$.

Let us consider problem (1) with the additional linear constraints

$$\boldsymbol{X}_k \in \mathcal{C}(\boldsymbol{X}_k^i) \quad \forall k \neq q, \tag{5}$$

with $\mathcal{C}(\cdot)$ denoting the column space of its argument and where q is some arbitrary node, which we will refer to as the updating node. By introducing the parametrization $\boldsymbol{X}_k = \boldsymbol{X}_k^i\boldsymbol{G}_k$ for $k \neq q$ corresponding to the linear subspace constraints (5), and defining the compressed signals of node k as $\boldsymbol{z}_k^i(t) \triangleq \boldsymbol{X}_k^{iT}\boldsymbol{y}_k(t)$ and $\boldsymbol{F}_k^i \triangleq \boldsymbol{X}_k^{iT}\boldsymbol{\Gamma}_k$, the new problem (1) equipped with (5) can be reformulated as

$$\begin{aligned}
&\bar{\boldsymbol{X}}^\star \in \operatorname*{argmin}_{\bar{\boldsymbol{X}}} f(\bar{\boldsymbol{X}}^T\boldsymbol{z}^i(t)) + g(\bar{\boldsymbol{X}}^T\boldsymbol{F}^i) && \text{(6a)}\\
&\text{s.t.} \quad \boldsymbol{G}_k^T\boldsymbol{z}_k^i(t) \in \mathcal{X}_k \quad \forall k \neq q && \text{(6b)}\\
&\qquad \boldsymbol{X}_q^T\boldsymbol{y}_q(t) \in \mathcal{X}_q && \text{(6c)}\\
&\qquad \bar{\boldsymbol{X}} = [\boldsymbol{G}_1^T, \ldots, \boldsymbol{X}_q^T, \ldots, \boldsymbol{G}_K^T]^T && \text{(6d)}\\
&\qquad \boldsymbol{z}^i(t) = [\boldsymbol{z}_1^T(t), \ldots, \boldsymbol{y}_q^T(t), \ldots, \boldsymbol{z}_K^T(t)]^T && \text{(6e)}\\
&\qquad \boldsymbol{F}^i = \text{BlkDiag}(\boldsymbol{X}_1^{iT}\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_q, \ldots, \boldsymbol{X}_K^{iT}\boldsymbol{\Gamma}_K). && \text{(6f)}
\end{aligned}$$

We can see that by collecting the compressed observations of every other node, some node q can compute a solution of the *local* problem (6), and equivalently of the linearly constrained *global* problem (1) with the addition of the constraints (5). We use the term *compressed* observations since, if $Q < M_k$, $\boldsymbol{z}_k^i(t)$ will have a lower dimension than $\boldsymbol{y}_k(t)$ and can therefore be more efficiently transmitted than the raw data. As $\boldsymbol{\Gamma}$ is assumed static, it only needs to be shared once, and only the $\boldsymbol{X}_k^i$ will need to be exchanged, unless $L < M_k$, in which case it is more efficient to share $\boldsymbol{F}_k^i$.

Note that the global and local problems (1) and (6) have the same general structure but with a different dimension, therefore if a solver exists for the global problem, it can also be used to solve the local problems. In other words, if we denote by $\mathbb{P}(\boldsymbol{y}(t), \boldsymbol{\Gamma})$ a particular instance of problem (1), solving (6) is equivalent to solving $\mathbb{P}(\boldsymbol{z}^i(t), \boldsymbol{F}^i)$.

Our iterative procedure consists in updating $\boldsymbol{X}^i$ by iteratively solving (6), each time selecting a new node q to act as the "updat-

[1]This does not restrict the set of allowable problems, but allows us to simplify the notation used in Section 4.

Algorithm 1: NS-DASF algorithm.

```
begin
    $i \leftarrow 0$, $q \leftarrow 1$, Randomly initialize $\boldsymbol{X}^0$
    loop
        for $k \in \{1, \dots, K\} \setminus \{q\}$ do
            At node $k$
                Collect a new batch of $N$ samples of $\boldsymbol{y}_k(t)$
                  and send the compressed samples
                  $\boldsymbol{z}_k^i(t) = \boldsymbol{X}_k^{iT}\boldsymbol{y}_k(t)$ along with
                  $\boldsymbol{F}_k^i = \boldsymbol{X}_k^{iT}\boldsymbol{\Gamma}_k$ to node $q$.
        At node $q$
            Obtain $\bar{\boldsymbol{X}}^\star$ by solving the local problem (6)
              using only the compressed data $\boldsymbol{z}^i(t)$ and $\boldsymbol{\Gamma}^i$.
              If the solution is not unique, select the one
              minimizing $\left\|\bar{\boldsymbol{X}}^\star - \bar{\boldsymbol{X}}^{i-1}\right\|_F$.
            Extract $\boldsymbol{X}_q^\star$ and the $\boldsymbol{G}_k^\star$'s from $\bar{\boldsymbol{X}}^\star$ according
              to the partitioning (6d).
            $\boldsymbol{X}_q^{i+1} \leftarrow \boldsymbol{X}_q^\star$
            for $k \in \mathcal{K} \setminus \{q\}$ do
                Send $\boldsymbol{G}_k^\star$ to node $k$.
                At node $k$
                    $\boldsymbol{X}_k^{i+1} \leftarrow \boldsymbol{X}_k^i\boldsymbol{G}_k^\star$
        $i \leftarrow i+1$, $q \leftarrow (q \mod K) + 1$
```

ing node" in a round-robin fashion. Formally, the procedure is as follows:

1. *Data collection:* Every node collects discrete N new observations of $\boldsymbol{y}_k(t)$.
2. *Aggregation:* Every node except the updating node q, computes its compressed data $\boldsymbol{z}_k^i(t)$ and $\boldsymbol{F}_k^i$ and transmits the corresponding N compressed samples to the updating node q.
3. *Local solution:* Based on the received compressed samples of $\boldsymbol{z}_k^i(t)$, and its own data $\boldsymbol{y}_q(t)$, the updating node q can estimate the signal statistics involved in (6) and solve it using any solver for $\mathbb{P}(\cdot,\cdot)$. It then updates its local block as $\boldsymbol{X}_q^{i+1} = \bar{\boldsymbol{X}}_q^\star$, and extracts the optimal update matrices $\boldsymbol{G}_k^\star$ from $\bar{\boldsymbol{X}}^\star$ using the partitioning (6d)[2].
4. *Solution update:* The updating node transmits the update matrices $\boldsymbol{G}_k^\star$ to their corresponding nodes. Each node except the updating node updates its block of the estimate of the solution as $\boldsymbol{X}_k^{i+1} = \boldsymbol{X}_k^i\boldsymbol{G}_k^\star$.

The full description of the algorithm is given by Algorithm 1, which we refer to as non-smooth DASF (NS-DASF).

4. CONVERGENCE

One can gain intuition about the algorithm's convergence by noting that $\boldsymbol{X}^i$ is always in the feasible set of problem (6), as it satisfies (5) trivially, ensuring a monotonic decrease of the objective. We will start by showing that fixed points of Algorithm 1 (i.e. points $\boldsymbol{X}^*$ such that if $\boldsymbol{X}^0 = \boldsymbol{X}^*$, then $(\boldsymbol{X}^i)_{i\in\mathbb{N}} = (\boldsymbol{X}^*)_{i\in\mathbb{N}}$) are stationary points of problem (1), and then reuse one of the result of [7] to show convergence to such a point. Let us first define

$$\begin{aligned} p(\boldsymbol{X}) &\triangleq f(\boldsymbol{X}^T\boldsymbol{y}(t)) \\ q(\boldsymbol{X}) &\triangleq g(\boldsymbol{X}^T\boldsymbol{\Gamma}) \\ q_k(\boldsymbol{X}_k) &\triangleq g_k(\boldsymbol{X}_k^T\boldsymbol{\Gamma}_k) \end{aligned} \quad \begin{aligned} \mathcal{D}_k &\triangleq \{\boldsymbol{X}_k \mid \boldsymbol{X}_k^T\boldsymbol{y}_k(t) \in \mathcal{X}_k\} \\ \mathcal{D} &\triangleq \mathcal{D}_1 \times \cdots \times \mathcal{D}_K \end{aligned} \tag{7}$$

where $\times$ denotes the cartesian product between sets. We assume that $p : \mathbb{R}^{M\times Q} \to \mathbb{R}$ is a smooth function with compact sublevel sets, $q : \mathbb{R}^{M\times Q} \to \mathbb{R}$ is a proper, lower semicontinuous and convex function, and $\mathcal{D}$ is a closed set in $\mathbb{R}^{M\times Q}$. We wish to show that the fixed points of the algorithm are also stationary points of problem (1), that is feasible points $\boldsymbol{X}^\star$ such that [10]

$$0 \in \nabla p(\boldsymbol{X}^\star) + \partial q(\boldsymbol{X}^\star) + N_{\mathcal{D}}(\boldsymbol{X}^\star), \tag{8}$$

where $\partial q(\cdot)$ denotes the set of subgradients of q at a particular point and $N_{(\cdot)}(\cdot)$ denotes the normal cone at a particular point of a set. The sum between sets must be interpreted as a Minkowski sum[3]. Equation (8) generalizes the well-known Karush-Kuhn-Tucker (KKT) conditions [11,12] to the case of non-smooth functions[4] (it therefore reduces to the KKT conditions in the smooth case). It merely gives necessary conditions for a feasible point to be a solution of (1), but the condition is also sufficient in the case of convex instances of the problem [10]. Intuitively, those points are such that all directional derivatives pointing inside the feasible set are positive (i.e. there is no feasible descent direction at that point, see [10, 14] for details).

Before stating our main result, we give an explicit expression of the normal cone corresponding to the subspace constraints (5) at a point $\boldsymbol{X}^i = \boldsymbol{X}$. We denote

$$\mathcal{L}_q(\boldsymbol{X}) \triangleq \mathcal{C}(\boldsymbol{X}_1) \times \cdots \times \mathbb{R}^{M_q\times Q} \times \cdots \times \mathcal{C}(\boldsymbol{X}_K) \tag{9}$$

the subspace constraints at node q, where $\boldsymbol{X}$ here corresponds to $\boldsymbol{X}^i$ in (5) and where $\mathbb{R}^{M_q\times Q}$ corresponds to the lack of constraints associated with node q. As the normal cone to a linear subspace is simply its orthogonal complement [14], we have

$$N_k(\boldsymbol{X}) \triangleq N_{\mathcal{L}_q(\boldsymbol{X})}(\boldsymbol{X}) = \mathcal{C}(\boldsymbol{X}_1)^\perp \times \cdots \times \{0\} \times \cdots \times \mathcal{C}(\boldsymbol{X}_K)^\perp, \tag{10}$$

where $(\cdot)^\perp$ denotes the orthogonal complement and the singleton $\{0\} = (\mathbb{R}^{M_q\times Q})^\perp$. We can now state a first result, which established the optimality of fixed points of Algorithm 1 under a mild technical condition which is akin to the well-known linear independence constraint qualification (LICQ).

Theorem 1. *Let $\boldsymbol{X}^*$ be a fixed point of Algorithm 1 and assume that the following constraint qualifications hold:*

$$N_{\mathcal{D}}(\boldsymbol{X}^*) \cap N_k(\boldsymbol{X}^*) = \{0\} \quad \forall k. \tag{11}$$

Then $\boldsymbol{X}^*$ satisfies the stationary conditions (8) *and is therefore a stationary point of problem* (1).

Proof. The qualification (11) can be viewed as a generalization of the traditional LICQ [14], and ensures that the solutions of the local problems (6) satisfy[5] the stationarity conditions of (6) (or equivalently (1) with the additional constraints (5)) [10]

$$0 \in \nabla p(\boldsymbol{X}^*) + \partial q(\boldsymbol{X}^*) + N_{\mathcal{D}}(\boldsymbol{X}^*) + N_k(\boldsymbol{X}^*) \quad \forall k, \tag{12}$$

[2]In the case where the local problem would have multiple solutions, the solution with the smallest distance to $\bar{\boldsymbol{X}}^{i-1} \triangleq [\boldsymbol{I}, \dots, \boldsymbol{X}_q^{i-1T}, \dots, \boldsymbol{I}]^T$ is selected [6]

[3]$A + B = \{a + b \mid a \in A,\ b \in B\}$.

[4] [13] contains a useful introduction to the concepts of stationnarity for non-smooth problems.

[5]This is true in part because all the properties of f, g, and $\mathcal{X}_k$ described at the beginning of Section 2 are inherited by p, q and $\mathcal{D}_k$. We omit this part of the proof due to the page limit.

or equivalently

$$\forall k, \exists z_k \in \partial q(\boldsymbol{X}^*) + N_{\mathcal{D}}(\boldsymbol{X}^*), \exists a_k \in N_k(\boldsymbol{X}^*) : \\ \nabla p(\boldsymbol{X}^*) + z_k + a_k = 0. \tag{13}$$

Let a_k^k and z_k^k denote the blocks corresponding to node k within a_k and z_k, respectively. Similarly, Let $\nabla_k p(\boldsymbol{X}^*)$ correspond to the block of the gradient associated with the block $\boldsymbol{X}_k^*$. Then $0 = z_k^k + \nabla_k p(\boldsymbol{X}^*)$, as $a_k^k \in \{0\}$ from the definition (10). From the block separability of $\mathcal{D}$ and q, we have that [10]

$$N_{\mathcal{D}} = N_{\mathcal{D}_1} \times \cdots \times N_{\mathcal{D}_K} \tag{14a}$$
$$\partial q(\boldsymbol{X}) = \partial q_1(\boldsymbol{X}) \times \cdots \times \partial q_K(\boldsymbol{X}). \tag{14b}$$

Therefore it must be that

$$z_k^k = -\nabla_k p(\boldsymbol{X}^*) \in \partial q_k(\boldsymbol{X}^*) + N_{\mathcal{D}_k}(\boldsymbol{X}^*) \quad \forall k, \tag{15}$$

and therefore $-\nabla p(\boldsymbol{X}^*) \in \partial q(\boldsymbol{X}^*) + N_{\mathcal{D}}(\boldsymbol{X}^*)$, i.e. (8) is satisfied. $\square$

In the case where the constraint set consists of smooth equality and inequality constraints, we have the following corollary.

Corollary 1. *(Proof omitted) Let $\boldsymbol{X}^*$ be a fixed point of Algorithm 1 and let $u_j^k : \mathbb{R}^{M\times Q} \to \mathbb{R}, v_l^k : \mathbb{R}^{M\times Q} \to \mathbb{R}$ be smooth functions $\forall j, l, k$. If the constraint sets $\mathcal{D}_k$ can be expressed as*

$$\mathcal{D}_k = \{\boldsymbol{X}_k \mid u_j^k(\boldsymbol{X}) = 0, v_l^k(\boldsymbol{X}) \le 0 \ \forall j, l\} \tag{16}$$

and it holds that the element of the set

$$\{\boldsymbol{X}_k^{*T} \nabla u_j^k(\boldsymbol{X}_k^*) \ \forall j;\ \boldsymbol{X}_k^{*T} \nabla v_l^k(\boldsymbol{X}_k^*) \ \forall l \in \mathbb{A}(\boldsymbol{X}^*)\}, \tag{17}$$

where $\mathbb{A}(\boldsymbol{X}^*)$ denotes the active inequality constraints at $\boldsymbol{X}^*$, are linearly independent for every k, then the qualification (11) *is satisfied and $\boldsymbol{X}^*$ is a stationary point of problem* (1).

The qualification (17) can be seen as a stricter version of the well-known LICQ, where each of the blocks of the gradients are required to be independent when projected on the column-spaces of the blocks of $\boldsymbol{X}^*$, instead of the gradients themselves.

We will now rephrase [7, Theorem 6], which asserts convergence of Algorithm 1 (the proof is the same as in [7] since it does not depend on the (non-)smoothness of the objective, except for the part associated with Theorem 1, which was proven above).

Theorem 2. *Let $(\boldsymbol{X}^i)_{i\in\mathbb{N}}$ denote a sequence of iterates generated by Algorithm 1 and assume that the solution set of* (1) *is non-empty and varies continuously*[6] *with the problem's parameters $\boldsymbol{y}(t)$ and $\boldsymbol{\Gamma}$. Furthermore, assume that the number of stationary points of* (1) *is finite (or the number of reachable stationary points of the solver of* (6) *is finite). Then $(\boldsymbol{X}^i)_{i\in\mathbb{N}}$ converges to a stationary point of problem* (1).

5. SIMULATED EXAMPLE

Consider the sparse multichannel Wiener filtering problem

$$\min_{\boldsymbol{X}} \mathbb{E}\left\{\left\|\boldsymbol{X}^T\boldsymbol{y}(t) - \boldsymbol{d}(t)\right\|_2^2\right\} + \lambda \left\|\boldsymbol{X}\right\|_1, \tag{18}$$

where $\boldsymbol{d}(t)$ is some desired Q-channel filter output signal. For the following simulations, we generated instances of the problem as $\boldsymbol{d}(t) = \boldsymbol{X}^{\star T}\boldsymbol{y}(t) + \boldsymbol{n}(t)$, where the entries of $\boldsymbol{y}(t)$ and $\boldsymbol{n}(t)$ are i.i.d. zero-mean random gaussian signals with variance 1 and 0.1, respectively. $\boldsymbol{X}^*$ is an $(\frac{M}{10})$-sparse random vector with zero-mean and unit variance gaussian entries. Furthermore, we set $\lambda = 1$, $Q = 1$,

[6]Continuity must here be understood in the context of point-to-set maps. More specifically, we require *upper hemicontinuity*. For details see [15, 16].

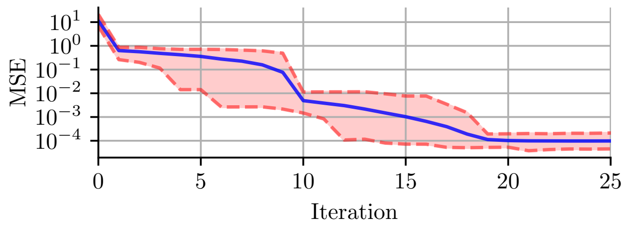

Fig. 1. Convergence of Algorithm 1 applied to problem (18). Dashed red curves correspond to the min-max convergence curves. The blue curve corresponds to the median convergence curve.

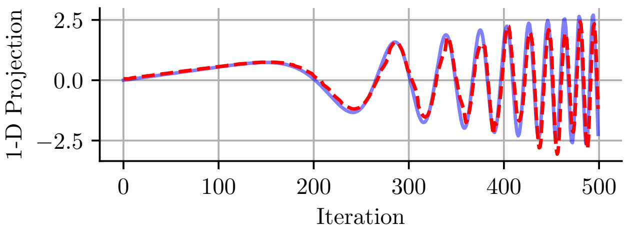

Fig. 2. Tracking of an accelerating solution over time. The blue curve corresponds to the optimal solution, the red dashed curve corresponds to NS-DASF's estimate.

$K = 10$, $M_k = 10$, and generate a 1000 samples of $\boldsymbol{y}(t)$, $\boldsymbol{d}(t)$ and $\boldsymbol{n}(t)$ for each experiment. The expectation in (18) is computed as a simple sample average. The local version of (18) was solved using Chambolle-Pock's algorithm [17], and we therefore only approximate the optimal local solution of (6).

For the case of a problem which does not vary in time, we performed a Monte Carlo simulation consisting of 100 runs, with the parameters described above. Different $\boldsymbol{y}(t)$, $\boldsymbol{X}^\star$ and $\boldsymbol{n}(t)$ were randomly generated for each run. Figure 1 depicts the convergence in terms of the relative mean-squared-error $\left\|\boldsymbol{X}^i - \boldsymbol{X}^\star\right\|_F^2 / \left\|\boldsymbol{X}^\star\right\|_F^2$. We see that the algorithm consistently converges to reasonable accuracy within two full rounds (i.e. each node has solved the local problem twice, corresponding to 20 iterations in our example). The remaining static error should be attributed to the error inherent to the iterative method used to solve the local problems, and not to our procedure itself (as implied by Theorem 1).

Although we do not provide any proof or quantitative relationship between the rate of change of $\boldsymbol{X}^\star(t)$ and the relative error of the algorithm's estimate of the solution, we illustrate the tracking capabilities of the algorithm with a particular example depicted in Figure 2. Two sparse vectors $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ were drawn from the same distribution used for $\boldsymbol{X}^\star$ in the static case, and $\boldsymbol{X}^\star(t)$ was computed as $w(t)\boldsymbol{X}_A + (1 - w(t))\boldsymbol{X}_B$, where $w(t) = t\cos(t^4)$. The time at iteration i is $t_i = i/180$. Figure 2 depicts the projection of the optimal solution and the algorithm's estimate on the line joining $\boldsymbol{X}_A$ to $\boldsymbol{X}_B$. We see that as the rate of change of the optimal solution increases, the algorithm starts lagging behind the optimal solution.

6. CONCLUSION AND FUTURE WORK

In this paper, we have described a distributed adaptive algorithm to solve a particular family of non-smooth spatial filtering problems. The algorithm was validated both by a formal proof and numerical simulations. In future works, we will provide an analysis of the convergence properties of the algorithm and investigate the link between the global solution accuracy, the local accuracy, and the rate of change of the data (i.e. the tracking performance of the algorithm).

7. REFERENCES

[1] H. Hotelling, “Analysis of a complex of statistical variables into principal components.” *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[2] J. R. Kettenring, “Canonical analysis of several sets of variables,” *Biometrika*, vol. 58, no. 3, pp. 433–451, 1971.

[3] S. Doclo, A. Spriet, J. Wouters, and M. Moonen, “Frequency-domain criterion for the speech distortion weighted multichannel wiener filter for robust noise reduction,” *Speech Communication*, vol. 49, no. 7-8, pp. 636–656, 2007.

[4] Z. J. Koles, M. S. Lazar, and S. Z. Zhou, “Spatial patterns underlying population differences in the background eeg,” *Brain topography*, vol. 2, no. 4, pp. 275–284, 1990.

[5] S. Haykin and K. R. Liu, *Handbook on array processing and sensor networks*. John Wiley & Sons, 2010, vol. 63.

[6] C. A. Musluoglu and A. Bertrand, “A unified algorithmic framework for distributed adaptive signal and feature fusion problems–part i: Algorithm derivation,” *arXiv preprint arXiv:2208.08867*, 2022.

[7] C. A. Musluoglu, C. Hovine, and A. Bertrand, “A unified algorithmic framework for distributed adaptive signal and feature fusion problems–part ii: Convergence properties,” *arXiv preprint arXiv:2208.09088*, 2022.

[8] J. A. Tropp and A. C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[9] D. L. Donoho, “Compressed sensing,” *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[10] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*. Springer Science & Business Media, 2009, vol. 317.

[11] W. Karush, “Minima of functions of several variables with inequalities as side constraints,” *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.

[12] H. Kuhn and A. Tucker, “Nonlinear programming,” in *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics*, 1951, pp. 481–492.

[13] J. Li, A. M.-C. So, and W.-K. Ma, “Understanding notions of stationarity in nonsmooth optimization: A guided tour of various constructions of subdifferential for nonsmooth functions,” *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 18–31, 2020.

[14] J. O. Royset and R. J. Wets, *An Optimization Primer*. Springer, 2021.

[15] C. Berge, *Topological Spaces: including a treatment of multi-valued functions, vector spaces, and convexity*. Courier Corporation, 1997.

[16] D. Charalambos and B. Aliprantis, *Infinite Dimensional Analysis: A Hitchhiker’s Guide*. Springer-Verlag Berlin and Heidelberg GmbH & Company KG, 2013.

[17] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” *Journal of mathematical imaging and vision*, vol. 40, no. 1, pp. 120–145, 2011.