

# **Distributed adaptive spatial filtering in resource-constrained sensor networks**

**Charles HOVINE**

Examination committee:  
Prof. dr. ir. P. Sas, chair  
Prof. dr. ir. A. Bertrand, supervisor  
Prof. dr. ir. S. Pollin  
Prof. dr. ir. M. Moonen  
Prof. dr. ir. P. Patrinos  
Prof. dr. ir. P. Di Lorenzo  
(Sapienza University of Rome)

Dissertation presented in partial  
fulfillment of the requirements for  
the degree of Doctor of Engineering  
Science (PhD): Electrical Engineer-  
ing

August 2024

© 2024 KU Leuven – Faculty of Engineering Science  
Uitgegeven in eigen beheer, Charles Hovine, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

# Preface

---

These last four years have been the opportunity to reflect on the path I want to take as an engineer, better understand what drives me, and go deep in a variety of remarkably interesting topics. I am grateful to live in a society that values knowledge to the extent that it believes that committing four person-years to research with no guarantee of return is a reasonable investment.

I wish to thank my promoter Alexander Bertrand for trusting me and giving me the full freedom to explore the topics that triggered my curiosity, even when it was obvious that I was not moving in straight lines. I believe that his feedback, always as suggestion, never as instruction, was key to the positive reviews received during those four years. I also wish to thank the members of my supervisory committee, Sofie Pollin and Marc Moonen for their periodical feedbacks and follow-ups. Being well-aware that this text is not as smooth a read as Joël Dicker's latest novel, I am very grateful for the time that the remaining members of my examination committee, Panos Patrinos, Paolo Di Lorenzo and Paul Sas, will take to review this thesis, and for anyone patient enough to go through this text.

This is also the opportunity to thank my family, and in particular my parents, as I am only starting to realize what was given to me.

I also thank all the colleagues I met over those four years, amongst them many truly bright individuals. A Ph.D. is a solitary task, but gathering around the coffee machine or lunch table was always a good reason to leave the comforts of home and brave the rain.

Finally and above all, I want to thank my wife Camille, who, in addition to her complete and unfaltering support, has given me two beautiful children, and a place that I can gladly call home.

*Tervuren, August 2024.*

# Abstract

---

Wireless sensor networks consist in a collection of battery-powered sensors able to gather, process and send data. They are typically used to monitor various phenomena, in a plethora of fields, from environmental studies to smart logistics. Their wireless connectivity and relatively small size allow them to be deployed practically anywhere, even underwater or embedded in everyday clothing, and possibly capture data over a large area for extended periods of time. Their usefulness is therefore tied to their ability to work autonomously, with as little human intervention as possible. This functional requirement directly translates into two design constraints: (i) bandwidth and on-board compute must be used sparingly, in order to extend battery-life as much as possible, and (ii) the system must be resilient to node failures and changing environment. Due to their limited computing capabilities, data processing is usually performed by a so-called “fusion center” outside of the network. This therefore requires all the nodes within the network to send and route their data to this fusion center, putting strain on both the batteries and wireless links of the sensor nodes. Furthermore, this data centralization approach harms the network’s resilience as, if the fusion center malfunctions, the sensor network loses its ability to process its collected data.

Several distributed signal processing algorithms have been developed to get around those drawbacks, by favoring local on-board data processing over data transmission. These algorithms typically work by having the nodes collaborate to solve a signal processing task *in-network*, and only forward the result, usually of significantly smaller size than the raw sensor signals, to the outside world. In particular, this approach has been successfully applied to the computation of “spatial filters”, consisting in lower-dimensional linear combinations of the aggregated raw sensor signals, that are optimal in some sense. To meet the

---

aforementioned design requirements, a particular family of distributed signal estimation algorithms, which includes as notable examples the distributed adaptive node-specific signal estimation algorithm (DANSE) and the distributed adaptive covariance matrix eigenvectors estimation algorithm (DACMEE), has introduced collaborative filter computation procedures that are energy-efficient, scalable, and adaptive. This last characteristic and the collaborative nature of the procedures allow the network to operate even in the case of node or sensor failures, and to adapt to a possibly changing environment, i.e., it can handle non-stationary signals effectively. Thanks to common algorithmic themes, this family of algorithms, which we call the “distributed signal fusion” (DSF) family of algorithms, has recently been summarized by the distributed adaptive signal fusion (DASF) framework, of which each of the DSF algorithms is a particular instance.

Despite their benefits, these algorithms are limited in the scope of problems they can handle. Indeed, each DSF algorithm is problem-specific, and the DASF framework requires some technical assumptions to hold to ensure convergence and correctness, which precludes some spatial filters to be covered. In this thesis, we propose several algorithms and algorithmic extensions that improve the energy-efficiency of the procedures introduced by the DSF and DASF algorithms. Noting that nodes observing uncorrelated signals do not gain from collaborating, and hence exchanging data, we first propose a DSF algorithm for the so-called MAXVAR problem, from which we derive an efficient procedure to evaluate the inter-node correlation structure, without the need to exchange the full raw signals to compute the network-wide correlation matrix. Secondly, we prove that the key algorithmic ideas underpinning the DSF algorithms, as formalized in the DASF framework, lead DASF-based algorithms to converge to optimal spatial filters, when applied to smooth and well-posed spatial filtering problems. Then, we extend the original scope of DASF to non-smooth spatial filtering problems, which notably allows the use of sparsity-inducing regularizers in the mathematical description of the spatial filters, opening the door to a network pruning that can be performed alongside the signal processing task, rather than as an ad-hoc step. Finally, we replace the ideal exact solver assumed by DASF by a family of iterative inexact finite-time solvers, resulting in further energy savings, and relaxed computational requirements. In addition, this last extension allows DASF to produce more stable filters and to function under significantly relaxed assumptions. In particular, it is shown to converge even if the optimization problem associated with the target filter has a solution set that is discontinuous relative to the input signals. Each of our contributions is supported by numerical simulations and extensive theoretical guarantees.

# Beknopte samenvatting

---

Draadloze sensornetwerken bestaan uit een verzameling batterijgevoede sensoren die gegevens kunnen verzamelen, verwerken en versturen. Ze worden meestal gebruikt om verschillende fenomenen te monitoren, in verschillende domeinen, van milieustudies tot smart logistics. Dankzij hun draadloze connectiviteit en relatief kleine formaat kunnen ze vrijwel overal geplaatst worden, zelfs onder water of in kleding, en gegevens vastleggen over een groot gebied voor langere perioden. Hun nut hangt daarom af van hun vermogen om autonoom te werken, met zo min mogelijk menselijke tussenkomst. Deze functionele eis vertaalt zich in twee ontwerpbeperkingen: (i) bandbreedte en rekenkracht moeten spaarzaam worden gebruikt om de levensduur van de batterij zo lang mogelijk te maken en (ii) het systeem moet veerkrachtig zijn tegen knooppuntstoringen en een veranderende omgeving. Vanwege hun beperkte rekencapaciteit wordt de gegevensverwerking meestal uitgevoerd door een zogenaamd “fusiecentrum” buiten het netwerk. Daarvoor moeten alle knooppunten binnen het netwerk hun gegevens naar dit fusiecentrum sturen en routeren, waardoor er druk gezet wordt op zowel de batterijen als op de draadloze verbindingen van elk sensor knooppunt. Bovendien schaadt deze gecentraliseerde aanpak de veerkracht van het netwerk, omdat het sensornetwerk de verzamelde gegevens niet meer kan verwerken als het fusiecentrum uitvalt.

Er zijn verschillende gedistribueerde signaalverwerkingsalgoritmen ontwikkeld om deze nadelen te omzeilen door de voorkeur te geven aan lokale gegevensverwerking boven gegevensoverdracht. Deze algoritmen werken meestal door de knooppunten te laten samenwerken om een signaalverwerkingstaak *in het netwerk* op te lossen en alleen het resultaat, door te sturen naar de buitenwereld. Deze aanpak is met name succesvol toegepast op de berekening van “spatiale filters”, bestaande uit lineaire combinaties van de geaggregeerde

ruwe sensorsignalen, die in zekere zin optimaal zijn. Om aan de bovengenoemde ontwerpeis te voldoen, hebben een bepaalde familie van gedistribueerde signaalverwerkingsalgorithmen, die de *distributed adaptive node-specific signal estimation* (DANSE) algoritme en de *distributed adaptive covariance matrix eigenvectors estimation* (DACMEE) algoritme begrijpt, een collaboratieve filter berekeningsprocedure geïntroduceerd die energieëfficiënt, schaalbaar en adaptief is. Deze eigenschappen zorgen ervoor dat het netwerk operationeel blijft, zelfs in het geval van sensorknooppuntstoringen, en om zich aan te passen aan een mogelijk veranderende omgeving. Dat betekent dus dat het netwerk ook kan omgaan met niet-stationaire signalen. Dankzij gedeelde ideeën, deze familie van algorithmen die “gedistribueerde signalfusie” (DSF) genoemd wordt, werd onlangs samengevat door de gedistribueerde signalfusie raamwerk (DASF).

Ondanks deze voordelen zijn deze algoritmen beperkt in de problemen die ze aankunnen. Elk DSF algoritme is namelijk specifiek voor een bepaald probleem intworpen, en het DASF-raamwerk vereist een aantal technische aannames om convergentie en correctheid te garanderen, die voor sommige spatiale filteringstechnieken niet voldaan zijn. In deze thesis stellen we verschillende algoritmen en algoritmische uitbreidingen voor die de energieëfficiëntie van de procedures geïntroduceerd door de DSF en DASF algoritmen verbeteren. Aangezien een samenwerking en gegevensuitwisseling tussen knooppunten die ongecorrleerde signalen waarnemen zinloos is, stellen we eerst een gedistribueerd algoritme voor voor het zogenaamde MAXVAR-probleem, waaruit we een efficiënte procedure afleiden om de correlatiestructuur tussen knooppunten te evalueren, zonder de noodzaak om de ruwe signalen uit te wisselen. Ten tweede bewijzen we dat de belangrijkste algoritmische ideeën die aan de basis liggen van de DSF algoritmen ertoe leiden dat DASF convergeert naar optimale spatiale filters, wanneer toegepast differentieerbare en *well-posed* spatiale filteringproblemen. Vervolgens breiden we het oorspronkelijke toepassingsgebied van DASF uit naar niet-differentieerbare spatiale filteringproblemen, wat het met name mogelijk maakt om *sparsity*-inducerende regularisatoren te gebruiken in de wiskundige beschrijving van de spatiale filters, wat de deur opent naar een data-gedreven sensorselectie die samen met de signaalverwerkingstaak kan geoptimaliseerd worden. Tot slot vervangen we de ideale exacte oplossingsmethode die wordt verondersteld door DASF door een familie van iteratieve inexacte oplossingsmethoden, wat resulteert in verdere energiebesparingen en minder rekenkundige eisen. Bovendien kan deze laatste uitbreiding DASF laten werken onder minder strenge assumpties en condities. Het is zelfs aangetoond dat DASF convergeert als het optimalisatieprobleem een oplossingsverzameling heeft die discontinu is ten opzichte van de ingangssignalen. Elk van onze bijdragen wordt ondersteund door numerieke simulaties en uitgebreide theoretische garanties.

---

Kindly translated from the original english by my wife, Camille, with some kind suggestions from Alexander Bertrand.

# Contents

---

<b>Preface</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Contents</b>	<b>vii</b>
<b>List of symbols</b>	<b>xii</b>
<b>List of abbreviations</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Contributions and structure of the thesis . . . . .	5
1.2 Preliminaries . . . . .	8
1.2.1 WSN setting and notation . . . . .	8
1.2.2 Mathematical background . . . . .	10
<b>2 A distributed algorithm for the MAXVAR problem</b>	<b>17</b>
2.1 Introduction . . . . .	19
2.2 MAXVAR . . . . .	20
2.2.1 Total squared correlation . . . . .	22
2.2.2 MAXVAR: extension to more than two subspaces . . . . .	23
2.2.3 Problem statement . . . . .	24
2.3 Distributed MAXVAR in fully-connected networks . . . . .	25
2.3.1 Algorithm derivation . . . . .	26
2.3.2 Convergence and optimality . . . . .	30
2.4 Distributed MAXVAR in arbitrary network topologies . . . . .	32
2.4.1 Star-topology networks . . . . .	32

2.4.2	Tree-topology networks . . . . .	35
2.4.3	Arbitrary networks . . . . .	39
2.4.4	Complexity and communication cost . . . . .	40
2.5	Correlation structure estimation . . . . .	41
2.5.1	MAXVAR as a low-rank approximation . . . . .	41
2.5.2	Naive TSC approximation using D-MAXVAR . . . . .	42
2.5.3	TSC approximation using a modified D-MAXVAR . . . . .	43
2.5.4	Selecting $Q$ . . . . .	45
2.6	Simulation results . . . . .	46
2.6.1	Simulation settings . . . . .	46
2.6.2	TSC approximation . . . . .	47
2.6.3	Convergence . . . . .	49
2.7	Discussion . . . . .	51
<b>3</b>	<b>A general framework for smooth distributed spatial filtering problems</b>	<b>53</b>
3.1	Preliminaries . . . . .	55
3.1.1	From Gauss-Seidel to DASF . . . . .	55
3.1.2	Data-driven spatial filtering . . . . .	58
3.2	DASF overview . . . . .	60
3.2.1	Fully-connected networks . . . . .	60
3.2.2	Arbitrary network topologies . . . . .	67
3.3	Convergence and optimality of DASF . . . . .	71
3.3.1	From global to local problems and back . . . . .	71
3.3.2	Key assumptions . . . . .	75
3.3.3	Convergence . . . . .	76
3.3.4	Optimality . . . . .	79
3.3.5	Convergence issues . . . . .	83
3.4	A note on ADMM . . . . .	86
3.5	Discussion . . . . .	87
<b>4</b>	<b>Non-smooth DASF</b>	<b>89</b>
4.1	Introduction . . . . .	91
4.2	Problem scope . . . . .	92
4.2.1	Extended scope of the NS-DASF framework . . . . .	93
4.3	The non-smooth DASF algorithm . . . . .	95
4.3.1	NS-DASF in fully-connected networks . . . . .	95
4.3.2	NS-DASF in arbitrary network topologies . . . . .	98
4.4	Convergence and optimality . . . . .	101
4.4.1	Notation and proof outline . . . . .	101
4.4.2	Subsequential convergence (in fully-connected networks) . . . . .	103
4.4.3	Optimality of fixed points (in fully-connected networks) . . . . .	110
4.4.4	Extension to arbitrary network topologies . . . . .	112

4.4.5	Constraint qualifications and an upper bound on the number of constraints . . . . .	114
4.4.6	Main result . . . . .	115
4.5	Numerical experiments . . . . .	117
4.6	Discussion . . . . .	120
<b>5</b>	<b>DASF with finite-time solvers</b>	<b>123</b>
5.1	Introduction . . . . .	125
5.2	Problem statement . . . . .	125
5.3	DASF with inexact local solvers . . . . .	127
5.3.1	DASF algorithm with inexact solver . . . . .	129
5.3.2	Convergence and optimality . . . . .	131
5.3.3	A bound on the convergence rate . . . . .	138
5.4	Numerical experiments . . . . .	139
5.5	Conclusion . . . . .	142
<b>6</b>	<b>Conclusion</b>	<b>143</b>
<b>A</b>	<b>Proofs of Chapter 2</b>	<b>147</b>
A.1	Proof of Theorem 2.1 . . . . .	147
A.2	Proof of Theorem 2.2 . . . . .	148
A.3	Proof of Theorem 2.3 . . . . .	152
A.4	Proof of Equation (2.40) . . . . .	152
A.5	Proof of Theorem 2.4 . . . . .	153
<b>B</b>	<b>Convergence of Gauss-Seidel</b>	<b>154</b>
<b>C</b>	<b>Proofs of Chapter 4</b>	<b>160</b>
C.1	Continuity of quadratic orthogonality constraints . . . . .	160
C.2	Proof of Lemma 4.1 . . . . .	160
C.3	Proof of Lemma 4.2 . . . . .	161
C.4	Proof of Lemma 4.4 . . . . .	162
C.5	Proof of Proposition 4.5 . . . . .	162
<b>D</b>	<b>Chambolle-Pock algorithm for the group-lasso problem</b>	<b>163</b>
<b>E</b>	<b>Common inexact solvers</b>	<b>165</b>
E.1	Line-search methods: gradient descent and Newton's method . . . . .	166
E.2	Regularized exact solver . . . . .	167
E.3	Power method . . . . .	168
<b>F</b>	<b>Useful lemmas</b>	<b>169</b>
	<b>Bibliography</b>	<b>173</b>

<b>List of publications</b>	<b>185</b>
<b>Acknowledgments</b>	<b>186</b>

# List of symbols

---

$(\cdot)^\dagger$	The Moore-Penrose pseudo-inverse.
$A^T$	The transpose of the matrix $A$ .
$A^{-1}$	The inverse of the matrix $A$ .
$P_{\mathcal{V}}(u)$	The orthogonal projection of $u$ onto the set $\mathcal{V}$ .
$\mathbb{E}\{\cdot\}$	The expectation operator.
$\mathbb{N}$	The set of natural numbers.
$\mathbb{R}$	The set of real numbers.
$\text{BlockDiag}(A_1, \dots, A_K)$	A block-diagonal matrix formed from the blocks $A_1, \dots, A_K$ .
$R_{\mathbf{a}, \mathbf{b}}$	The covariance matrix of two multi-channel stochastic signals $\mathbf{a}$ and $\mathbf{b}$ .
$\delta_{\mathcal{X}}(\cdot)$	The indicator function of a set $\mathcal{X}$ .
$\ell_1$	The $\ell_1$ -norm $\ \cdot\ _1$ .
$\nabla f(u)$	The gradient of $f$ at a point $u$ .
$\ \cdot\ $	A norm, The Frobenius norm is implied in the case of a matrix argument..
$\ \cdot\ _F$	The Frobenius norm.

---

$\mathbb{R}$	The extended real line $\mathbb{R} \cup \{-\infty, \infty\}$ .
$\preceq$	Element-wise $\leq$ .
$(u^i)_{i \in \mathcal{I}}$	A sequence indexed by some set $\mathcal{I}$ .
Span $\mathbf{y}$	The span of the channels of $\mathbf{y}$ .
Acc $\cdot$	The set of accumulation points of a sequence.
Null( $\cdot$ )	The null space of matrix.
Range( $\cdot$ )	The column space of matrix.
Tr ( $\cdot$ )	The trace operator.

# List of abbreviations

---

CCA	Canonical correlation analysis.
CCTV	Closed-circuit television.
D-MAXVAR	Distributed MAXVAR (algorithm).
DACME	Distributed adaptive covariance matrix eigenvector estimation (algorithm).
DANSE	Distributed adaptive node-specific signal estimation (algorithm).
DASF	Distributed adaptive signal fusion (framework), or distributed adaptive signal filtering (framework).
EEG	Electro-encephalography.
FC	Fusion center.
GEVC	Generalized eigenvectors.
GEVD	Generalized eigenvalue decomposition.
GEVL	Generalized eigenvalues.
ICA	Independent component analysis.
KKT	Karush-Kuhn-Tucker.
KLT	Karhunen-Loève transform.

LCMV	Linearly constrained minimum variance beamforming.
LICQ	Linear independence constraint qualifications.
MAXVAR	Maximum variance generalized canonical correlation analysis.
MM	Majorization-minimization.
PCA	Principal component analysis.
SNR	Signal-to-noise ratio.
WSN	Wireless sensor network.





# Chapter 1

---

## Introduction

---

As “artificial intelligence” has become a common-place expression, it now seems that our ability to model the world is not anymore restricted by our physical understanding thereof, but by the amount of data that we are able to collect, and subsequently process. If massive amounts of compute are today remotely available in data-centers scattered around the globe, data generated at the so-called “edge” still needs to be collected and routed from its source to the nearest compute-hub [1], [2]. This is trivial for our personal data, which is more likely than not being generated directly by a fleet of connected devices. The task becomes more arduous when it comes to the measurement of physical phenomenons in remote areas, or for any data sensed by an energy-constrained battery-powered device, such as a wireless sensor node [3].

**Wireless sensor networks** A wireless sensor network (WSN) consists of a collection of such sensor nodes with limited computing and wireless communication capabilities, and are typically used to continuously and autonomously collect data, with as little human intervention as possible. They must therefore be resilient to hardware failures to reduce maintenance actions, and low-power to maximize battery-life [4]–[6]. WSNs are, or are at least envisioned to be, the eyes and ears of tomorrow’s “smart” systems. They can be used to collect data in heterogeneous fields such as traffic management [7], personal health [8], environmental monitoring [9], [10], wildlife preservation [11] manufacturing processes [12] and agriculture [13], to name a few. This data can then be fed back to a decision system [14]–[16], or simply stored or analyzed [17].

A typical WSN aggregates and forwards the data it collects to a so-called “fusion center” (FC) [18], where it can be processed or further forwarded to a more powerful device [2]–[4]. However, using WSNs as mere sensing devices, without leveraging their computing capabilities, can be quite ineffective and comes with several drawbacks [19].

Firstly, the transmission of the sensor nodes’ raw data is likely to become the main energy bottleneck of the system [20], in particular in networks where multi-hop data relaying is required, resulting in shorter battery life, making remote deployments impractical. This poses a particular challenge for applications that generate continuous streams of high-throughput sensor data, such as audio in acoustic sensor networks [21], video in CCTV systems or video sensor networks [22], biomedical signals in EEG<sup>1</sup> sensor networks [23], [24], or radio signals in multistatic radars [25], [26]. Secondly, computations are cheap relative to data transmission, a back-of-the-envelope calculation gives that within the energy budget corresponding to the transmission of a single 32-bit float, roughly 300000 load and multiply-accumulate<sup>2</sup> instructions can be performed<sup>3</sup>. Thirdly, the fusion center needs to have sufficient computing capabilities to handle the continuous processing of the stream of high-dimensional input data it receives. This comes with scalability issues, as the increase in the number of nodes or input data will eventually lead to unrealistic hardware requirements at the fusion center in terms of bandwidth, memory, and compute. Finally, the use of a fusion center introduces a single point of failure [29], which can be prohibitive if maintenance actions are difficult or costly, or if high availability is required. These drawbacks associated with centrally processing the data motivate the use of distributed data processing algorithms that distribute the computational burden amongst the nodes and favor local processing over data transmission.

**Spatial filtering** Spatial filtering is a common signal processing task in such networks, which consists in the extraction of some target signal from the aggregated sensor signals. A spatial filter is typically expressed as a linear combination of the sensor signals, producing a lower-dimensional output signal that is optimal in some sense, for example, in terms of signal-to-noise ratio, output power, or correlation with a target signal. Common examples include principal component analysis (PCA) [30] and the Karhunen–Loève transform (KLT) [31], which extracts the highest-power or highest variance component in a multi-channel signal, and can be used as a compression method, Wiener filtering [32] which extracts the component closest to some target signal, canonical

---

<sup>1</sup>Electro-encephalography.

<sup>2</sup>These are the basic two instructions used in the computation of a digital filter, or the estimation of a covariance matrix.

<sup>3</sup>This example assumes the PicoRV32 RISC-V processor [27] for computations, and a typical bluetooth-low-energy module operating at peak efficiency [28] for data transmission.

correlation analysis (CCA) [33], which finds highly correlated components between multiple sets of signals, independent component analysis (ICA) [34], which separates statistically independent components, linearly constrained minimum variance beamforming (LCMV) [35], which can selectively block interference sources based on their spatial location, and Max-SNR filtering [36], which finds a projection maximizing the signal-to-noise ratio of a desired signal.

These filters can either be computed statically, or adaptively. In the first case, the sensor data is initially used to “train” the filter, which is then applied on the new data being generated. This approach has the benefit of separating the computation of the filter, and that of its output, allowing the filter to be computed fully offline, by a potentially more powerful system, free from the constraints of a WSN. Still, this only works in the case of highly stationary data, i.e., whose statistics are fairly constant over long periods of time. A filter computed in such a fashion, would, for example, not be able to recover from a sensor failure. In the second case, the computation of the filter and its evaluation are fully entangled, and new data is constantly used to optimize the performance of the filter via some feedback mechanism [37]. Such a filter continuously adapts to the changing statistics of the sensor signals, and is therefore resilient to sensor failures and other non-stationarities in the data.

**Distributed signal processing and optimization** In view of the high-cost of data transmission in WSNs and the need for adaptive filtering, there is a strong incentive to process the collected data *in-network*, rather than sending it to an FC. Due to the limited number of hardware resources available at each node, this typically requires the computational load to be shared amongst the nodes, which requires specialized algorithms able to efficiently distribute computations while also minimizing data transmission.

A typical machine-learning or statistical signal processing task will at some point rely on the inter-feature or inter-channel statistics of the data or signals, which will be used to produce predictions, classify or summarize the data. Those statistics are usually estimated from some dataset, and distilled down by the relevant model into the target information [38], [39]. We can identify two classes of distributed signal processing algorithms, associated with two corresponding kinds of distributed datasets. The first kind covers algorithms operating on the observations of a stochastic signal  $\mathbf{y} \in \mathbb{R}^M$ , whose observations, or samples, are distributed across the nodes, such that each node has access to all  $M$  entries of  $\mathbf{y}$ , but only a subset of the realizations of the random data associated with those entries. This allows each node to locally estimate the covariance structure of the full signal. The optimality criterion can in this case typically be expressed as a sum of local per-node objectives, and the distributed algorithm

will usually consist in performing some local processing on the node, and then exchanging some low-dimensional vector containing intermediate estimates of the optimization variables (rather than signal samples), used to refine the individual nodes' decisions, or reach an agreement on the optimal output. Typical examples of algorithms suited for such datasets are the alternating direction method of multipliers (ADMM) [40], consensus and diffusion strategies [41], [42], and many of the techniques studied by the federated learning community [43].

The second kind of distributed algorithms deals with datasets where the entries (or channels) of  $\mathbf{y}$  are spread across the nodes, such that no individual node can estimate the correlation structure of the network-wide signal  $\mathbf{y}$  (i.e. the signal consisting in the concatenation of all the channels of the per-node signals). Indeed, the computation of the correlation between two signals requires them to be co-located, and at least one signal thus needs to be transferred, possibly partially. Therefore, such algorithms typically require that signal samples (rather than low-dimensional parameter vectors) are shared between the nodes, in order to learn the inter-node statistics and to properly evaluate the optimality criterion. Distributed spatial filtering problems or regression/classification problems with distributed features fit in this second class, and are more difficult to solve if these inter-node statistics are not known a-priori. For streaming data or datasets where the number of observations of  $\mathbf{y}$  is much larger than the number of channels  $M$ , the typical work-horse distributed methods (ADMM, consensus, diffusion, etc.) do not straightforwardly apply [44], or result in highly inefficient multi-rate communication schemes with nested iterations [45], [46]. Therefore, due to the absence of a one-fit-all off-the-shelf solution, they are typically solved with ad-hoc problem-specific algorithms [35], [46]–[52].

The distributed adaptive node-specific signal estimation (DANSE) algorithm [53] and related algorithms [49], [54]–[57], which we call the “distributed signal fusion” (DSF) family of algorithms, are of particular interest in this last category, as they have been successfully applied to some of the example problems mentioned above. They allow the distributed and adaptive computation of some typical data-driven spatial filters in a bandwidth-efficient manner, and with good scaling properties. Although each of those algorithms is still problem-specific, they share common ideas, later leveraged by the distributed adaptive signal fusion (DASF) algorithm, [58], which describes a unifying framework under which the DSF algorithms all fall, and therefore opened the doors to a much broader range of spatial filtering problems.

---

This introductory text is in part based on the introductions of [52], [59], [60].

## 1.1 Contributions and structure of the thesis

This thesis builds on the DSF algorithms [49], [53]–[56] and the ground work laid out in [58] on the general distributed framework of DASF. If those algorithms and the framework hold their promises in terms of adaptivity, limited compute/bandwidth requirements, and correctness, they can be improved in several key aspects, which we will explore throughout the chapters of this thesis.

Although it is not an intrinsic limitation of those algorithms, it is generally assumed that every node within a given WSN can usefully contribute to the target signal processing task. When this is not the case, precious resources are wasted on void computations. It is therefore important to identify which subsets of nodes can usefully contribute to the task at hand. When computing a data-driven spatial filter depending on the network-wide covariance matrix, the blocks of the covariance matrix associated with nodes observing mostly uncorrelated signals can typically be neglected, e.g. when the covariance matrix has some block structure, and the task can then usually be split in several independent sub-tasks, involving different subsets of nodes, which do not need to exchange data. In **Chapter 2**, we propose an efficient task-agnostic way of identifying such subsets of nodes, via a DSF algorithm that computes the solutions of the so-called MAXVAR problem. This first chapter also serves as an introduction to the general structure of a DSF algorithm, with a concrete problem-specific algorithm. This chapter is based on:

- \* **C. Hovine** and A. Bertrand, “Distributed MAXVAR: Identifying common signal components across the nodes of a sensor network”, in *2021 29th European Signal Processing Conference (EUSIPCO)*, IEEE, 2021, pp. 2159–2163
- \* **C. Hovine** and A. Bertrand, “MAXVAR-based distributed correlation estimation in a wireless sensor network”, *IEEE Transactions on Signal Processing*, vol. 70, pp. 5533–5548, 2022

**Chapter 3** introduces the general framework of DASF, and provides intuition on its benefits by contrasting it with a naive non-linear Gauss-Seidel method. We then show that, under the proper assumptions, the key ideas of the DSF algorithms lead to converging algorithms with optimality guarantees when applied to generic smooth spatial filtering problems via DASF. We describe a proof of convergence, and suggest a first modification of DASF that converges under relaxed conditions. This chapter is partly based on:

- \* C. A. Musluoglu, **C. Hovine**, and A. Bertrand, “A unified algorithmic framework for distributed adaptive signal and feature fusion problems — part

II: Convergence properties”, *IEEE Transactions on Signal Processing*, vol. 71, pp. 1879–1894, 2023

We broaden the scope of DASF in **Chapter 4** by extending it to a new class of spatial filters that can only be expressed as the solutions of non-smooth optimization problems. This extension allows us to leverage the sparsity-inducing property of some norms to, e.g., dynamically prune the network in a task-specific fashion, removing the need to use inter-node correlation as a proxy for usefulness, and not requiring an extra pruning step to be performed, as suggested in **Chapter 2**. This chapter is based on:

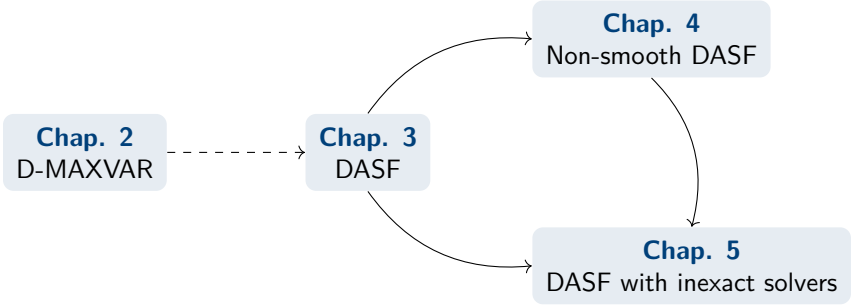
- \* **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems”, in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5
- \* **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems in wireless sensor networks”, *IEEE Transactions on Signal Processing*, vol. 72, pp. 4682–4697, 2024

Finally, in **Chapter 5**, we address computational cost of the algorithm by showing that DASF shows better convergence properties when replacing the exact solver used in the original framework by an inexact and finite-time solver, opening the door to the use of computationally efficient off-the-shelf iterative solvers such as (proximal-)gradient descent. This chapter also relaxes several of the stricter convergence assumptions of DASF. This chapter is based on:

- \* **C. Hovine** and A. Bertrand, “Distributed adaptive spatial filtering with inexact local solvers”, *arXiv preprint arXiv:2405.03277*, 2024 (submitted for publication)

We conclude this thesis with a summary of our results, along with a discussion on the applicability and relevance of the proposed methods, and suggest future research directions in **Chapter 6**.

The following diagram depicts the dependencies between the chapters of this thesis. The first dashed line denotes an optional dependency.



## 1.2 Preliminaries

This short section formalizes the general WSN setting in which the algorithms of this thesis are assumed to operate. It also provides a short introduction to some of the key mathematical concepts on which the text relies.

### 1.2.1 WSN setting and notation

We consider a WSN consisting of  $K$  nodes in which each node  $k \in \mathcal{K} = \{1, \dots, K\}$  collects discrete observations of a real-valued  $M_k$ -channel sensor signal  $\mathbf{y}_k = [y_{k,1}, \dots, y_{k,M_k}]^T$ . We model  $\mathbf{y}_k \in \mathbb{R}^{M_k}$  as a zero-mean stochastic process and denote  $\mathbf{y}_k(t)$  its value at time  $t$ . Finally, we define the network-wide observation vector as the  $M$ -channel vector  $\mathbf{y}$  obtained by stacking the  $\mathbf{y}_k$ 's and where  $M = \sum_k M_k$ . The nodes are organized in a network that defines a communication graph, precluding certain nodes to communicate directly with one another. The typical network topologies we consider in this thesis are illustrated in Figure 1.1.

In order to make the theoretical analysis mathematically tractable, we will often implicitly make two key assumptions, described hereafter. The first assumption allows us to factor-out statistical estimation errors from our developments.

**Assumption 1.1** (Perfect statistical estimation from finite samples). *Any node is able to estimate an expectation operator up to an arbitrary accuracy from sampled data.*

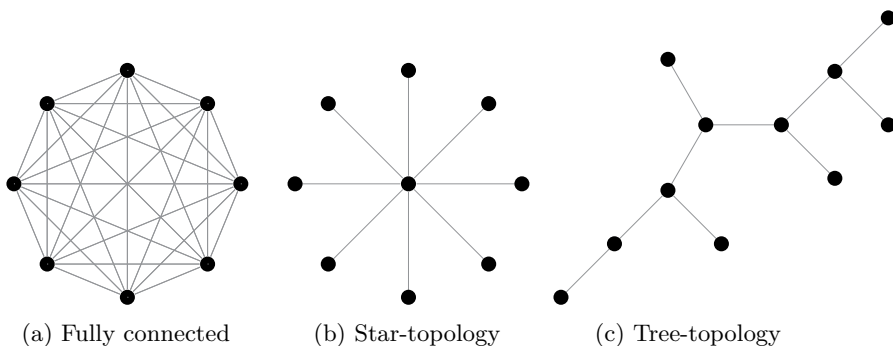


Figure 1.1: Some typical network topologies.

Note that this assumption is only made for mathematical tractability. In practice, these expectations are computed from sample averages over a finite set of samples, assuming that  $\mathbf{y}$  is ergodic and short-time stationary, allowing us to estimate the slowly varying covariance matrices from sample averages over sufficiently long segments of data:

$$R_{\mathbf{y}_k \mathbf{y}_l} \triangleq \mathbb{E} \{ \mathbf{y}_k(t) \mathbf{y}_l^T(t) \} \approx \frac{1}{T} Y_k(t) Y_l^T(t) \quad (1.1)$$

where  $\mathbb{E} \{ \cdot \}$  denotes the expectation operator and  $Y_k(t)$  denotes the  $M_k \times T$  observation matrix containing  $T$  consecutive observations of  $\mathbf{y}_k$  centered around  $t$  in its columns. If  $T$  is large enough, Assumption 1.1 becomes “asymptotically” valid. All the convergence results in this thesis should be viewed in this asymptotic setting.

This next assumption allows us to decouple the error associated with the iterative optimization procedures used within our algorithms, from the errors arising due to change in the statistics of the data.

**Assumption 1.2 (Stationarity).** *The statistics of the data do not change over time.*

We emphasize this fact by dropping the time index  $t$  of  $\mathbf{y}(t)$  for most of the remainder of this text. We obviously do not require this assumption to hold in practice, as it would prevent any sort of adaptivity, but it allows us to significantly simplify our mathematical derivations (as often done in the theoretical analysis of adaptive filters). Note that the stationarity assumption is mostly true in the case where the dynamics of our algorithms are much faster than the change in signal statistics. The practical requirement is therefore not full stationarity, but short-time stationarity. In order to validate the adaptive properties of our algorithms, this assumption is lifted in the relevant simulations. Finally, note that we in practice only require the signals to be *wide-sense* stationary, i.e., only the first and second order moments should not change over time. Indeed, none of the problems considered in this thesis depend on higher-order moments.

**Power consumption model** As one of our stated objective is the reduction of the power consumption of the wireless sensor nodes, we describe here a rough model mapping computations, data transmission and memory accesses to energy consumption. The dynamic part of the power consumption, i.e. dependent on the amount of data processed, can be split into three contributions

- \*  $e_t$ , the energy spent to perform inter-node data transfers.
- \*  $e_m$ , the energy spent to write and read data to and from memory.
- \*  $e_c$ , the energy spent to perform computations (once data is loaded in the proper CPU registers).

All three contributions are monotonically increasing functions of the amount of data processed.  $e_t$  is many times larger than  $e_c$  and  $e_m$  [6], [20], and data transmissions should therefore be minimized wherever possible. If  $e_c$  and  $e_t$  can be assumed to be linear functions of the amount of data processed, it is likely not the case of  $e_m$ . Indeed, modern computer architectures typically have a hierarchical memory structure, where frequently accessed data are stored in fast and energy-efficient “on-chip cache” memories close to the CPU, and the remainder of the data is stored in an energy-intensive external “off-chip” memory [64].  $e_m$  is therefore a piece-wise linear function, where, under a certain threshold, data can fit in the cache and be accessed with a low energy cost, and above that threshold, data must be fetched from external memory with a much higher energy expenditure [65], [66]. In this work, we describe methods that decrease the required memory<sup>4</sup> at each node, and simultaneously trade data transmissions for local computations. Whether this trade-off is interesting, largely depends on the ratio between the energy required to process a batch of samples on-board, compared to the energy required to transmit it [6], and is therefore both application and technology dependent.

## 1.2.2 Mathematical background

We very briefly review the essential mathematical concepts used in this text. This review is not meant to be exhaustive, and some of the usual rigor found in mathematical texts is traded for conciseness. For an in-depth treatment of the concepts introduced hereafter, we refer the readers to the reference works [67], [68] for variational analysis, and [69], [70] for general analysis.

**Sequences in metric spaces** Euclidean space is an inner product space, i.e., it is associated with the *inner product*

$$\langle u, v \rangle \triangleq \sum_k u_k v_k \quad (1.2)$$

---

<sup>4</sup>In this work, the required memory is assumed to be at least proportional to the dimension of the processed data, which is a fair assumption, as memory and computational complexity often grow quadratically or faster for common applications, e.g. the required memory for the computation of the covariance matrix of an  $N$ -dimensional signal is proportional to  $N^2$ .

where  $u_k$  denotes the  $k$ -th entry of the vector  $u$ . From this inner product, we can define the *norm*

$$\|u\|^2 \triangleq \langle u, u \rangle, \quad (1.3)$$

and from this norm, we can define the *distance* between two points as

$$d(u, v) \triangleq \|u - v\|. \quad (1.4)$$

A space equipped with a distance function is called a *metric* space. All the spaces we work with in this text are metric spaces.

Let  $(u^i)_{i \in \mathbb{N}}$  be a sequence in a metric space. We say that  $(u^i)_{i \in \mathbb{N}}$  *converges* to its *limit*  $\bar{u}$  if

$$\forall \varepsilon > 0, \exists N \in \mathbb{N} : i > N \Rightarrow d(u^i, \bar{u}) < \varepsilon, \quad (1.5)$$

which we denote

$$\lim_{i \rightarrow \infty} u^i = \bar{u}. \quad (1.6)$$

An *accumulation point* of  $(u^i)_{i \in \mathbb{N}}$  is a point  $\bar{u}$  for which we can find some index set  $\mathcal{I} \subseteq \mathbb{N}$ , such that

$$\forall \varepsilon > 0, \exists N \in \mathbb{N} : i > N, i \in \mathcal{I} \Rightarrow d(u^i, \bar{u}) < \varepsilon, \quad (1.7)$$

which we denote

$$\lim_{i \in \mathcal{I} \rightarrow \infty} u^i = \bar{u}. \quad (1.8)$$

A set  $\mathcal{U}$  is said to be *closed* if the limit of every convergent sequence living in  $\mathcal{U}$  is in  $\mathcal{U}$ , i.e., for any sequence  $(u^i)_{i \in \mathbb{N}}$ ,

$$u^i \in \mathcal{U} \forall i \in \mathbb{N} \Rightarrow \lim_i u^i \in \mathcal{U}. \quad (1.9)$$

A set  $\mathcal{U}$  is said to be *bounded* if we can find some  $M > 0$  such that

$$d(u, v) < M \quad \forall u, v \in \mathcal{U}. \quad (1.10)$$

A subset of Euclidean space that is both closed and bounded is said to be *compact*. In a compact set, every sequence has a convergent subsequence.

**Functions** We denote by  $u \mapsto f(u)$  or  $f : \mathcal{U} \rightarrow \mathcal{V}$ , a *function*, *map* or *mapping*  $f$  mapping points from some set  $\mathcal{U}$  to some set  $\mathcal{V}$ . We say that  $f : \mathcal{U} \rightarrow \mathcal{V}$  is *continuous* if for any sequence  $(u^i)_{i \in \mathbb{N}}$ ,

$$\lim_i u^i = \bar{u} \Rightarrow \lim_i f(u^i) = f(\bar{u}). \quad (1.11)$$

We define the *limit inferior* and *limit superior* of a sequence as the infimum and supremum, respectively, of its set of accumulation points. We denote it  $\liminf$  and  $\limsup$ . We say that a function is *lower semicontinuous* if

$$\liminf_i u^i = \bar{u} \Rightarrow \liminf_i f(u^i) \geq f(\bar{u}), \quad (1.12)$$

and *upper semicontinuous* if

$$\limsup_i u^i = \bar{u} \Rightarrow \limsup_i f(u^i) \leq f(\bar{u}). \quad (1.13)$$

A function is continuous if and only if it is both lower and upper semicontinuous.

Sometimes, we consider functions from  $\mathbb{R}^N$  that can take values in the *extended real line*  $\mathbb{R} = \mathbb{R} \cup \{-\infty, \infty\}$ . Where  $\infty$  is defined as the value such that  $\infty > r$  for any  $r \in \mathbb{R}$ . We say that  $f$  is *proper* if  $f(u) > -\infty$  for any  $u \in \mathbb{R}^N$  and there is some  $u$  such that  $f(u) < \infty$ . We denote the *domain* of  $f$

$$\text{dom}f \triangleq \{u \in \mathbb{R}^N \mid f(u) < \infty\}. \quad (1.14)$$

We define the sub-level sets of a function  $f$ , as the sets

$$\{u \in \text{dom}f \mid f(u) \leq \alpha\}. \quad (1.15)$$

We say that  $f$  is *closed* if all its sub-level sets are closed.

**Set-valued functions** A function  $\mathcal{F}$  mapping points from a set  $\mathcal{U}$  to subsets of some set  $\mathcal{V}$ , sometimes called a *correspondence*, is denoted by  $\mathcal{F} : \mathcal{U} \rightrightarrows \mathcal{V}$ . By introducing the proper distance function (e.g. the Hausdorff distance), we can turn a space of sets into a metric space, and extend the usual definitions of convergence and continuity. Still, we can refine our characterization of the continuity of such maps by introducing the following definitions.

A set-valued map is *outer semicontinuous* if for any sequences  $(u^i)_{i \in \mathbb{N}}$  and  $(v^i)_{i \in \mathbb{N}}$ , the convergence of  $u^i$  to  $\bar{u}$ ,  $v^i \in \mathcal{F}(u^i)$  and the convergence of  $v^i$  to  $\bar{v}$  imply that  $\bar{v} \in \mathcal{F}(\bar{u})$ .

A set-valued map is *inner semicontinuous* if for any sequence  $(u^i)_{i \in \mathbb{N}}$  converging to  $\bar{u}$  and any  $\bar{v} \in \mathcal{F}(\bar{u})$ , we can find some index set  $\mathcal{I} \subseteq \mathbb{N}$  and sequence of  $v^i \in \mathcal{F}(u^i)$  such that  $(v^i)_{i \in \mathcal{I}}$  converges to  $\bar{v}$ .

It is continuous if it is both outer and inner semicontinuous<sup>5</sup>.

We say that  $u$  is a fixed point of  $\mathcal{F} : \mathcal{U} \rightrightarrows \mathcal{U}$  if  $u \in \mathcal{F}(u)$ .

---

<sup>5</sup>This particular concept of set-continuity has different names depending on fields and authors. Authors in set analysis typically use “upper” and “lower” semicontinuity [71], authors in mathematical economics use upper and lower “hemicontinuity” [70], and authors in variational analysis use the terminology we adopt in this thesis [68].

**Indicator function** The *indicator function* of a set  $\mathcal{X}$  is defined as

$$\delta_{\mathcal{X}}(u) \triangleq \begin{cases} 0 & \text{if } u \in \mathcal{X} \\ \infty & \text{otherwise.} \end{cases} \quad (1.16)$$

**Differentiability** We say that a function  $f : \mathbb{R}^N \rightarrow \overline{\mathbb{R}}$  is *differentiable* if there is some unique  $w \in \mathbb{R}^N$  such that for any sequence  $u \rightarrow \bar{u}$ ,

$$\lim_{u \rightarrow \bar{u}} \frac{f(u) - f(\bar{u}) - \langle w, u - \bar{u} \rangle}{\|u - \bar{u}\|} = 0. \quad (1.17)$$

If it exists,  $w$  is the gradient of  $f : \mathbb{R}^N \rightarrow \overline{\mathbb{R}}$  at  $\bar{u}$  and is denoted  $\nabla f(\bar{u})$ . We say that  $f$  is *smooth* or *continuously differentiable* if its gradient  $u \mapsto \nabla f(u)$  exists and is continuous everywhere.

**Subgradients** Following [68, Definition 8.3], we define the set of regular subgradients of  $f$  at  $\bar{u}$  (where  $f(\bar{u})$  is finite) as

$$\hat{\partial}f(\bar{u}) \triangleq \left\{ w \in \mathbb{R}^N \mid \liminf_{\substack{u \rightarrow \bar{u} \\ u \neq \bar{u}}} \frac{f(u) - f(\bar{u}) - \langle w, u - \bar{u} \rangle_F}{\|u - \bar{u}\|} \geq 0 \right\}, \quad (1.18)$$

and the set of general subgradients of  $f$  at  $\bar{u}$  as<sup>6</sup>

$$\partial f(\bar{u}) \triangleq \{ w \in \mathbb{R}^N \mid \exists u^j \rightarrow \bar{u}, f(u^j) \rightarrow f(\bar{u}), w^j \in \hat{\partial}f(u^j) \rightarrow w \}, \quad (1.19)$$

which we refer to simply as “subgradients” from hereon. This definition of a subgradient does not require any structural property such as convexity, smoothness, or even continuity from  $f$ , and extends the usual definition of subgradients from convex analysis. Indeed, (1.19) generalizes the familiar notion of the subgradient of a convex function.

Note that for the particular case of an indicator function  $\delta_{\mathcal{X}}$ , the set of subgradients of  $\delta_{\mathcal{X}}$  at  $u$  corresponds to the *normal cone* of the set  $\mathcal{X}$  at  $u$  [72], denoted

$$\partial \delta_{\mathcal{X}}(u) = N_{\mathcal{X}}(u), \quad (1.20)$$

and the concepts can in this case be used interchangeably.

---

<sup>6</sup>(1.19) can be read as “the set of elements  $w$  such that there exist sequences  $(u^j)_j$  converging to  $\bar{u}$ ,  $(f(u^j))_j$  converging to  $f(\bar{u})$  and  $(w^j)_j$  converging to  $w$ , such that  $w^j$  is a regular subgradient of  $f$  at  $X^j$ ”.

**Local optimality** If  $L : \mathbb{R}^N \mapsto \overline{\mathbb{R}}$  is lower-semicontinuous and  $u^*$  is a local minimum of  $L$ , then [68, Theorem 8.15]

$$0 \in \partial L(u^*). \quad (1.21)$$

Any point  $u^*$  such that  $L(u^*)$  is finite and satisfying (1.21) is called a *stationary point* of  $L$ . This definition reduces to the usual “KKT” (Karush-Kuhn-Tucker) optimality conditions [73], [74] in the smooth case. Indeed, if

$$L(u) \triangleq f(u) + \delta_{\mathcal{X}}(u) \quad (1.22)$$

where  $f$  is a smooth function, and  $\mathcal{X}$  is the closed set  $\{u \mid g(u) \leq 0, h(u) = 0\}$  where  $h$  and  $g$  are smooth functions, then we have at any point  $u$  where  $L(u)$  is finite (and thus  $u \in \mathcal{X}$ ) [72, Proposition 4.58]

$$\partial L(u) = \nabla f(u) + \partial \delta_{\mathcal{X}}(u). \quad (1.23)$$

If  $\nabla g(u)$  and  $\nabla h(u)$  are linearly independent (this is stricter than required, but is only meant for illustrative purposes), then [72, Example 4.49]

$$\begin{aligned} \partial \delta_{\mathcal{X}}(u) &= \{\lambda_g \nabla g(u) + \lambda_h \nabla h(u) \mid \\ &\lambda_h \in \mathbb{R}, \lambda_g = 0 \text{ if } g(u) < 0, \text{ else } \lambda_g \geq 0\}. \end{aligned} \quad (1.24)$$

Defining the *Lagrangian*

$$\mathfrak{L}(u, \lambda_g, \lambda_h) \triangleq f(u) + \lambda_g g(u) + \lambda_h h(u), \quad (1.25)$$

we have for a feasible point  $u$  the equivalence

$$\begin{aligned} 0 \in \partial L(u) &\Leftrightarrow \\ &\exists \lambda_g = 0 \text{ if } g(u) < 0, \text{ else } \lambda_g \geq 0, \lambda_h : \nabla_u \mathfrak{L}(u, \lambda_g, \lambda_h) = 0. \end{aligned} \quad (1.26)$$

## Linear algebra

We denote the *span* of a set of vectors

$$\text{Span}\{u_1, \dots, u_K\} \triangleq \{\alpha_1 u_1 + \dots + \alpha_K u_K \mid \alpha_1, \dots, \alpha_K \in \mathbb{R}\}, \quad (1.27)$$

where  $u_k$  now denotes a vector rather than a component. The span of a set of vectors is a linear vector space. We say that two vector spaces  $\mathcal{U}$  and  $\mathcal{V}$  are *orthogonal*, denoted  $\mathcal{U} \perp \mathcal{V}$  if

$$u^T v = 0 \quad \forall u \in \mathcal{U}, v \in \mathcal{V}. \quad (1.28)$$

The *orthogonal projection* of a point onto a set  $\mathcal{V} \subseteq \mathcal{U}$  is defined as

$$P_{\mathcal{V}}(u) \triangleq \min_{v \in \mathcal{V}} d(u, v). \quad (1.29)$$

If  $\mathcal{U}$  and  $\mathcal{V}$  are vector spaces, then  $\text{Span}\{u - P_{\mathcal{V}}(u)\} \perp \mathcal{V}$ .

Four linear spaces are associated with a matrix  $U \in \mathbb{R}^{M \times N}$ : The *column space*  $\text{range } U$  or *range*, is the span of the columns of  $U$ . The *row space*  $\text{range } U^T$ , is the span of the columns of  $U^T$ . The *null space*  $\text{null } U$  is the space orthogonal to the row space of  $U$ , i.e.

$$\text{null } U \triangleq \{x \mid Ux = 0\}. \quad (1.30)$$

The *left null space*  $\text{null } U^T$  is the space orthogonal to the column space of  $U$ .

As  $\mathbb{R}^{M \times N}$  is trivially homeomorphic to the Euclidean space  $\mathbb{R}^{MN}$ , we can extend the norm of  $\mathbb{R}^N$  to matrices by defining the so-called *Frobenius norm*

$$\|U\|_F^2 \triangleq \sum_k \|u_k\|^2, \quad (1.31)$$

where  $u_k$  is the  $k$ -th column of  $U$ , and all the above definitions can be equivalently applied to sequences and functions of matrices.



## Chapter 2

---

# A distributed algorithm for the MAXVAR problem

---

This chapter is based on

- \* **C. Hovine** and A. Bertrand, “Distributed MAXVAR: Identifying common signal components across the nodes of a sensor network”, in *2021 29th European Signal Processing Conference (EUSIPCO)*, IEEE, 2021, pp. 2159–2163
- \* **C. Hovine** and A. Bertrand, “MAXVAR-based distributed correlation estimation in a wireless sensor network”, *IEEE Transactions on Signal Processing*, vol. 70, pp. 5533–5548, 2022

---

## Chapter abstract

---

The performance of most array signal processing tasks relies on the presence of correlation between sensor signals. In a wireless sensor network, where sensor nodes are spread out over a relatively large area, it is useful to identify nodes observing similar sensor signals and hence common phenomena, for example to partition the network according to the observed latent signals and corresponding correlation structure. This can be achieved via the so-called MAXVAR formulation of generalized canonical correlation analysis, which finds a low-dimensional subspace that highlights correlated signal components between multiple nodes' observed signal subspaces. The classical procedure for computing the solutions of MAXVAR consists in performing a generalized eigenvalue decomposition after collecting all the sensors' signals at a fusion center. However, this typically incurs high communication and computational costs. In this chapter, we describe a low communication and computational cost distributed algorithm that computes the solutions of MAXVAR without aggregating the nodes' observations at a central location. We show how a subset of those solutions can be used locally by each node to estimate the global correlation structure across all nodes in the network, thereby allowing any node to evaluate the presence of correlated signals at any other node, even if no direct link is shared. We prove the convergence of the algorithm and validate our method for estimating the correlation structure via simulations.

## 2.1 Introduction

Array signal processing tasks such as signal estimation, filtering, or subspace estimation, generally boil down to the extraction of specific signal components, often split across many sensor signals. Due to the presence of sensing noise and the spatial distribution of the underlying signal sources, different nodes typically observe different but correlated signals, some sharing common components and others not. It is therefore of great importance to identify which node pairs share a common latent signal subspace and which do not, as this knowledge can be used to prune the network or cluster the nodes according to the similarity between their signal subspaces, resulting in significant bandwidth and computational complexity reduction at each node [75], [76].

In this chapter, we leverage the fact that the so-called principal angles derived from the solutions of the canonical correlation analysis (CCA) problem can be used to quantify the similarity between two nodes' sensor signal subspaces [77], [78]. Indeed, CCA can be used to estimate the signal components that are maximally correlated between two different nodes. It is closely related to principal component analysis [30] and the Karhunen–Loève transform [31], which extract the highest power components, yet not necessarily observed by both nodes. As CCA is only applicable to identify the most correlated signal components within the signal subspaces of two nodes, applying it to a network of more than two nodes would result in a combinatorial complexity scaling. In this chapter, we show that one of its multi-set generalizations, the so-called “Maximum Variance” (MAXVAR) generalization [79]–[81], can be used to approximate the solutions of the pairwise CCA problems and therefore the complete correlation structure of the network with a complexity that scales linearly with the network's size. MAXVAR has indeed been shown to provide a description of the intersection between multiple subspaces [82], which fits with the general purpose of finding “shared” signals subspaces across multiple nodes of a WSN.

There exist multiple multi-set generalizations of CCA, each characterized by a specific objective function and set of constraints [81]. The MAXVAR formulation historically refers to the objective function introduced by Horst [79]. Carroll [80] later introduced a new set of constraints for this objective which turns the formulation into an easily interpretable subspace decomposition and on which we focus in this chapter.

As using MAXVAR in a centralized fashion would hinder our progress towards bandwidth and complexity reduction, we present a distributed and adaptive algorithm for tracking the solutions of MAXVAR, relying on the exchange of compressed signals between the nodes. We describe variants of this distributed

MAXVAR (D-MAXVAR) algorithm for fully connected, star-topology and tree-topology networks, and use the later as a basis for extending the algorithm to arbitrary network topologies. We then explain how its solution can be used to evaluate the pair-wise correlation between any pair of nodes in the network, even if they are not connected via a direct link.

Previous works have investigated similar distributed subspace decompositions, targeting the union of the nodes' subspaces by extracting the components of greatest power [46], [55], maximal SNR [6], [57] or maximizing correlation between two sets of signals [49], [83]. In this work, we focus on finding the components that most adequately describe the inter-node correlation coefficient, also known as Pearson's correlation coefficient. Distributed algorithms already exist for the so-called SUMCORR generalization of CCA [48] but, to our knowledge, no distributed algorithm has been developed for MAXVAR, which, contrarily to SUMCORR, admits an analytical solution. In addition, the focus of [48] was on computational efficiency, rather than on bandwidth scaling, as is the case of the present work. Finally, as an alternative to distributed methods, several works investigate the applicability of CCA-like techniques to scenarios involving two sensor arrays and where only a limited number of samples is available at each node [84]–[86].

This chapter is organized as follows. In Section 2.2, we cover the preliminaries and algebraic concepts related to MAXVAR and explain how it can be used in the context of WSNs. In Section 2.3, we describe a distributed MAXVAR algorithm for fully connected networks and discuss its convergence properties. The algorithm is extended to general topologies in Section 2.4. Section 2.5 discusses how a specific subset of the solutions of MAXVAR can be used by each node to estimate the inter-node correlation structure, and in particular evaluate the degree to which each node's signals correlate with any other node, even several hops away. In Section 2.6, we assess the algorithm's performance through various simulations. We conclude the chapter by a brief discussion in Section 2.7.

## 2.2 MAXVAR

We wish to characterize the intensity of correlations between the signals of any two nodes. The so-called *canonical correlation coefficients* provide such a characterization, which can be found by means of canonical correlation analysis [87]. Considering two multi-channel signals  $\mathbf{y}_k$  and  $\mathbf{y}_l$  associated with two nodes  $k, l \in \mathcal{K}$ , CCA computes spatial filters  $x_k$  and  $x_l$  that maximize the correlation coefficient  $\rho^{kl}$  between their output signals  $z_k = x_k^T \mathbf{y}_k$  and  $z_l = x_l^T \mathbf{y}_l$ . The

output signals  $z_l$  and  $z_k$  are referred to as the first canonical directions, and  $\rho^{kl}$  is referred to as the first canonical correlation coefficient. Additional canonical directions and coefficients can be found by computing additional pairs of spatial filters that maximize the correlation between their outputs while having their outputs remain uncorrelated (orthogonal) to the previous canonical directions. Formally, the  $i$ -th canonical correlation coefficient  $\rho_i^{kl}$  and *canonical directions*  $z_{k,i}, z_{l,i}$  are defined as

$$\rho_i^{kl} = \mathbb{E} \{z_{k,i} z_{l,i}\} = \max_{x_{k,i}, x_{l,i}} \mathbb{E} \{x_{k,i}^T \mathbf{y}_k \mathbf{y}_l^T x_{l,i}\} \quad (2.1a)$$

$$\text{s.t.} \quad z_{k,i} = x_{k,i}^T \mathbf{y}_k, \quad z_{l,i} = x_{l,i}^T \mathbf{y}_l \quad (2.1b)$$

$$\mathbb{E} \{z_{k,i}^2\} = \mathbb{E} \{z_{l,i}^2\} = 1 \quad (2.1c)$$

$$\mathbb{E} \{z_{k,i} [z_{k,1}, \dots, z_{k,i-1}]\} = 0 \quad (2.1d)$$

$$\mathbb{E} \{z_{l,i} [z_{l,1}, \dots, z_{l,i-1}]\} = 0 \quad (2.1e)$$

The three last constraints require the canonical directions to have unit-variance and be orthogonal with respect to each other within each node. Note that  $\rho_i^{kl} = 1$  implies that the pairs  $(z_{k,i}, z_{l,i})$  span the exact intersection between the sensor signal subspaces of nodes  $k$  and  $l$ .

If we now express the set of canonical directions of node  $k$  with respect to node  $l$  as

$$\mathbf{z}_k = [z_{k,1}, \dots, z_{k,M_{kl}}]^T = X_k^T \mathbf{y}_k \quad (2.2)$$

with  $M_{kl} = \min(M_k, M_l)$  and  $X_{kl} \in \mathbb{R}^{M_k \times M_{kl}}$ , it can be shown [88] that the canonical directions and coefficients between two nodes  $k$  and  $l$  are the solutions of the following generalized eigenvalue problem:

$$\begin{bmatrix} O & R_{\mathbf{y}_k \mathbf{y}_l} \\ R_{\mathbf{y}_l \mathbf{y}_k} & O \end{bmatrix} \begin{bmatrix} X_k \\ X_l \end{bmatrix} = \begin{bmatrix} R_{\mathbf{y}_k} & O \\ O & R_{\mathbf{y}_l} \end{bmatrix} \begin{bmatrix} X_k \\ X_l \end{bmatrix} \Lambda \quad (2.3a)$$

$$\begin{bmatrix} X_k^T & X_l^T \end{bmatrix} \begin{bmatrix} R_{\mathbf{y}_k} & O \\ O & R_{\mathbf{y}_l} \end{bmatrix} \begin{bmatrix} X_k \\ X_l \end{bmatrix} = I \quad (2.3b)$$

where  $R_{\mathbf{y}_k}$  is a shorthand notation for  $R_{\mathbf{y}_k \mathbf{y}_k}$  and  $\Lambda$  is a diagonal matrix whose diagonal entries are non-negative and correspond to the canonical correlation coefficients. As a result, the optimal solution of (2.1) is given by the generalized eigenvectors corresponding to the largest generalized eigenvalues. By expressing the canonical directions via the parametrization introduced by (2.2), CCA can indeed be seen as the problem of finding two sets of spatial filters whose corresponding outputs have maximal total correlation.

## 2.2.1 Total squared correlation

From the definition of canonical correlation coefficients, we can define the *total squared correlation* (TSC)

$$\Theta_{kl} \triangleq \sum_{i=1}^{M_{kl}} (\rho_i^{kl})^2 = \text{Tr}(\Lambda_{kl}^2), \quad (2.4)$$

where  $\text{Tr}(\cdot)$  denotes the trace operator. (2.4) can be interpreted as a generalization of correlation between random variables to correlation between sets of random variables. Indeed, if we denote  $\mathcal{X}_k = \text{Span } \mathbf{y}_k$ , i.e. the signal subspace spanned by the channels of  $\mathbf{y}_k$ , we have:

$$\Theta_{kl} = 0 \Leftrightarrow \mathcal{X}_k \perp \mathcal{X}_l \quad (2.5)$$

$$\Theta_{kl} = M_{kl} \Leftrightarrow \mathcal{X}_k \subseteq \mathcal{X}_l \quad (2.6)$$

assuming  $\dim(\mathcal{X}_k) \leq \dim(\mathcal{X}_l)$ . Note that in the one-dimensional case ( $M_k = M_l = 1$ ), the TSC reduces to the usual squared correlation coefficient. Contrarily to correlation, the TSC is not concerned with the direction of the relationship (i.e. positively or negatively correlated), only the absolute magnitude of the relationship is captured. The TSC is positive and symmetric but does not satisfy the triangle inequality and is therefore not a distance [89]. Still, in the case where the channels of  $\mathbf{y}_k$  and  $\mathbf{y}_l$  are linearly independent (which is in practice always true in noisy settings), it can easily be turned into one by defining

$$d_{kl} = \sqrt{\max\{M_k, M_l\} - \Theta_{kl}} \quad (2.7)$$

which satisfies the triangle equality [90] and corresponds to the so-called Chordal distance in the particular case where  $M_k = M_l$  [91]. Considering these above properties, the TSC is an adequate metric for characterizing the degree to which the signals of two nodes correlate, which is one of the objectives of this chapter. In a WSN context, the TSC could for example be used as the key ingredient to an adaptive network where links between low-TSC node pairs are pruned, while more bandwidth is allocated to links between high-TSC node pairs. Similarly, the TSC could be used to define a weighted graph describing the correlation structure of the network, and on which spectral clustering techniques could be applied [92].

Computing all the pairwise TSCs between the nodes would require us to solve a CCA problem for every node pair. In this paper, we show that the full set of TSCs can be estimated by solving a single-instance of a problem related to CCA, described in the next subsection.

## 2.2.2 MAXVAR: extension to more than two subspaces

Problem (2.1) can be generalized to more than two nodes or subspaces in various fashions. As it admits a closed form solution<sup>1</sup>, we consider here the so-called *Maximum-Variance* (MAXVAR) generalization of the CCA problem [80], which can intuitively be derived as follows for  $K$  signal subspaces. Let  $R_D \triangleq \text{Blkdiag}(R_{\mathbf{y}_1}, \dots, R_{\mathbf{y}_K})$  be the block diagonal matrix containing the node-specific covariance matrices, then we can naturally extend the formulation (2.3) to

$$(R_{\mathbf{y}\mathbf{y}} - R_D)X = R_D X \Lambda \quad (2.8a)$$

$$X^T R_D X = I_Q \quad (2.8b)$$

where  $X \triangleq [X_1^T \dots X_K^T]^T$  is the matrix obtained by stacking the  $X_k$ 's and  $Q$  is the number of desired components. The above formulation is equivalent to

$$\max_{x_{k,i}, x_{l,i}} \sum_k \sum_{l \neq k} \mathbb{E} \{ x_{k,i}^T \mathbf{y}_k \mathbf{y}_l^T x_{l,i} \} \quad (2.9a)$$

$$\text{s.t.} \quad z_{k,i} = x_{k,i}^T \mathbf{y}_k, \quad z_{l,i} = x_{l,i}^T \mathbf{y}_l \quad (2.9b)$$

$$\frac{1}{K} \sum_k \mathbb{E} \{ z_{k,i}^2 \} = 1 \quad (2.9c)$$

$$\frac{1}{K} \sum_k \mathbb{E} \{ z_{k,i} [z_{k,1}, \dots, z_{k,i-1}] \} = 0, \quad (2.9d)$$

and the norm and orthogonality constraints on the  $z_{k,i}$  are now only required to hold on average across the nodes. It can be shown that this formulation reduces to (2.1) when  $K = 2$  [93].

In this paper, we follow an alternate, but equivalent (in the sense that the solutions span the same subspace), formulation that admits an intuitive interpretation. Initially described by Carroll [80], it is defined as follows:

$$\min_{\{X_k\}} \min_{\mathbf{s}} \sum_{k=1}^K \mathbb{E} \{ \|\mathbf{s} - \mathbf{z}_k\|^2 \} \quad (2.10a)$$

$$\text{s.t.} \quad \mathbf{z}_k = X_k^T \mathbf{y}_k \quad (2.10b)$$

$$\mathbb{E} \{ \mathbf{s} \mathbf{s}^T \} = I_Q \quad (2.10c)$$

---

<sup>1</sup>Which is not the case for SUMCORR, discussed earlier.

From the parametrization introduced by constraint (2.10b), the problem becomes equivalent to finding a  $Q$ -outputs filter  $X_k \in \mathbb{C}^{M_k \times Q}$  per node such that the filtered observations  $\mathbf{z}_k$ , which we refer to as the per-node *MAXVAR directions*, are as close as possible to some common network-wide  $Q$ -dimensional signal  $\mathbf{s}$ , and hence as close as possible to each other. From (2.10), it is clear that the per-node MAXVAR directions are the orthogonal projections of  $\mathbf{s}$  onto the nodes' signal subspaces  $\text{Span}(\mathbf{y}_k)$ . Hence,  $\text{Span}(\mathbf{s})$  is the  $Q$ -dimensional subspace whose signals have minimal average projection error onto the nodes' individual signal subspaces.

In [94], it is shown that the solution of (2.10) satisfies

$$\mathbf{s} = \frac{1}{K} \sum_k \mathbf{z}_k = X^T \mathbf{y} \quad (2.11)$$

Substituting (2.11) in (2.10a), the objective can be reformulated as

$$\min_{\{X_k\}} \sum_{k,l=1}^K \mathbb{E} \left\{ \|X_l^T \mathbf{y}_l - X_k^T \mathbf{y}_k\|^2 \right\}. \quad (2.12)$$

The problem therefore consists in finding the set of node-specific filters whose outputs are as close to each other as possible in a minimum squared error sense. Furthermore, it is also shown in [94] that the  $X_k$ 's are solutions to the following eigenvalue problem:

$$R_D X = R_{\mathbf{y}\mathbf{y}} X \Lambda \quad (2.13a)$$

$$X^T R_{\mathbf{y}\mathbf{y}} X = I_Q \quad (2.13b)$$

where  $\Lambda$  is a diagonal matrix. Solving problem (2.10) is therefore equivalent to computing the generalized eigenvalue decomposition (GEVD) of the matrix pencil  $(R_D, R_{\mathbf{y}\mathbf{y}})$ , keeping only the generalized eigenvectors (GEVC) corresponding to the  $Q$  smallest generalized eigenvalues (GEVL). One can check that the solution of (2.13) indeed spans the same subspace as the solution of (2.8) [95].

### 2.2.3 Problem statement

For each pair of nodes, we wish to assess how “close” their observed signal subspaces are to each other. This can be achieved by computing the TSC as described by (2.4). However, this approach has two major drawbacks. Firstly, it requires the computation of  $R_{\mathbf{y}_k \mathbf{y}_l}$  which is only possible if the full observations  $\mathbf{y}_k$  and  $\mathbf{y}_l$  are co-located at a single node (or an FC), therefore incurring high

communication cost. Secondly, a CCA solution needs to be computed for each of the  $K(K-1)/2$  pairs of nodes, further increasing the communication and computational cost of the procedure. As an efficient alternative, we propose to jointly approximate the TSCs between all node pairs using a distributed procedure based on MAXVAR. Indeed, we will show in Section 2.5.1 that solving a slightly modified version of the MAXVAR problem produces a joint approximation of the TSC of each node pair. As it is useful in its own right and of more general interest, we will first describe a distributed procedure for computing the solutions of the classical MAXVAR problem (2.13). In order to obtain such a solution, all nodes would typically need to share their observations to an FC where the full covariance matrix could be estimated. This requires a large communication bandwidth between the nodes and the FC, in particular if all nodes are not directly connected to the FC, which increases the stress on the communication links of the nodes that are close to the fusion center. In addition, such a centralized processing does not leverage the fact that the signal subspace in which inter-node correlation is present typically manifests a low dimension, and that its signals can therefore be efficiently described by only a few components.

In Section 2.3, we present a distributed algorithm for solving (2.13) in a distributed fashion and relying on the transmission of low-dimensional compressed views of the data between neighboring nodes, thus lifting the need to transfer the raw  $M_k$ -channel sensor observations of all nodes to an FC through a possibly multi-hop network. Instead, the signal observations are directly fused with other observations within the network, and such that each node eventually have access to an estimate of the solutions, thereby avoiding the need for a fusion center altogether. In Section 2.5, we describe how the distributed MAXVAR procedure can be modified to produce an approximation of (2.4).

## 2.3 Distributed MAXVAR in fully-connected networks

In this section, we derive a distributed iterative algorithm for computing the  $Q$  first per-node MAXVAR directions associated with each node, and which is such that each node  $k$  eventually has access to the network-wide estimate of  $\mathbf{s}$  and its own per-node MAXVAR direction  $\mathbf{z}_k$  as defined in (2.10b). Relying on our interest in a subset of the components only (which is motivated in Section 2.5) and the particular problem structure, we show that neighboring nodes only need to share  $Q$ -dimensional compressed views of their observed signal

subspaces at each iteration, which eventually converge to their first  $Q$  per-node MAXVAR directions  $\mathbf{z}_k$ . In order to facilitate the reader's understanding and intuition in the algorithm development, we first derive the algorithm for the simpler case of fully-connected networks and later extend it to more general network topologies.

### 2.3.1 Algorithm derivation

In a fully-connected network, any node can communicate with any other node via a single hop. By denoting the set of neighbors of node  $k$  as  $\mathcal{N}_k$ , we have for such networks  $\mathcal{N}_k = \mathcal{K} \setminus \{k\}$ .

Considering the GEVD formulation (2.13), the problem of finding the  $Q$  first per-node MAXVAR directions is equivalent to finding the  $Q$  GEVCs of the matrix pencil  $(R_D, R_{\mathbf{y}\mathbf{y}})$  associated with its smallest GEVLs and corresponding to the columns of  $X$ . The algorithm iteratively updates the  $M_k \times Q$  matrices  $X_k^i$  (where the superscript  $i$  denotes the iteration index), which act as the local estimates of  $X_k$  and, as will be shown, as a node-specific compressor of the nodes' signals. It can be shown [96] that the GEVCs (corresponding to columns of  $X$ ) associated with the  $Q$  smallest GEVLs from the GEVD in (2.13) and hence the MAXVAR solutions of (2.10) coincide with the solution of the following trace minimization problem<sup>2</sup>

$$\min_{X \in \mathbb{R}^{M \times Q}} \text{Tr}(X^T R_D X) \quad (2.14a)$$

$$\text{s.t.} \quad X^T R_{\mathbf{y}\mathbf{y}} X = I_Q \quad (2.14b)$$

where  $\text{Tr}(\cdot)$  denotes the trace operator. The core idea of the algorithm is to have the nodes solve a local version of (2.14) in turns, and expressed in terms of the node's own raw observations and the other nodes' compressed observations  $\mathbf{z}_k^i = X_k^{iH} \mathbf{y}_k$  ( $X_k^{iH}$  denotes the compression matrix associated with the signals of node  $k$  at iteration  $i$  of the algorithm). The notation  $\mathbf{z}_k^i, X_k^{iH}$  for the compressed observations and compression matrices is chosen deliberately, as they also correspond to node  $k$ 's local estimate of its part of the solution of (2.10). We indeed expect that eventually

$$\lim_{i \rightarrow \infty} X_k^i = X_k \text{ and } \lim_{i \rightarrow \infty} \mathbf{z}_k^i = \mathbf{z}_k. \quad (2.15)$$

The algorithm in fully connected networks is as follows. At the beginning of each iteration, an updating node  $q$  is selected. Every other node transmits a

<sup>2</sup>Although MAXVAR is also often expressed as a trace maximization problem, we write it as a minimization problem to stay consistent with Carroll's formulation (2.10).

batch of its compressed observations  $\mathbf{z}_k^i \forall k \in \mathcal{N}_q$  to the updating node, such that it can locally solve the following problem:

$$\min_{\tilde{X}} \quad \text{Tr} \left( \tilde{X}^T R_{D_q}^i \tilde{X} \right) \quad (2.16a)$$

$$\text{s.t.} \quad \tilde{X}^T R_{\bar{\mathbf{y}}_q}^i \tilde{X} = I_Q \quad (2.16b)$$

where  $R_{\bar{\mathbf{y}}_q}^i$  is the covariance matrix of

$$\bar{\mathbf{y}}_q^i \triangleq \left[ \mathbf{y}_q^T \quad \mathbf{z}_1^i{}^T \quad \cdots \quad \mathbf{z}_{q-1}^i{}^T \quad \mathbf{z}_{q+1}^i{}^T \quad \cdots \quad \mathbf{z}_K^i{}^T \right]^T \quad (2.17)$$

and

$$R_{D_q}^i \triangleq \text{Blkdiag}(R_{\mathbf{y}_q}, R_{\mathbf{z}_1}, \dots, R_{\mathbf{z}_{q-1}}, R_{\mathbf{z}_{q+1}}, \dots, R_{\mathbf{z}_K}) \quad (2.18)$$

is a block diagonal matrix with  $R_{\mathbf{z}_k}^i = \mathbb{E} \left\{ \mathbf{z}_k^i \mathbf{z}_k^i{}^T \right\}$ . Note that the solution of (2.16) can again be found from a GEVD, this time applied to the pencil  $(R_{D_q}^i, R_{\bar{\mathbf{y}}_q}^i)$ . An update rule for the network-wide  $X^{i+1}$  emerges naturally by noticing that solving the local problem (2.16) is equivalent to solving the original centralized problem (2.14) with additional constraints:

$$\min_X \quad \text{Tr} (X^T R_D X) \quad (2.19a)$$

$$\text{s.t.} \quad X^T R_{\mathbf{y}_q} X = I_Q \quad (2.19b)$$

$$\text{range } X_k \subseteq \text{range } X_k^i \quad \forall k \neq q \quad (2.19c)$$

where the operator  $\text{range} \cdot$  denotes the column space of its argument. Indeed, notice that the range constraints in (2.19c) can be equivalently formulated by introducing the following parametrization of  $X$  in (2.14):

$$X = \begin{bmatrix} X_q \\ X_1^i \tilde{X}_1 \\ \vdots \\ X_{q-1}^i \tilde{X}_{q-1} \\ X_{q+1}^i \tilde{X}_{q+1} \\ \vdots \\ X_K^i \tilde{X}_K \end{bmatrix} \quad (2.20)$$

where the multiplication with  $\tilde{X}_k \in \mathbb{R}^{Q \times Q}$  allows to select a new  $X_k$  within the column space of  $X_k^i$ . With this parametrization, (2.19a) becomes

$$\text{Tr}(X^T R_D X) = \text{Tr}(X_q^T R_{y_q} X_q) + \sum_{k \neq q} \text{Tr} \left( \tilde{X}_k^T \underbrace{X_k^{iT} R_{y_k} X_k^i}_{=R_{z_k}^i} \tilde{X}_k \right). \quad (2.21)$$

Finally introducing  $\tilde{X}$  as

$$\tilde{X} = \begin{bmatrix} X_q \\ \tilde{X}_1 \\ \vdots \\ \tilde{X}_{q-1} \\ \tilde{X}_{q+1} \\ \vdots \\ \tilde{X}_K \end{bmatrix} \quad (2.22)$$

and substituting in (2.21), we indeed obtain the objective of the local problem (2.16a). A similar derivation can be done for the constraints (2.19c). The parametrization therefore indeed defines the update rule for the local estimate of  $X_k$  at each node:

$$X_k^{i+1} = \begin{cases} X_q & \text{if } k = q \\ X_k^i \tilde{X}_k & \text{if } k \neq q \end{cases} \quad (2.23)$$

where  $X_q$  and  $\tilde{X}_k$  are extracted from the solution of (2.16) based on the partitioning (2.22). Note that the range constraints make (2.19) different from a traditional nonlinear Gauss-Seidel approach where all  $X_k$  would be fixed to  $X_k^i$  for  $k \neq q$ , in which case (2.19) could not be solved as a GEVD anymore.

Concretely, the procedure at each iteration can be divided into three steps (the detailed procedure is formally described in Algorithm 2.1):

**(i) Aggregation** Each node  $k \in \mathcal{N}_q$  sends a new batch of  $T$  observations of the locally compressed signals  $z_k^i$  to the updating node  $q$ .

**(ii) Local solution** The updating node  $q$  solves problem (2.16), expressed exclusively in terms of locally available data. From the solution, it extracts the update of its own local estimate of the solution  $X_q$  as well as the update matrices  $\tilde{X}_k$  corresponding to each other node, as defined by the partitioning

(2.22). Due to the sign ambiguities of the GEVCs<sup>3</sup>, the solution of (2.16) is not unique and independent from the sign changes of its columns. Therefore, in order to ensure convergence, we select the solution resulting in the smallest difference  $\|X^{i+1} - X^i\|_F$ . In practice, this amounts to checking whether

$$\|\tilde{x}^{i+1} - \tilde{x}^i\|_F < \|\tilde{x}^{i+1} + \tilde{x}^i\|_F$$

for each of the columns  $\tilde{x}^{i+1}$  and  $\tilde{x}^i$  of  $\tilde{X}$  and  $[X_q^{iT} I_Q \cdots I_Q]^T$ , respectively, and choosing the signs of the columns accordingly.

**(iii) Update** Node  $q$  sends the update matrices  $\{\tilde{X}_k\}_{k \neq q}$  to each node, which update their local estimates  $X_k^i$  according to (2.23). Note that the transmission cost of these  $Q \times Q$  update matrices is negligible compared to the transmission costs in the aggregation step (assuming  $T \gg Q^2$ ). The role of the updating node is finally passed on to node  $(q + 1) \bmod K$ . If required, the common components  $\mathbf{s}$  can be estimated at any iteration by performing an in-network summation:

$$\mathbf{s}^i = \frac{1}{K} \sum_k z_k^i. \quad (2.24)$$

It is noted that the updating node  $q$  has access to all the data required to compute (2.24), such that no additional bandwidth is required to compute  $\mathbf{s}$  at node  $q$ .

**Remark 2.1.** The batch of  $T$  (compressed) samples that is transmitted by each node during the aggregation step would typically consist of different samples than the ones used in the previous iteration, in order to avoid retransmitting the same data multiple times which would substantially increase the bandwidth. Therefore, the time index  $t$  corresponding to the first sample of a batch is typically updated as  $t + iT'$  at each iteration (with  $T'$  possibly smaller than  $T$ ). This implies that the iterations are spread out over time, and that the algorithm behaves as an adaptive filter tracking the signal statistics over time. In order to make the convergence analysis mathematically tractable, we therefore have to assume that the signal statistics change sufficiently slowly compared to the convergence dynamics of the algorithm (see Assumption 1.2).

---

<sup>3</sup>Multiple solutions could also appear due to a potential (and extremely unlikely) collapse of the eigenspace resulting from GEVLs with multiplicity larger than 1, but the update should in this case be skipped as it violates Assumption 2.1 (described in the next subsection).

---

**Algorithm 2.1:** D-MAXVAR algorithm in a fully connected network.

---

```

begin
   $i \leftarrow 0$ 
   $q \leftarrow 1$ 
  Randomly initialize  $X^0$ 
  loop
    for  $k \in \mathcal{K} \setminus \{q\}$  do
      At node  $k$ 
      | Send a new batch of  $T$  samples  $\mathbf{z}_k^i(t) = X_k^{iT} \mathbf{y}_k(t)$  to node  $q$ 
    At node  $q$ 
    | Compute  $R_{\mathbf{y}_q}^i$  and  $R_{D_q}^i$  based on the received samples and
    |   according to (2.17) and (2.18)
    | Compute the  $Q$  GEVCs corresponding to the  $Q$  smallest GEVLs
    |   of the matrix pencil  $(R_{D_q}^i, R_{\mathbf{y}_q}^i)$ , scaled to satisfy constraint
    |   (2.16b) and minimizing  $\|X^{i+1} - X^i\|_F$ 
    | Put the  $Q$  resulting GEVCs in the columns of  $\tilde{X}_q$ 
    |  $X_q^{i+1} \leftarrow [I_{M_q} \ O] \tilde{X}_q$ 
    | for  $k \in \mathcal{K} \setminus \{q\}$  do
    | | Select  $\tilde{X}_k$  as the block of  $\tilde{X}_q$  (see (2.22)) corresponding to
    | |   node  $k$  and send to node  $k$ 
    | | At node  $k$ 
    | | |  $X_k^{i+1} \leftarrow X_k^i \tilde{X}_k$ 
    |  $i \leftarrow i + 1$ 
    |  $q \leftarrow (q \bmod K) + 1$ 

```

---

### 2.3.2 Convergence and optimality

In this subsection, we provide some insight in the convergence and optimality properties of the D-MAXVAR algorithm, and establish formal convergence proofs.

A first important observation is that the solution of (2.19) at iteration  $i$  is by definition in the constraint set of the problem at iteration  $i + 1$  (corresponding to selecting  $\tilde{X}_k = I_Q$ ). As a result, the objective function (2.19a) decreases monotonically and must therefore converge, as the objective function is bounded in the constraint set. However, the convergence of the sequence of optimization variables  $(X^i)_{i \in \mathbb{N}}$  is less straightforward. Indeed, monotonic convergence of the objective function does not imply convergence of its arguments, nor that the

global minimum is eventually attained. Nonetheless, by making an assumption which is always satisfied in practice, we can show several interesting properties about the convergence behavior and limit points of the algorithm.

**Assumption 2.1.** *The accumulation points of the sequence of pencils  $\left( (R_{D_q}^i, R_{\mathbf{y}_q}^i) \right)_{i \in \mathbb{N}}$  are non singular and have distinct  $Q$ -th and  $(Q + 1)$ -th smallest GEVLs.*

Due to the presence of uncorrelated sensor noise, the non-singularity assumption is always verified. The assumption about distinct eigenvalues is merely technical, as we can show that such points corresponding to degenerate local problems would be unstable in practice, unless they correspond to the global solution of (2.14) (see Theorem 2.3 below), and the algorithm would therefore eventually diverge from such points.

In what follows, a fixed point  $X^*$  is defined as a point that is invariant under the updates of Algorithm 2.1, i.e.,  $(X^i)_{i \in \mathbb{N}} = (X^*)_{i \in \mathbb{N}}$  if  $X^0 = X^*$ . The following theorem gives an important characterization of the algorithm's fixed points:

**Theorem 2.1.** *The columns of matrices which are fixed points of Algorithm 2.1 are stationary points of problem (2.14) and therefore GEVCs of the pencil  $(R_D, R_{\mathbf{y}\mathbf{y}})$ .*

*Proof.* See Appendix A.1. □

Assumption 2.1 guarantees that the local problems have well-defined solutions at accumulation points of the algorithm, allowing us to state our main convergence result:

**Theorem 2.2 (Convergence).** *If Assumption 2.1 holds,  $(X^i)_{i \in \mathbb{N}}$  converges to a fixed point, and hence stationary point, of problem (2.14).*

*Proof.* See Appendix A.2. □

Finally, we show that the global minimizers of (2.14) are the only stable fixed

points. In other words, all convergence trajectories to limit cycles or stationary points where (2.14) is not minimized are unstable in the sense that the algorithm can be kicked out of such trajectories by infinitesimally small perturbations. This is formalized in the following theorem:

**Theorem 2.3** (Unstable Accumulation Points). *Let  $X^*$  be an accumulation point of Algorithm 2.1. Then  $X^*$  is an unstable accumulation point if and only if it is not a global minimizer of problem (2.14).*

*Proof.* See Appendix A.3. □

Therefore, in the presence of numerical noise, we expect the algorithm to converge to a global minimizer of problem (2.14), as demonstrated by the simulations performed in Section 2.6.

## 2.4 Distributed MAXVAR in arbitrary network topologies

Before describing the D-MAXVAR algorithm for more general topologies, we first explain how it can be established in a star topology, thereby introducing some important insights towards further extensions.

### 2.4.1 Star-topology networks

In a star-topology network, we can distinguish two kinds of nodes: the central node  $k_c$ , which shares a link with all other nodes in the network, and the leaf nodes  $k \in \mathcal{L} = \mathcal{K} \setminus \{k_c\}$ , which are exclusively connected to the central node. Therefore,

$$\mathcal{N}_{k_c} = \mathcal{L} \tag{2.25}$$

$$\mathcal{N}_k = \{k_c\} \quad \forall k \in \mathcal{L}. \tag{2.26}$$

A naive strategy to apply the algorithm presented above for fully connected networks to a star-topology network would be to let the central node act as a relay between the leaf nodes. We discard this solution for two reasons: Firstly, the bandwidth required at the central node would grow linearly with

the number of nodes in the case of a broadcast communication protocol or quadratically in the case of one-to-one communication. The maximum network size would therefore largely depend on the bandwidth available at the central node. Secondly, further bandwidth savings can be achieved by allowing the central node to compress and fuse the signals it receives from the leaf nodes.

We will apply a separate treatment to the iterations where the updating node is the central node and those where it is a leaf node, of which we give a brief overview hereafter:

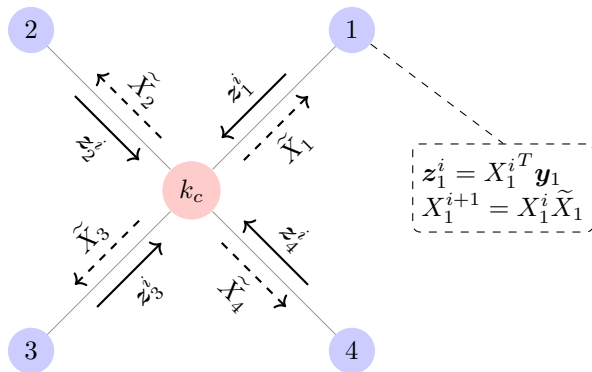
**The updating node is the central node ( $q = k_c$ )** As all nodes share a link with the updating node, the network proceeds as in the fully-connected case. All nodes send their compressed observations  $\mathbf{z}_k^i$  to the central node which then solves (2.16). The leaf nodes update their local estimates of the solution as in the fully-connected case.

**The updating node is a leaf node ( $q \neq k_c$ )** As only the central node shares a link with the updating node, it collects the compressed observations of the other leaf nodes. It fuses (i.e. adds) them together with its own observations and sends them to the updating node. The updating node now acts *as if the data received from the central node were the compressed observations of a single node* and proceeds to compute its local solution estimate and the *single* update matrix  $\tilde{X}_{k_c}$  for the central node accordingly. The central node then relays the update matrix  $\tilde{X}_{k_c}$  to the other leaf nodes, which then all update their local estimate with this single update matrix. The data flow for star-topology networks is illustrated in Figure 2.1. Note that the computation of the local solution at node  $q$  also requires aggregating and fusing the second order signal statistics through the network similarly to the signal observations, as will be explained below.

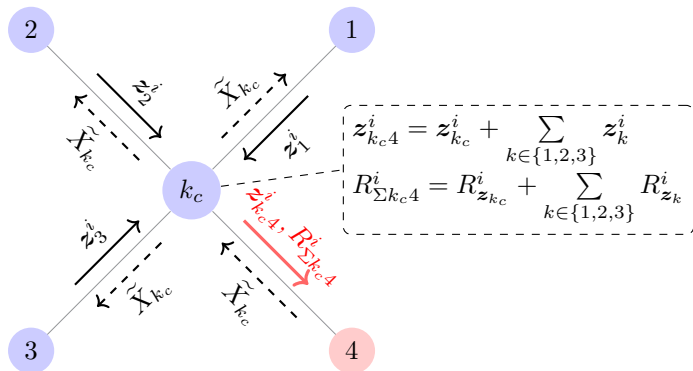
Formally, for a star-topology network, when the updating node is a leaf node  $q \in \mathcal{L}$ , node  $q$  receives the following from the central node:

$$\mathbf{z}_{k_c q}^i \triangleq \mathbf{z}_{k_c}^i + \sum_{k \in \mathcal{N}_{k_c} \setminus q} \mathbf{z}_k^i \quad (2.27)$$

$$R_{\Sigma k_c q}^i \triangleq R_{\mathbf{z}_{k_c}}^i + \sum_{k \in \mathcal{N}_{k_c} \setminus q} R_{\mathbf{z}_k}^i \quad (2.28)$$



(a) The updating node is the center node ( $q = k_c$ ). As in the fully connected case, one update matrix  $\tilde{X}_k^{i+1}$  is computed for each leaf node.



(b) The updating node is a leaf node ( $q = 4$ ). A single update matrix  $\tilde{X}_{k_c}^{i+1}$  is computed.

Figure 2.1: D-MAXVAR data flow in a star-topology network with  $K = 5$

It then constructs

$$\bar{\mathbf{y}}_q^i \triangleq \left[ \mathbf{y}_q^T \quad \mathbf{z}_{k_c q}^i{}^T \right]^T \quad (2.29)$$

$$R_{D_q}^i \triangleq \text{Blkdiag}(R_{\mathbf{y}_q}, R_{\Sigma_{k_c q}}^i) \quad (2.30)$$

as it did in the fully-connected case according to (2.17)–(2.18), and solves problem (2.16). With those new definitions, we can show that solving the local problem (2.16) is equivalent to solving

$$\min_X \quad \text{Tr}(X^T R_D X) \quad (2.31a)$$

$$\text{s.t.} \quad X^T R_{\mathbf{y}_q} X = I_Q \quad (2.31b)$$

$$\text{range } X_{-q} \subseteq \text{range } X_{-q}^i \quad \text{if } q \in \mathcal{L} \quad (2.31c)$$

$$\text{range } X_k \subseteq \text{range } X_k^i \quad \forall k \in \mathcal{L} \quad \text{if } q \notin \mathcal{L} \quad (2.31d)$$

where  $X_{-q}$  is the matrix obtained by removing the rows of  $X$  corresponding to  $X_q$ . Note that when the updating node is the central node (i.e.  $q \notin \mathcal{L}$ ), problem (2.31) reduces to the fully connected case problem (2.19). When  $q \in \mathcal{L}$ , constraint (2.31c) results in the equivalent parametrization

$$X = \left[ X_q^T \quad (X_{-q}^i \tilde{X}_{k_c})^T \right]^T \quad (2.32)$$

where  $\tilde{X}_{k_c} \in \mathbb{R}^{Q \times Q}$ .

**Remark 2.2.** Note that the number of degrees of freedom is lower for iterations where the updating node  $q$  is a leaf node (constraint (2.31c) is active) than for iterations where the updating node  $q$  is the center node (constraint (2.31d) is active). We therefore generally expect a lower decrease of the objective function (on average) for iterations in which a leaf node is the updating node.

## 2.4.2 Tree-topology networks

A tree-topology network has an acyclic graph, which implies that there is a unique path between any two nodes. The nodes with a single neighbor are also referred to as leaf nodes and constitute the end points of the branches in the tree. As we did for star-topology networks, we denote the set of leaf

nodes  $\mathcal{L}$ . Note that the tree concept should not be viewed as an actual topology constraint, but instead as a framework to organize the data streams within the network, i.e., to define in which order nodes have to transmit their data through the network, which will be essential when extending our algorithm to general topologies.

The procedure in tree-topology networks is conceptually similar to the star-topology case, where the updating node behaves as if it were the center node of a star-topology network, as described in Section 2.4.1. Consider the two isolated subtrees obtained by disconnecting the updating node  $q$  from one of its neighbors  $k \in \mathcal{N}_q$ . We denote  $\mathcal{B}_{kq}$  the set of nodes in the subtree containing  $k$  (see Figure 2.2). At each iteration, each of the neighboring nodes  $k \in \mathcal{N}_q$  of node  $q$  recursively collects and sums the compressed observations and related covariance matrices of its respective subtree  $\mathcal{B}_{kq}$  and sends them to node  $q$ . The updating node  $q$  therefore receives

$$\mathbf{z}_{kq}^i \triangleq \sum_{l \in \mathcal{B}_{kq}} \mathbf{z}_l^i = \mathbf{z}_k^i + \sum_{l \in \mathcal{N}_k \setminus q} \mathbf{z}_{lk}^i \quad \forall k \in \mathcal{N}_q \quad (2.33)$$

$$R_{\Sigma_{kq}}^i \triangleq \sum_{l \in \mathcal{B}_{kq}} R_{\mathbf{z}_l}^i = R_{\mathbf{z}_k}^i + \sum_{l \in \mathcal{N}_k \setminus q} R_{\Sigma_{lk}}^i \quad \forall k \in \mathcal{N}_q \quad (2.34)$$

Note that these definitions are recursive and that those values can be efficiently computed by performing an in-network summation in a recursive fashion. This recursive aggregation procedure is described by Algorithm 2.2 and illustrated in Figure 2.3. Note that even in nodes where  $Q > M_k$  (e.g. in case of single-channel sensor nodes) this aggregation process realizes an overall bandwidth reduction due to the in-network fusion of data (as opposed to straightforwardly relaying the raw data). Similarly to the other topologies, we define

$$\tilde{X}_q \triangleq \left[ X_q^T \mid \tilde{X}_{k_1}^T \mid \dots \mid \tilde{X}_{k_{n_q}}^T \right]^T \quad (2.35)$$

$$\bar{\mathbf{y}}_q^i \triangleq \left[ \mathbf{y}_q^T \mid \mathbf{z}_{k_1q}^{iT} \mid \dots \mid \mathbf{z}_{k_{n_q}q}^{iT} \right]^T \quad (2.36)$$

$$R_{D_q}^i \triangleq \text{Blkdiag}(R_{\mathbf{y}_q}, R_{\Sigma_{k_1q}}^i, \dots, R_{\Sigma_{k_{n_q}q}}^i) \quad (2.37)$$

with  $n_q = |\mathcal{N}_q|$  and  $\{k_1, \dots, k_{n_q}\} = \mathcal{N}_q$ . This allows us to solve the local problem (2.16) which is again equivalent to the global problem (2.14) equipped with additional range constraints, this time for each subtree  $\mathcal{B}_{(\cdot)}$ . A complete description of the procedure is given by Algorithm 2.3.

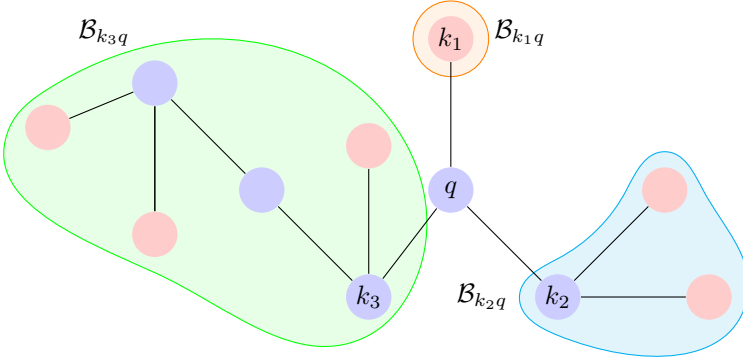


Figure 2.2: In this example tree, the subtree  $\mathcal{B}_{k_1q}$  is highlighted in orange,  $\mathcal{B}_{k_2q}$  in blue and  $\mathcal{B}_{k_3q}$  in green. Leaf nodes are colored red.

---

**Algorithm 2.2:** Recursive procedure for aggregating observations in the branch  $\mathcal{B}_{kp}$  to obtain  $\mathbf{z}_{kp}$

---

**procedure** `aggregate`( $k, p$ )

**for**  $l \in \mathcal{N}_k \setminus \{p\}$  **do**

`aggregate`( $l, k$ )

**At node**  $k$

**if**  $k \notin \mathcal{L}$  **then**

      Compute  $\mathbf{z}_{kp}[t] = \mathbf{z}_k[t] + \sum_{l \in \mathcal{N}_k \setminus p} \mathbf{z}_{lk}[t]$  from a new batch of  $T$

      samples  $\mathbf{z}_k^i[t] = X_k^{iT} \mathbf{y}_k[t]$

$R_{\Sigma_{kp}} = R_{\mathbf{z}_k} + \sum_{l \in \mathcal{N}_k \setminus p} R_{\Sigma_{lk}}$

      Send  $(\mathbf{z}_{kp}, R_{\Sigma_{kp}})$  to node  $p$

**else**

      Send  $\mathbf{z}_{kp} = \mathbf{z}_k$  to node  $p$

---

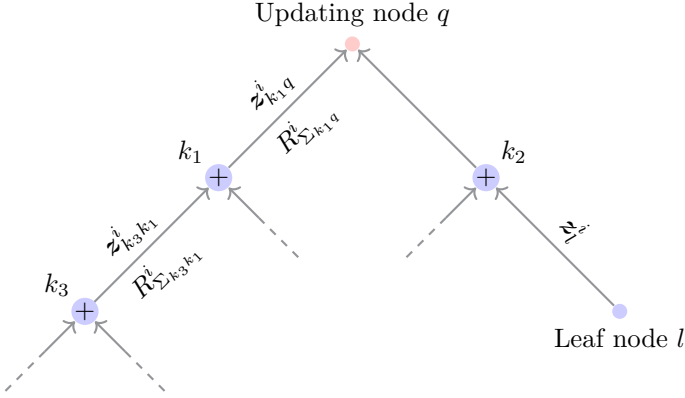


Figure 2.3: In-network summing and aggregation at the updating node. Covariance matrices and compressed observations are recursively propagated from the leaves towards the updating node.

---

**Algorithm 2.3:** D-MAXVAR algorithm in a tree-topology network.

---

**begin**

$i \leftarrow 0$

Initialize updating node as  $q \leftarrow 1$

Randomly initialize the  $X_k^0$ 's

**loop**

**for**  $k \in \mathcal{N}_q$  **do**

$\lfloor$  **aggregate**( $k, q$ ) (see Algorithm 2.2)

**At node**  $q$

    Compute the  $Q$  GEVCs corresponding to the  $Q$  smallest GEVLs of the matrix pencil  $(R_{D_q}^i, R_{y_q}^i)$ , scaled to satisfy constraint

    (2.16b) and minimizing  $\|X^{i+1} - X^i\|_F$

    Put the  $Q$  resulting GEVCs in the columns of  $\tilde{X}_q$

$X_q^{i+1} \leftarrow [I_{M_q} \ O] \tilde{X}_q$

**for**  $k \in \mathcal{N}_q$  **do**

        Select  $\tilde{X}_k^{i+1}$  as the block of  $\tilde{X}_q$  (see (2.35)) corresponding to node  $k$  and disseminate within subtree  $\mathcal{B}_{kq}$

**for**  $l \in \mathcal{B}_{kq}$  **do**

**At node**  $l$

$\lfloor$   $X_l^{i+1} \leftarrow X_l^i \tilde{X}_k$

$i \leftarrow i + 1$

$q \leftarrow (q \bmod K) + 1$

---

Finally, note that similarly to (2.24), the common components in  $\mathbf{s}$  can be estimated as

$$\mathbf{s}^i = \frac{1}{K} \sum_k \mathbf{z}_k^i. \quad (2.38)$$

### 2.4.3 Arbitrary networks

It is in tree-topology networks that our algorithm offers the greatest benefits in terms of both complexity and communication costs. Even though tree-topology networks are quite specific, the D-MAXVAR algorithm can be generalized to networks with arbitrary topologies by overlaying it with a topology management layer responsible for designing and maintaining a tree-topology at each iteration (assuming the initial graph is connected). Ideally, such a tree should preserve all the neighbors of the updating node  $q$ . Indeed, cutting off neighbors would reduce the number of update matrices  $\tilde{X}_{(\cdot)}$  in (2.35), which would reduce the degrees of freedom in the minimization of (2.16), using the definitions in (2.35)–(2.37). Algorithm 2.4 describes a simple distributed procedure for constructing a tree rooted at node  $q$  and preserving the links to its neighbors. It has a per-node message complexity at node  $k$  of  $\mathcal{O}(\mathcal{N}_k)$  and an overall time complexity of  $\mathcal{O}(D)$  where  $D$  is the diameter of the network (i.e. longest shortest-path between two-nodes). The procedure is as follows. Each neighbor of node  $q$  sends a message to each of its neighbors offering to become its parent in the tree. A node accepts to become a child if it does not yet have a parent. Once it has a parent, it does the same with its own neighbors (its parent excluded). One can see that at each step of the algorithm, the nodes that have a parent are part of a tree. The algorithm stops when each node has a parent, producing a tree spanning the entire network graph.

In practice, the data exchange required to construct such a tree is expected to be negligible compared to the exchange of raw sensor data in Algorithm 2.3, in which a whole batch of  $T$  multi-channel signal samples is transmitted by each node. Furthermore, we expect the network topology to be relatively static and change only after many iterations of the algorithm, as the objective function of the algorithm would otherwise not be relevant anymore. In that scenario, it is reasonable to only periodically compute a new spanning tree using either Algorithm 2.4 or a more elaborate distributed spanning tree algorithm such as those described in [97].

A similar convergence statement as for fully-connected networks can be made for tree networks and their extension to arbitrary networks, and the associated proof can be relatively straightforwardly adapted to these cases, although the notation and definitions become substantially more convoluted.

---

**Algorithm 2.4:** Distributed algorithm for computing a spanning tree preserving the neighbors of node  $q$ .

---

```

procedure buildtree( $q$ )
  for  $l \in \mathcal{K}$  do
    Initialize set of children of node  $l$ :  $\mathcal{L}_l \leftarrow \{\}$ 
    Initialize number of acknowledgment messages received:  $r_l \leftarrow 0$ 
  At node  $l \in \mathcal{N}_q$ 
    Set parent to node  $q$ :  $p_l \leftarrow q$ 
  At node  $l \neq q$ 
    while  $p_l$  is not set do
      Wait for message (ADOPT,  $m$ )
      Set parent to node  $m$ :  $p_l \leftarrow m$ 
      Send (OK,  $l$ ) to  $p_l$  as an acknowledgment
    Send (ADOPT,  $l$ ) to  $\mathcal{N}_l \setminus \{p_l\}$ 
    while  $r_l < |\mathcal{N}_l|$  do
      Wait for message ( $a, m$ )
      if  $a = ADOPT$  then
        Send (NOK,  $\cdot$ ) to node  $m$  to indicate that it already has a
        parent
      else
        if  $a = OK$  then
          Add  $m$  to set of children of  $l$ :  $\mathcal{L}_l \leftarrow \mathcal{L}_l + \{m\}$ 
         $r_l \leftarrow r_l + 1$ 

```

---

## 2.4.4 Complexity and communication cost

Table 2.1 summarizes the communication cost and complexity of our algorithm. The major benefit of the arbitrary-topology variant is that it scales well, i.e. the per-node communication cost and transmission cost is independent of the network size (as opposed to a naive multi-hop relay procedure, which would grow with the depth of the tree). This is because the sensor observations of the different nodes are fused along the way when the data travels through the network, as described in the aggregation step of Algorithm 2.2.

	Fully-connected	Arbitrary Topology
Transmission cost per node	$\propto Q$	$\propto Q$
Complexity at updating node $q$	$\propto (M_q + Q(K-1))^3$	$\propto (M_q + Q \mathcal{N}_q )^3$

Table 2.1: Communication cost and complexity of the D-MAXVAR algorithm

The above figures assume the transmission cost of the update matrices  $\tilde{X}_k$  and of the messages involved in the distributed spanning-tree procedure to be negligible compared to a batch of  $T$  samples.

## 2.5 Correlation structure estimation

In this section, we describe how the D-MAXVAR algorithm can be modified to efficiently approximate the TSC of all node pairs, even for a pair of nodes that do not directly exchange signals with each other. We first describe how the TSC relates to a low-rank approximation of a particular block-whitened correlation matrix (defined below). We then show how this low-rank approximation can be computed by computing a different subset of the per-node MAXVAR directions, instead of the ones corresponding to the  $Q$  smallest GEVLs. Finally, we describe how to select an appropriate value for  $Q$ .

### 2.5.1 MAXVAR as a low-rank approximation

The covariance matrix between node  $k$  and  $l$  of the (per-node) whitened signals is

$$P_{\mathbf{y}_k \mathbf{y}_l} \triangleq R_{\mathbf{y}_k}^{-\frac{1}{2}} R_{\mathbf{y}_k \mathbf{y}_l} R_{\mathbf{y}_l}^{-\frac{1}{2}} \quad (2.39)$$

and the TSC between nodes  $k$  and  $l$  can be shown to be equal to

$$\Theta_{kl} = \|P_{\mathbf{y}_k \mathbf{y}_l}\|_F^2, \quad (2.40)$$

where  $\|\cdot\|_F$  denotes the Frobenius norm (See Appendix A.4 for a proof). We first show how  $X$ , i.e. the MAXVAR solution of (2.10), relates to the eigenvalue decomposition of  $P_{\mathbf{y}\mathbf{y}} = R_D^{-\frac{1}{2}} R_{\mathbf{y}\mathbf{y}} R_D^{-\frac{1}{2}}$ , the network-wide covariance matrix of the (per-node) whitened signals. Note that  $P_{\mathbf{y}\mathbf{y}}$  is a block matrix such that

$$[P_{\mathbf{y}\mathbf{y}}]_{kl} = P_{\mathbf{y}_k \mathbf{y}_l} \quad (2.41)$$

where  $[\cdot]_{kl}$  denotes the  $M_k \times M_l$  block corresponding to node  $k$  and  $l$ , and where in particular the diagonal blocks  $[P_{\mathbf{y}\mathbf{y}}]_{kk} = I_{M_k}$ . We stated earlier that the spatial filters  $X$  were solutions of the GEVD (2.13). If we assume that  $\Lambda = X^T R_D X$  (see (2.13)) only contains non-zero GEVLs (which is always the case in the presence of noise), we can define

$$U \triangleq R_D^{\frac{1}{2}} X \Lambda^{-\frac{1}{2}}, \quad (2.42)$$

allowing us to write (2.13) as

$$U = P_{\mathbf{y}\mathbf{y}}U\Lambda \quad (2.43)$$

$$U^T P_{\mathbf{y}\mathbf{y}}U = \Lambda^{-1} \quad (2.44)$$

which can be reorganized as

$$P_{\mathbf{y}\mathbf{y}}U = U\Lambda^{-1} \quad (2.45a)$$

$$U^T U = I_Q. \quad (2.45b)$$

The columns of  $U$  are therefore orthogonal eigenvectors of  $P_{\mathbf{y}\mathbf{y}}$  and  $\Lambda^{-1}$  contains the corresponding eigenvalues. As a result, the  $Q$ -dimensional filters computed by MAXVAR are homeomorphic to the eigenvectors corresponding to the  $Q$  largest eigenvalues of  $P_{\mathbf{y}\mathbf{y}}$  (i.e. the smallest diagonal elements of  $\Lambda$ ) via (2.42).

## 2.5.2 Naive TSC approximation using D-MAXVAR

Considering the definition (2.40) of the TSC, it seems reasonable to approximate the TSC by replacing  $P_{\mathbf{y}_k\mathbf{y}_l}$  by  $[\hat{P}_{\mathbf{y}\mathbf{y}}]_{kl}$  in (2.40). Indeed, if we denote the  $Q$ -rank approximation of  $P_{\mathbf{y}\mathbf{y}}$  as  $\hat{P}_{\mathbf{y}\mathbf{y}}$ , we have

$$\hat{P}_{\mathbf{y}\mathbf{y}} = \underset{A}{\operatorname{argmin}} \sum_{k,l \in \mathcal{K}} \|P_{\mathbf{y}_k\mathbf{y}_l} - [A]_{kl}\|_F^2 \quad (2.46)$$

$$\text{s.t.} \quad \operatorname{Rank}(A) = Q. \quad (2.47)$$

$\hat{P}_{\mathbf{y}\mathbf{y}}$  can therefore be interpreted as a joint low-rank approximation of the matrices  $P_{\mathbf{y}_k\mathbf{y}_l}$ . This approach, however, has two issues. Consider the split of (2.46) between diagonal and off-diagonal blocks:

$$\min_A \sum_{k,l \in \mathcal{K}, k \neq l} \|P_{\mathbf{y}_k\mathbf{y}_l} - [A]_{kl}\|_F^2 + \sum_{k \in \mathcal{K}} \|I_{M_k} - [A]_{kk}\|_F^2 \quad (2.48)$$

First, if the off-diagonal blocks of  $P_{\mathbf{y}\mathbf{y}}$  have a very small norm, the joint low-rank approximation will be focused on reconstructing the diagonal blocks, which do not contain any information about the inter-node correlation structure. Second,  $P_{\mathbf{y}\mathbf{y}}$  might have full rank, even though the inter-node correlation structure can be described by a small number of components (e.g. in the case of a completely uncorrelated network,  $P_{\mathbf{y}\mathbf{y}} = I$ ). This makes the choice of an appropriate value for  $Q$  quite challenging as it is seemingly unrelated to the rank of  $P_{\mathbf{y}\mathbf{y}}$ . Ideally, our approximation method should be such that increasing the value of  $Q$ , and

therefore allocating more bandwidth, always results in a better approximation of the TSC. In addition, when the inter-node correlation structure can be described by only a few components, the TSC should be perfectly recovered by selecting  $Q \ll M$ .

### 2.5.3 TSC approximation using a modified D-MAXVAR

In Subsection 2.5.1, we have shown that MAXVAR was equivalent to computing the low-rank approximation of  $P_{\mathbf{y}\mathbf{y}}$  and in the previous subsection, we have described how using this low-rank approximation to approximate the TSC would result in an approximation lacking multiple desirable properties. Via a slight modification of the original problem, we can obtain a low-rank approximation of  $P_{\mathbf{y}\mathbf{y}} - I$ , from which we can derive a TSC approximation that satisfies the desired properties for  $Q$ . For this purpose, we propose to approximate the TSC by using  $\hat{P}_{\mathbf{y};\mathbf{y}_l} \triangleq [\hat{P}_{\mathbf{y}\mathbf{y}}^D]_{kl}$  in (2.40), where  $\hat{P}_{\mathbf{y}\mathbf{y}}^D$  denotes the low-rank approximation of  $P_{\mathbf{y}\mathbf{y}} - I$  instead of  $P_{\mathbf{y}\mathbf{y}}$ . This would resolve the problems mentioned in the previous subsection, as it removes the influence of the (irrelevant) diagonal entries in (2.48). In the following, we describe how the D-MAXVAR algorithm can be modified to efficiently compute this new TSC approximation.

We first note that the eigenvectors of  $P_{\mathbf{y}\mathbf{y}} - I$  and  $P_{\mathbf{y}\mathbf{y}}$  are the same, and that the corresponding eigenvalues can be obtained by subtracting 1 from the eigenvalues of  $P_{\mathbf{y}\mathbf{y}}$ . Indeed, the full eigenvalue decomposition of  $P_{\mathbf{y}\mathbf{y}}$  can be expressed as

$$P_{\mathbf{y}\mathbf{y}} = U_M \Lambda_M^{-1} U_M^T, \quad (2.49)$$

where  $U_M$  is an orthogonal matrix that contains the full set of  $M$  eigenvectors in its columns, and  $\Lambda_M^{-1}$  is the diagonal matrix of corresponding eigenvalues. The previously defined matrices  $U$  and  $\Lambda^{-1}$  are therefore  $Q$ -columns submatrices of  $U_M$  and  $\Lambda_M^{-1}$ , respectively. Subtracting  $I$  from both sides yields

$$P_{\mathbf{y}\mathbf{y}} - I = U_M (\Lambda_M^{-1} - I) U_M^T. \quad (2.50)$$

One can therefore obtain a  $Q$ -rank approximation of  $P_{\mathbf{y}\mathbf{y}} - I$  by computing the  $Q$  eigenvectors of  $P_{\mathbf{y}\mathbf{y}}$  corresponding to the  $Q$  largest diagonal elements of  $\Lambda_M^{-1} - I$  in absolute magnitude<sup>4</sup>. We can link  $U_M$  to the full set of GEVCs  $X_M$  of  $(R_D, R_{\mathbf{y}\mathbf{y}})$  by noting that (2.42) is valid for any subset of eigenvectors. Therefore

$$X_M = R_D^{-\frac{1}{2}} U_M \Lambda_M^{\frac{1}{2}}, \quad (2.51)$$

---

<sup>4</sup>As  $P_{\mathbf{y}\mathbf{y}} - I$  is no longer positive semidefinite, the components corresponding to negative eigenvalues also contribute to the low-rank approximation error proportionally to the squared magnitude of their associated eigenvalue.

and (2.50) in combination with the orthogonality of  $U_M$  yields

$$(R_{\mathbf{y}\mathbf{y}} - R_D)X_M = R_D X_M \Gamma_M \quad (2.52a)$$

$$X_M^T R_{\mathbf{y}\mathbf{y}} X_M = I \quad (2.52b)$$

with  $\Gamma_M \triangleq \Lambda_M^{-1} - I$ .  $X_M$  thus contains GEVCs of the pencil  $(R_{\mathbf{y}\mathbf{y}} - R_D, R_D)$  in its columns, and  $\Gamma_M$  contains the corresponding GEVLs. The problem of finding a  $Q$ -rank approximation of  $P_{\mathbf{y}\mathbf{y}} - I$  is therefore equivalent to finding the  $Q$  GEVCs of  $(R_{\mathbf{y}\mathbf{y}} - R_D, R_D)$  associated with its  $Q$ -largest (in absolute magnitude) GEVLs. These can be found based on a slightly modified version of the D-MAXVAR algorithm, described hereafter.

**Modified D-MAXVAR Algorithm:** Let  $X_Q$  denote the  $Q$ -columns submatrix of  $X_M$  associated with the  $Q$ -rank approximation of  $P_{\mathbf{y}\mathbf{y}} - I$ . In order to compute  $X_Q$  in a distributed fashion, Algorithm 2.3 must be modified such that, at each iteration  $i$ ,  $\tilde{X}^i$  contains the  $Q$  GEVCs of  $(R_{\tilde{\mathbf{y}}_q}^i - R_{D_q}^i, R_{D_q}^i)$  associated with the largest GEVLs (in absolute magnitude) instead of the  $Q$  smallest GEVLs of  $(R_{D_q}^i, R_{\tilde{\mathbf{y}}_q}^i)$ . This change has no impact on Theorems 2.1 and 2.2, which remain valid as  $\tilde{X}_Q$  is just another stationary point of the original problem (2.14). A similar result to Theorem 2.3 can be obtained by showing that selecting the components as described above is equivalent to solving a related optimization problem for which Theorem 2.3 applies, that is find the  $Q$  largest GEVCs of the pencil  $(R_{\mathbf{y}\mathbf{y}} - R_D, R_D)$ .

**Remark 2.3.** We note that the original MAXVAR components in  $X$  generally differ from the components in  $X_Q$ . Additional bandwidth would therefore need to be allocated if *both* the TSC approximation and the solutions to the classical MAXVAR problem would be required. If the goal is only to find an approximate TSC matrix (as in the example of Section 2.6), D-MAXVAR can be replaced with its modified version, in which case no additional bandwidth is required.

An expression for the approximation of the TSC using  $\hat{P}_{\mathbf{y}_k \mathbf{y}_l} \triangleq [\hat{P}_{\mathbf{y}\mathbf{y}}^D]_{kl}$  in (2.40) can be obtained as follows. Let  $\Gamma_Q$  be the diagonal submatrix of  $\Gamma_M$  corresponding to the GEVCs in  $X_Q$  obtained by the modified D-MAXVAR algorithm. We define the approximate TSC as

$$\hat{\Theta}_{kl} \triangleq \left\| [\hat{P}_{\mathbf{y}\mathbf{y}}^D]_{kl} \right\|_F^2 = \|U_{Q,k} \Gamma_Q U_{Q,l}^T\|_F^2 \quad (2.53)$$

where  $U_{Q,k} = R_{\mathbf{y}_k}^{\frac{1}{2}} X_{Q,k} \Lambda^{-\frac{1}{2}}$  (from (2.51)). From the definition of the Frobenius norm, we have

$$\|U_{Q,k} \Gamma_Q U_{Q,l}^T\|_F^2 = \text{Tr} (U_{Q,k} \Gamma_Q U_{Q,l}^T U_{Q,l} \Gamma_Q U_{Q,k}^T) \quad (2.54)$$

and from the cyclic property of the trace

$$\|U_{Q,k} \Gamma_Q U_{Q,l}^T\|_F^2 = \text{Tr} (U_{Q,k}^T U_{Q,k} \Gamma_Q U_{Q,l}^T U_{Q,l} \Gamma_Q). \quad (2.55)$$

From (2.51) and the fact that  $\Lambda_M^{\frac{1}{2}} = (\Gamma_M + I)^{-\frac{1}{2}}$  (from the definition of  $\Gamma_M$ ), we find that

$$U_{Q,k}^T U_{Q,k} = (\Gamma_Q + I_Q)^{\frac{1}{2}} X_{Q,k}^T R_{\mathbf{y}_k} X_{Q,k} (\Gamma_Q + I_Q)^{\frac{1}{2}} \quad (2.56)$$

$$= (\Gamma_Q + I_Q)^{\frac{1}{2}} R_{\mathbf{z}_k}^Q (\Gamma_Q + I_Q)^{\frac{1}{2}} \quad (2.57)$$

where  $R_{\mathbf{z}_k}^Q \triangleq X_{Q,k}^T R_{\mathbf{y}_k} X_{Q,k}$  is the covariance matrix of the compressed signal  $\mathbf{z}_k$  associated with  $X_{Q,k}$ , such that we finally obtain

$$\hat{\Theta}_{kl} = \text{Tr} (R_{\mathbf{z}_k}^Q (\Gamma_Q^2 + \Gamma_Q) R_{\mathbf{z}_l}^Q (\Gamma_Q^2 + \Gamma_Q)). \quad (2.58)$$

Note that  $R_{\mathbf{z}_k}^Q$  can be locally computed at node  $k$  and shared with negligible communication cost compared to the compressed signal observations themselves. Indeed, instead of sharing  $M_k T$  samples ( $T$  being the window length considered) samples, the nodes only need to share  $Q \times Q$  matrices, which is cheap even for nodes several hops away. Finally, we have from (2.52) that  $\Gamma_Q$  can be computed as

$$\Gamma_Q = \Lambda_Q^{-1} - I_Q = X_Q^T R_D X_Q = \left( \sum_k R_{\mathbf{z}_k}^Q \right)^{-1} - I_Q, \quad (2.59)$$

which can be efficiently estimated at each node using (2.34) without any additional communication.

## 2.5.4 Selecting $Q$

The following theorem provides a useful bound on the TSC total approximation error:

**Theorem 2.4.** *Let  $\gamma$  denote the  $Q$ -largest element of  $\Gamma$  in absolute magnitude. Then the total TSC approximation error*

$$\sum_{k,l \in \mathcal{K}, k \neq l} \left( \sqrt{\hat{\Theta}_{kl}^Q} - \sqrt{\Theta_{kl}} \right)^2 \leq (M - Q) \gamma^2. \quad (2.60)$$

*Proof.* See Appendix A.5. □

Any node can compute the above bound to assess the quality of the TSC approximation, as  $\gamma$  also corresponds to the smallest diagonal element of  $\Gamma_Q$ , locally computable via (2.59).

Note that if  $\text{Rank}(P_{\mathbf{y}\mathbf{y}} - I)$  is known a priori, selecting  $Q = \text{Rank}(P_{\mathbf{y}\mathbf{y}} - I)$  would result in a perfect recovery of  $P_{\mathbf{y}\mathbf{y}} - I$ , and hence of the TSC (by definition (2.40)). However, the rank is in practice not known in advance, and the node must therefore estimate it by increasing  $Q$  until  $\gamma = 0$ , as then  $\text{Rank}(P_{\mathbf{y}\mathbf{y}} - I) = Q - 1$ .

**Remark 2.4.** One can also take a statistical approach and test whether the absolute magnitudes of the eigenvalues are significantly larger than 0, i.e., whether they truly capture correlated components or whether their energy is explained by the estimation noise. The latter can be tested, e.g., via a maximum likelihood ratio test as in [33], [84]–[86], [98].

## 2.6 Simulation results

In this section, we validate the TSC approximation described in Section 2.5 and demonstrate the convergence properties of the D-MAXVAR algorithm in tree-topology networks.

### 2.6.1 Simulation settings

In the context of WSNs, it is appropriate to model the observed signals as noisy observations of a mixture of uniformly spatially distributed latent sources, i.e.

$$\mathbf{y}_k = A_k \mathbf{s} + \alpha \mathbf{n}_k \quad (2.61)$$

where  $\mathbf{n}_k$  is  $M_k$ -dimensional spatially white sensing noise at node  $k$  and uncorrelated with the noise at other nodes,  $\mathbf{s}$  is a  $d$ -dimensional spatially white latent signal and  $A_k$  is the mixing matrix associated with node  $k$ . Finally,  $\alpha$  is a network-wide parameter allowing us to modulate the signal-to-noise ratio (SNR). Let  $A_k = [a_k^1, \dots, a_k^d]$ , where  $a_k^j$  denote the steering vector at node  $k$

associated with source  $s^j$ . We model it as

$$a_k^j = \frac{\mathbf{g}_k^j}{\max\{0.1, \|\mathbf{m}_k - \mathbf{l}_j\|^2\}} \quad (2.62)$$

where  $\mathbf{g}_k^j$  is an  $M_k$ -dimensional vector of random variables drawn uniformly from  $[0.95, 1.05]$ , modeling the slight discrepancies in the channel gains, and  $\mathbf{m}_k$  and  $\mathbf{l}_j$  are the random coordinates of node  $k$  and source  $j$ , uniformly drawn from a 10 by 10 square. With this model the sources can be seen as point sources radiating energy uniformly in all directions. The associated covariance matrix can be computed as

$$R = AA^H + \alpha^2 I \quad (2.63)$$

where  $A$  is the matrix obtained by stacking the  $A_k$ 's. The total power of the latent sources picked up by the nodes is  $P_s = \|A\|_F^2$  and the total noise power is  $\alpha^2 M$ . We set  $\alpha^2 = P_s(M \text{ SNR})^{-1}$  to obtain the desired SNR.

For each simulated scenario, we performed 1000 Monte-Carlo runs where the covariance matrices were directly computed via (2.63). Additionally, we set  $M_k = 8$  and  $d = 3$ .

## 2.6.2 TSC approximation

Using the above model and settings with  $K = 10$ , we obtained the results depicted by Figure 2.4 with the average absolute error  $E$  defined as

$$E \triangleq \frac{1}{K(K-1)} \sum_{k,l \neq k} |\hat{\Theta}_{kl} - \Theta_{kl}| \quad (2.64)$$

As expected, the quality of the approximation increases with increasing  $Q$  and always eventually reaches 0 for a sufficiently high value of  $Q$ . As the SNR grows, so does the value of  $Q$  required to obtain a perfect approximation of the TSC. Indeed, a larger SNR causes the eigenvalues of  $R_{\mathbf{y}\mathbf{y}}$ , and hence  $P_{\mathbf{y}\mathbf{y}}$ , to grow closer to 0. As a consequence, the corresponding eigenvalues of  $P_{\mathbf{y}\mathbf{y}} - I$  become closer to  $-1$ , resulting in more power in the off-diagonal blocks which needs to be accounted for. In practice, using our method in such a setting where the signal subspace is perfectly observed by every node does not make much sense as a simple distributed PCA (based for example on [55]) would yield an excellent approximation of  $R_{\mathbf{y}\mathbf{y}}$  with only  $Q = d$  components. On the other hand, such a PCA-based method would perform very poorly in low-SNR regimes, as it would first compute the components of highest power, which would in that case

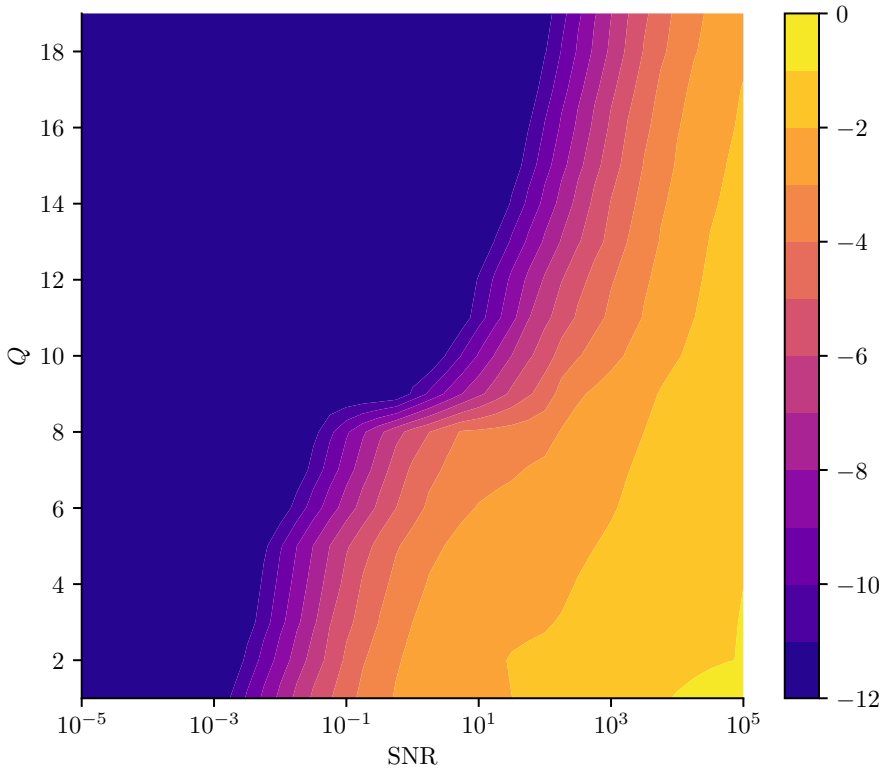


Figure 2.4: Monte-Carlo simulations of the TSC approximation error for varying  $Q$  and SNR values. The color scale corresponds to the log of the mean value obtained for  $E$  after 1000 Monte-Carlo runs. The log-error values are clipped to -12.

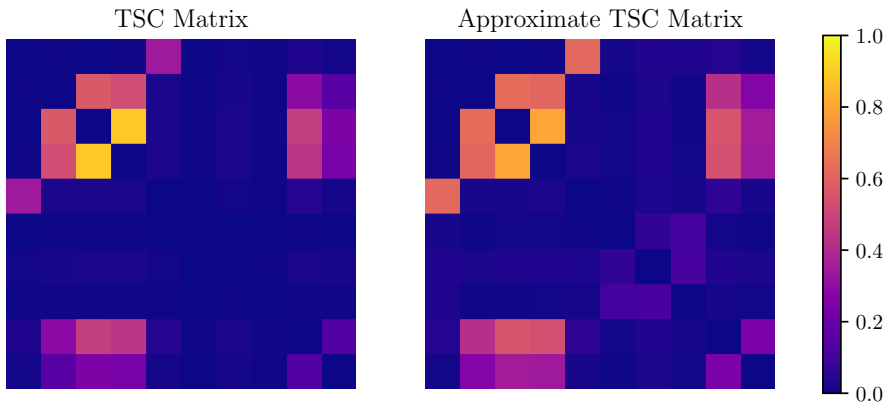


Figure 2.5: Example TSC matrices obtained with  $K = 10$ ,  $M_k = 8$ ,  $Q = 3$ , 3 latent sources and  $\text{SNR} = 1$ . Each pixel represents the (approximate) TSC of a pair of node. Diagonal values are not shown.

correspond to noise and would therefore not yield any information about the correlation structure. Our method is therefore best suited for scenarios where the node-specific subspaces and common subspaces have similar power levels and are hard to separate. Figure 2.5 showcases such a scenario. We see that the correlation structure is well captured by the approximate TSC using only 3 components when node-specific noise and latent signals have similar power levels.

### 2.6.3 Convergence

Figure 2.6 shows the convergence of the D-MAXVAR algorithm to the global solution. The metric used is

$$e^i = \frac{f(X^i)}{f^*} - 1 \quad (2.65)$$

where  $f(X) = X^H R_D X$  is the objective function of (2.14) and  $f^*$  is its value at a global minimizer. In the top plot, we see that convergence is indeed faster with a larger number of components  $Q$ . In the bottom plot, we see the impact of the tree depth: the deeper the tree, the higher the compression (due to the recursive summing resulting from Algorithm 2.2). We expect convergence speed to increase in more densely connected networks, that is networks with more links, as this increases the number of neighbors of each node and hence

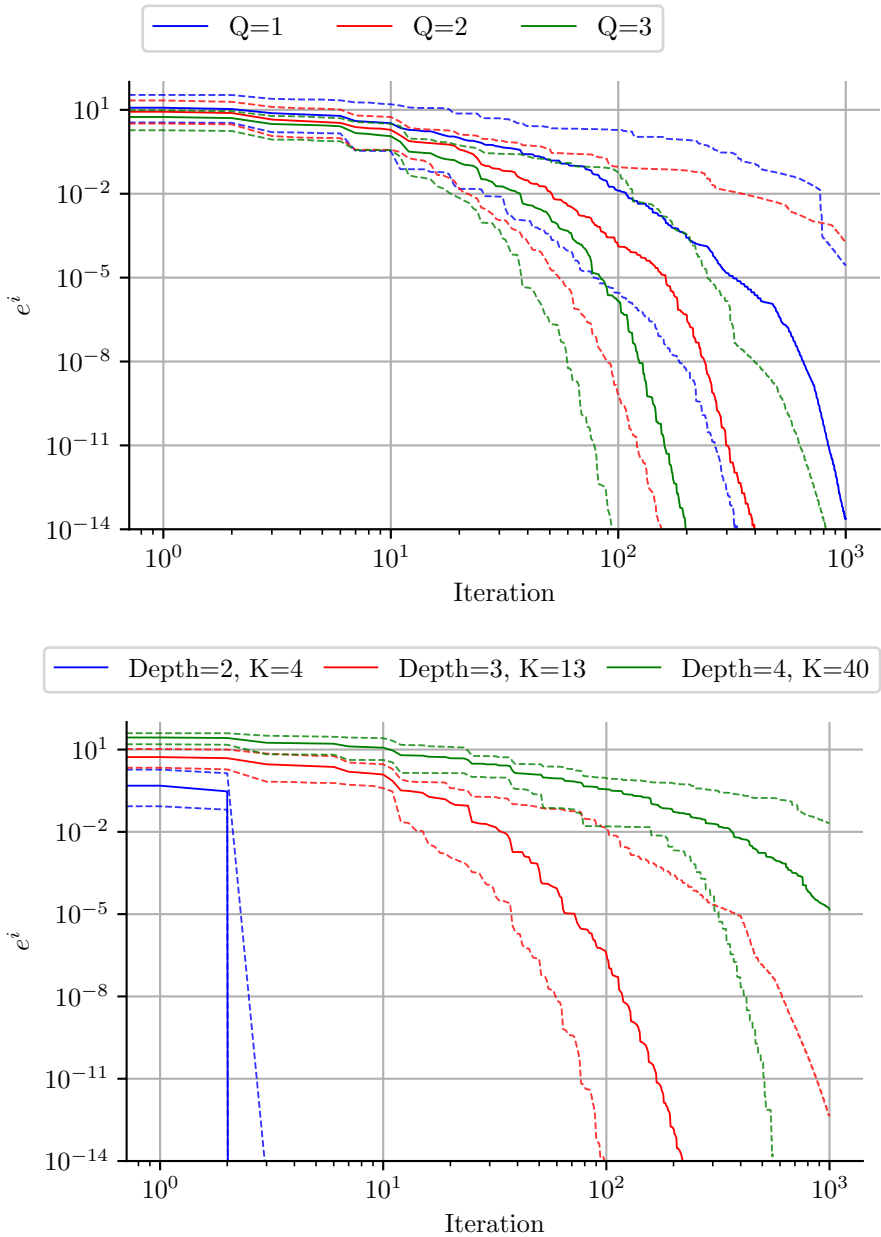


Figure 2.6: Monte-Carlo simulations of the convergence of the D-MAXVAR algorithm in tree-topology networks. Solid curves depict the mean values. Shaded areas depict the 5-95% percentile regions. The top plot was generated with  $K = 13$ . The bottom plots were generated with  $K = 4, 13, 40$ .

the number of  $\tilde{X}_k$  matrices possibly used at each iteration (that is assuming that an instance of Algorithm 2.4 is run at each iteration). In tree-topology networks, the total number of links is  $K - 1$ , while it is  $K^2$  in fully connected networks, the convergence speed is therefore expected to drop much more rapidly with increasing network size  $K$  in the case of tree-topology networks as the compensation in terms of added neighbors per node is much less significant than for fully connected networks.

Finally, we see that, in practice, convergence to a global minima is always achieved due to the instability of other fixed points (see Theorem 2.3).

## 2.7 Discussion

In this chapter, we have presented an algorithm that computes the solution of the so-called MAXVAR problem in a distributed setting. Our algorithm displays significant savings in computational and communication requirements compared to a centralized procedure where signal observations are collected at a single location. In particular, in arbitrary-topology networks, the communication cost is independent of the network size and only depends on the degree of each node and the chosen compression factor  $Q$ . We have proven the convergence properties of the algorithm, and shown via simulations that the condition for global convergence holds in practice. In the next chapter, we introduce the DASF framework, of which D-MAXVAR is in fact a particular case, and we show that the convergence properties of D-MAXVAR still hold when the same algorithmic ideas are applied to more general problems.

We have also shown how the networks' correlation structure can be efficiently estimated from the algorithm's outputs and for negligible additional cost, via an approximation of the total squared correlation. Even though the proposed approximation can be arbitrarily precise, it is not adapted to very high-SNR regimes, as the number of required components can quickly grow to become larger than the number of channels at a given node. Nevertheless, we have demonstrated that the approximation is quite accurate for lower SNR regimes, even with low  $Q$ , and allows significant computational savings compared to the exact computation of all pair-wise TSCs. Finally, we note that, in most cases, there will be no interest in perfectly estimating the TSC. Indeed, for most applications, the TSC will be thresholded in order to determine whether the link between two nodes should be kept alive, or simply used as a distance input to some clustering algorithm.



## Chapter 3

---

# A general framework for smooth distributed spatial filtering problems

---

This chapter is mostly based on joint work with C. Musluoglu and A. Bertrand in

- \* C. A. Musluoglu, **C. Hovine**, and A. Bertrand, “A unified algorithmic framework for distributed adaptive signal and feature fusion problems — part II: Convergence properties”, *IEEE Transactions on Signal Processing*, vol. 71, pp. 1879–1894, 2023

which was rewritten and reorganized to fit the notation of this thesis. Subsections 3.1.2 and 3.2 were freely adapted from the reviews of DASF described in

- \* **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems in wireless sensor networks”, *IEEE Transactions on Signal Processing*, vol. 72, pp. 4682–4697, 2024
- \* **C. Hovine** and A. Bertrand, “Distributed adaptive spatial filtering with inexact local solvers”, *arXiv preprint arXiv:2405.03277*, 2024

Subsection 3.1.1 on non-linear Gauss-Seidel and accompanying appendices, and Subsection 3.3.5 on the convergence issues of DASF, along with all simulated examples, are unique to this thesis.

---

## Chapter abstract

---

This chapter describes a general framework under which D-MAXVAR and the DSF algorithms fall. We first explore how a naive alternating optimization approach could be used to compute an unconstrained spatial filter in a distributed fashion. We then describe the family of problems covered by the framework, and build a generic procedure that will serve as the basis on which more specialized algorithms can be built. We show the convergence of this generic procedure for smooth spatial filters, and under somewhat restrictive assumptions, which will be relaxed in future chapters.

### 3.1 Preliminaries

DASF is a flexible framework that applies to most spatial filtering problems, but for the sake of simplicity, we will first consider the most basic problem covered by DASF: a smooth unconstrained problem, which we can express as

$$X^* \in \operatorname{argmin}_{X \in \mathbb{R}^{M \times Q}} \varphi(X^T \mathbf{y}, X^T B) \quad (3.1)$$

where  $X \mapsto \varphi(X^T \mathbf{y}, X^T B)$  from  $\mathbb{R}^{Q \times M}$  to  $\mathbb{R}$  is a smooth function. The filter and optimization variable  $X$  can be partitioned according to the corresponding partition of  $\mathbf{y}$  in  $\mathbf{y}_k$ 's described in Subsection 1.2.1, i.e.,

$$X \triangleq \begin{bmatrix} X_1 \\ \vdots \\ X_K \end{bmatrix}. \quad (3.2)$$

With the constraint that each node  $k$  only has access to the realizations of its own block of  $\mathbf{y}$ ,  $\mathbf{y}_k$ , and considering that  $\varphi$  is not block separable, i.e., we cannot express it as a sum

$$\varphi(X^T \mathbf{y}) = \sum_k \varphi_k(X_k^T \mathbf{y}_k) \quad (3.3)$$

for some functions  $\varphi_k$ , how can we evaluate the  $Q$ -dimensional output  $\mathbf{z} \triangleq X^* T \mathbf{y}$  in a bandwidth efficient manner?

#### 3.1.1 From Gauss-Seidel to DASF

A first approach could consist in successively optimizing each of the blocks  $X_k$  of  $X$ , in the hope that it eventually converges to the optimal filter. This method is known as non-linear Gauss-Seidel, or alternating optimization [99]. Applied to the WSN setting, this would result in an iterative procedure where, at each iteration  $i$ , a different node  $q^i$ , which we call the *updating node* (we use the node index  $q$  without any iteration index to refer to the updating node when the iteration is clear from the context), would update its own block of the filter  $X$ . The procedure could be summarized by the following two steps:

**(i) Data aggregation** Each node  $k$  collects  $N$  new samples of  $\mathbf{y}_k$  and sends a block of  $N$   $Q$ -dimensional *compressed* samples of

$$\hat{\mathbf{y}}_k^i \triangleq X_k^{iT} \mathbf{y}_k \quad (3.4)$$

and

$$\widehat{B}_k^{iT} \triangleq X_k^{iT} B_k \quad (3.5)$$

to the updating node  $q$ . Note that  $X_k^i$  acts both as a compression matrix and as the current estimate of the filter of node  $k$ .

**(ii) Local solution** Using the compressed data it received in the previous step, the updating node  $q$  computes the solution of the *local problem*

$$X_q^* = \operatorname{argmin}_{X_q \in \mathbb{R}^{M_q \times Q}} \varphi \left( X_q^T \mathbf{y}_q + \sum_{k \neq q} \widehat{\mathbf{y}}_k^i, X_q^T B_q + \sum_{k \neq q} \widehat{B}_k^i \right) \quad (3.6)$$

which we assume unique, and updates its block of the filter as  $X_q^{i+1} \leftarrow X_q^*$ .

At each iteration, we select a new updating node, typically in a sequential manner. The updating node can estimate the target filter output as

$$\mathbf{z} \approx X_q^{i+1T} \mathbf{y}_q + \sum_{k \neq q} \widehat{\mathbf{y}}_k^i \quad (3.7)$$

without requiring any additional data transmission. The full procedure is described by Algorithm 3.1.

We expect that such a procedure would converge towards a solution, or a least to a stationary point of (3.1), as stated in the following theorem.

---

**Algorithm 3.1:** Distributed non-linear Gauss-Seidel algorithm.

---

**begin**

$i \leftarrow 0, q \leftarrow 1$ , Randomly initialize  $X^0$

**loop**

**for**  $k \in \mathcal{K} \setminus \{q\}$  **do**

**At node**  $k$

Collect a new batch of  $N$  samples of  $\mathbf{y}_k(t)$  and send the compressed samples  $\widehat{\mathbf{y}}_k^i(t) = X_k^{iT} \mathbf{y}_k(t)$  and  $\widehat{B}_k^i = X_k^{iT} \widehat{B}_k$  to node  $q$ .

**At node**  $q$

Obtain  $X_q^*$  by solving (3.6).

$X_q^{i+1} \leftarrow X_q^*$

$i \leftarrow i + 1, q \leftarrow (q + 1) \bmod K$

---

**Theorem 3.1** (Convergence of Gauss-Seidel). *Let  $X \mapsto \varphi(X^T \mathbf{y}, X^T B)$  have compact sub-level sets. Then any sequence generated by Algorithm 3.1 converges to the set of stationary points of problem (3.1). Furthermore, if the problem has finitely-many stationary points, then the sequence converges to a single point.*

*Proof.* We provide a full proof in Appendix B for the interested reader. Note that it shares many similarities with the convergence proofs described in later chapters, and can serve as a smooth introduction to our general proof methodology.  $\square$

Without further assumptions such as convexity, we cannot state a stronger optimality result.

The Gauss-Seidel procedure is conceptually simple and has several appealing characteristics. As the updating node only requires access to the aggregated sums of data in (3.4) and (3.5), the sums can be computed recursively, where each node sends data only once, similarly to the procedure described by Algorithm 2.2 for D-MAXVAR. The per-node bandwidth requirements therefore only scales with the dimension  $Q$  of the output filter, and is independent of the network's size. Similarly, the updating node need only solve a problem with dimension  $QM_q$ , independently of the number of nodes, making the procedure suited for problems in potentially very large networks. This must be contrasted with a centralized procedure, where each node would need to send (with potential data-relaying in a multi-hop network) its full  $M_k$  dimensional output signal to a fusion center, which would in turn need to be able to compute the solution of the full problem. This scaling dependent on  $M$  will always result in a hard limit on the total number of nodes  $K$  whose data can be jointly processed.

Despite its benefits, the Gauss-Seidel procedure is lacking in some important aspects. Firstly, the updating node is not making full-use of the received data. Indeed, it only updates its block  $X_q$ , while it could technically also partially manipulate the other blocks  $X_k$  via the introduction of an appropriate parametrization, as was done with the  $\tilde{X}_k$  for D-MAXVAR in (2.23). Secondly, the local problems (3.6) are not guaranteed to have the same structure as the original problem (3.1), and a dedicated solver for the local problems might thus be required. Finally, the method described here is not suited for general constraints, and typically requires the constraints to be per-node separable [99]. It is these short-comings that are addressed in the rest of this chapter.

### 3.1.2 Data-driven spatial filtering

This subsection extends the basic problem in (3.1) and describes the general scope of spatial filtering problems we consider. In its most generic form, a data-driven spatial filtering problem can be expressed as

$$X^* \in \underset{X \in \mathbb{R}^{M \times Q}}{\operatorname{argmin}} \varphi(X^T \mathbf{y}(t), X^T B) \quad (3.8a)$$

$$\text{s.t.} \quad \forall j \in \mathcal{J}_I, \eta_j(X^T \mathbf{y}(t), X^T D_j) \leq 0, \quad (3.8b)$$

$$\forall j \in \mathcal{J}_E, \eta_j(X^T \mathbf{y}(t), X^T D_j) = 0. \quad (3.8c)$$

where the matrices  $B$  and  $D_j$ <sup>1</sup> are deterministic matrices,  $\varphi$  is a smooth real-valued function encoding some design objective for the filter output,  $\mathcal{J}_I$  and  $\mathcal{J}_E$  are the sets of inequality and equality constraints indices, respectively, and  $\eta_j$  are smooth functions enforcing some hard constraints on the filter outputs and/or filter coefficients. We denote by  $\mathbb{P}(\mathbf{y}, B, \mathcal{D})$ , where  $\mathcal{D}$  denotes the collection of  $D_j$ , the family of parametric problems defined by (3.8).

As  $\mathbf{y}(t)$  is a stochastic signal, we assume that the aforementioned functions implicitly contain an operator extracting some statistics from the signal, such as, e.g., an expectation or covariance operator, hence keeping the problem deterministic<sup>2</sup>. As previously described in Subsection 1.2.1, in order for the nodes to be able to evaluate, or at least approximate those quantities, we assume that  $\mathbf{y}(t)$  is ergodic, i.e. its statistics can be evaluated using sample averages. We furthermore assume that  $\mathbf{y}(t)$  is short-time stationary and that its statistics change sufficiently slowly such that they can be tracked by the updates of the DASF algorithm.

The peculiar structure of problem (3.8) ensures that  $X$  always appears either in an inner product with  $\mathbf{y}(t)$  or in an inner product with some pre-determined matrices ( $B$  or  $D_j$ ). Most traditional spatial filtering or linear estimation problems satisfy this structure (examples are given hereafter, and more extensive illustrations can be found in [58]). The DASF algorithm will exploit this inner-product structure to achieve a bandwidth reduction. Note that the framework also allows for complex-valued signals, in which case all transpose operators should be replaced with conjugate (a.k.a. Hermitian) transpose operators. Finally, it is noted that we only include a single  $\mathbf{y}$  and  $B$ -matrix in the loss

<sup>1</sup>The non sequitur notation is chosen to stay consistent with the notation introduced in [58], where  $C_k$  has another meaning, used later in this text.

<sup>2</sup>In order to be perfectly rigorous, we could have defined the domain of the above functions as a subset of some Hilbert space of random signals. This would however have introduced unnecessary complexity, as for all intents and purposes, the random signals could be replaced by sample matrices and leave our developments mostly unchanged.

function  $\varphi$ , and a single  $D$ -matrix in each constraint. This is for the sake of notational convenience, as the DASF framework also allows multiple instances of each (details in [58]).

By imposing the proper structure on  $\mathbf{y}$ ,  $\varphi$ , and the constraints, a wide range of problems can be cast to the form of (3.1). For example, the Wiener filtering problem [100] fits perfectly within the framework:

$$\min_X \mathbb{E} \left\{ \|X^T \mathbf{y} - \mathbf{d}\|_F^2 \right\} \quad (3.9)$$

where  $\mathbf{d}$  is a known target signal taking values in  $\mathbb{R}^Q$ . LCMV filters [36] also match the structure of (3.8):

$$\begin{aligned} X^* &\in \operatorname{argmin}_X X^T \mathbb{E} \{ \mathbf{y} \mathbf{y}^T \} X \\ \text{s.t.} \quad &X^T D = H, \end{aligned} \quad (3.10)$$

with  $D$  and  $H$  some matrices of appropriate dimensions. Finally,

$$\begin{aligned} X^* &\in \operatorname{argmax}_X X^T \mathbb{E} \{ \mathbf{y} \mathbf{y}^T \} X \\ \text{s.t.} \quad &X^T X = I_Q \end{aligned} \quad (3.11)$$

is a typical instance of (3.8). It produces a filter that extracts the principal components of  $\mathbf{y}$ . In this particular case  $B = 0$ , and the sole constraint is

$$\eta(X^T \mathbf{y}, X^T D) = X^T D X = I_Q, \quad (3.12)$$

with  $D = I_Q$ . Although the  $D$  matrix looks superfluous here, it plays an important algorithmic role, and allows DASF to solve (3.8) by solving a sequence of lower-dimensional versions of (3.11), but where  $D$  (or  $B$  when it is non-zero) is not equal to the identity matrix anymore.

**Remark 3.1** (Spatio-temporal filters). The scope of (3.8) can be straightforwardly extended to spatio-temporal filters rather than merely spatial filters by including delayed versions of the channels of  $\mathbf{y}(t)$  in the problem formulation. That is, the signal processed by node  $k$  would in fact be

$$\mathbf{y}'_k(t) \triangleq \begin{bmatrix} \mathbf{y}_k(t) \\ \mathbf{y}_k(t - t_d) \\ \vdots \\ \mathbf{y}_k(t - Lt_d) \end{bmatrix} \quad (3.13)$$

where  $t_d$  and  $L$  are chosen delay parameters, and  $\mathbf{y}_k(t)$  is the original, non-delayed signal sensed by node  $k$ . The resulting filter  $X$  will therefore have a non-flat frequency response (with regards to the original  $\mathbf{y}(t)$ ). All the conclusions and developments made for spatial filters in the next sections hold for such spatio-temporal filters.

## 3.2 DASF overview

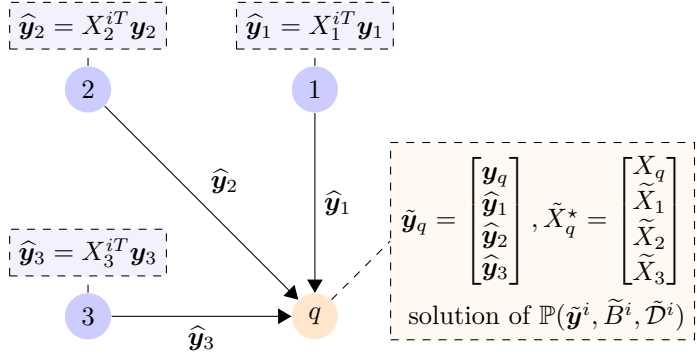
In this section, we briefly review the DASF algorithm. We refer to [58] for more details and illustrative examples.

The DASF algorithm collaboratively updates the estimate of the optimal filter  $X^*$  and tracks the filtered signal  $\mathbf{z}$  by having the nodes solve in turn a local “miniature” version of (3.8), which, contrarily to the Gauss-Seidel method, is guaranteed to preserve the original problem’s structure. This makes the DASF algorithm “plug-and-play” as it only requires a solver for the centralized problem (3.8), which is directly re-used as a local “sub-solver” in the compressed local problems in the different iterations of the distributed algorithm [58].

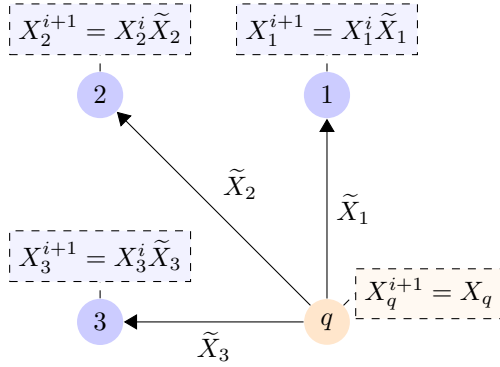
Contrarily to the Gauss-Seidel procedure, there are slight algorithmic differences between the fully-connected and arbitrary network topologies. We therefore first introduce the algorithm in fully-connected networks, before extending it to arbitrary network topologies in the next section.

### 3.2.1 Fully-connected networks

An iteration  $i$  of the DASF algorithm is fully characterized by the current updating node index  $q^i$  and the current estimate of the filter  $X^i$ , and consists of the following three steps (after a random initialization of  $X^0$ ):



(a) Data aggregation and local solution



(b) Parameters update

Figure 3.1: Overview of a single iteration of the DASF Algorithm in fully-connected networks, where node  $q$  is the updating node. The matrices  $B_k$  and  $D_{j,k}$  are omitted for readability, but their treatment is the same as  $\mathbf{y}_k$ .

**(i) Data aggregation** At the beginning of a new iteration  $i$ , a new updating node  $q$  is selected. Each node  $k$  collects a new batch of  $N$  samples of  $\mathbf{y}(t)$  for  $t = iN, \dots, (i+1)N - 1$ , and compresses these into  $Q$ -dimensional<sup>3</sup> samples according to

$$\widehat{\mathbf{y}}_k^i \triangleq X_k^{iT} \mathbf{y}_k. \quad (3.14)$$

The compressed  $N$ -samples batch is then transmitted to the updating node, along with the compressed matrices

$$\widehat{D}_{j,k} \triangleq X_k^{iT} D_{j,k} \quad (3.15)$$

$$\widehat{B}_k^i \triangleq X_k^{iT} B_k \quad (3.16)$$

(typically of negligible size compared to the transmission of (3.14) when the sample batch size  $N$  is large). Here,  $D_{j,k}$  and  $B_k$  are the blocks of  $D_j$  and  $B$ , respectively, associated with the block  $X_k$  of  $X$  in the products  $X^T D_j$  and  $X^T B$ .

**(ii) Local solution** The updating node constructs a *local* view  $\tilde{\mathbf{y}}_q$  of  $\mathbf{y}$  by concatenating the received signals with its own sensor signals such that

$$\tilde{\mathbf{y}}^i = \begin{bmatrix} \mathbf{y}_q \\ \widehat{\mathbf{y}}_1^i \\ \vdots \\ \widehat{\mathbf{y}}_{q-1}^i \\ \widehat{\mathbf{y}}_{q+1}^i \\ \vdots \\ \widehat{\mathbf{y}}_K^i \end{bmatrix}. \quad (3.17)$$

Similarly, it constructs

$$\tilde{B}^i = \begin{bmatrix} B_q \\ \widehat{B}_1^i \\ \vdots \\ \widehat{B}_{q-1}^i \\ \widehat{B}_{q+1}^i \\ \vdots \\ \widehat{B}_K^i \end{bmatrix} \quad \text{and} \quad \tilde{D}_j^i = \begin{bmatrix} D_q \\ \widehat{D}_{j,1}^i \\ \vdots \\ \widehat{D}_{j,q-1}^i \\ \widehat{D}_{j,q+1}^i \\ \vdots \\ \widehat{D}_{j,K}^i \end{bmatrix}. \quad (3.18)$$

<sup>3</sup>Here we assume that  $Q < M_k$ , otherwise there is no compression possible at node  $k$ . Nodes for which  $M_k \leq Q$  can add their channels  $y_k$  to the channels of a neighboring node to create a virtual “super-node” within the DASF algorithm. Note that in multi-hop network topologies, a bandwidth reduction can even be achieved if  $Q > M_k$  in all nodes (when compared to relayed data aggregation). See Subsection 3.2.2 for more details.

The updating node then obtains the local solution  $\tilde{X}^*$  by solving  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{\mathcal{D}}^i)$  for the received  $N$ -sample batch, where, similarly to  $\mathcal{D}$ ,  $\tilde{\mathcal{D}}^i$  denotes the collection of  $\tilde{D}_j^i$ . Note that the solver for the network-wide problem  $\mathbb{P}(\mathbf{y}, B, \mathcal{D})$  can be used as-is to solve the “compressed” problem  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{\mathcal{D}}^i)$  at updating node  $q$  [58]. If the local problem has more than one solution, then the solution minimizing the distance  $\|\tilde{X} - \tilde{X}^i\|_F$  is chosen, where

$$\tilde{X}^i \triangleq \begin{bmatrix} X_q^i \\ I_Q \\ \vdots \\ I_Q \end{bmatrix}, \tag{3.19}$$

(see rationale below).

**(iii) Parameters update** The updating node partitions the local solution  $\tilde{X}^*$  as<sup>4</sup>

$$\tilde{X}^* = \begin{bmatrix} X_q^* \\ \tilde{X}_1^* \\ \cdots \\ \tilde{X}_{q-1}^* \\ \tilde{X}_{q+1}^* \\ \cdots \\ \tilde{X}_K^* \end{bmatrix}. \tag{3.20}$$

It then updates its own block of the filter as

$$X_q^{i+1} \leftarrow X_q^*, \tag{3.21}$$

and transmits each  $\tilde{X}_k$  to its corresponding node, which in turn updates its local filter as

$$X_k^{i+1} \leftarrow X_k^i \tilde{X}_k^*. \tag{3.22}$$

Note that, from the update rules (3.21) and (3.22), the local  $\tilde{X}^i$  in (3.19) corresponds to the global iterate  $X^i$ . The distance minimization performed in step (ii) therefore favors solution closest from the previous iterate. The three steps are illustrated in Figure 3.1 and the full procedure is described by Algorithm 3.2.

---

<sup>4</sup>Note that  $X_q^* \in \mathbb{R}^{M_q \times Q}$  and  $\tilde{X}_k^* \in \mathbb{R}^{Q \times Q}$  for all other  $k \neq q$ . This follows from the dimensions of the partitioning in (3.17).

**Algorithm 3.2:** DASF algorithm in fully-connected networks.

---

```

begin
   $i \leftarrow 0, q \leftarrow 1$ , Randomly initialize  $X^0$ 
  loop
    for  $k \in \mathcal{K} \setminus \{q\}$  do
      At node  $k$ 
      | Collect a new batch of  $N$  samples of  $\mathbf{y}_k(t)$  and send the
      | compressed samples  $\tilde{\mathbf{y}}_k^i(t) = X_k^{iT} \mathbf{y}_k(t)$  along with
      |  $\hat{B}_k^i = X_k^{iT} \hat{B}_k$  and  $\hat{D}_k^i = X^{iT} D_k$  to node  $q$ .
    At node  $q$ 
    | Obtain  $\tilde{X}^*$  by solving and selecting any solution of  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i)$ .
    | In the case of multiple solutions, select the one minimizing
    |  $\|\tilde{X}^* - \tilde{X}^i\|_F$ , with  $\tilde{X}^i$  defined in (3.19).
    | Extract  $X_q^*$  and the  $\tilde{X}_k^*$ 's from  $\tilde{X}^*$  according to (3.20).
    |  $X_q^{i+1} \leftarrow X_q^*$ 
    for  $k \in \mathcal{K} \setminus \{q\}$  do
      | Send  $\tilde{X}_k^*$  to node  $k$ .
      At node  $k$ 
      |  $X_k^{i+1} \leftarrow X_k^i \tilde{X}_k^*$ 
     $i \leftarrow i + 1, q \leftarrow (q + 1) \bmod K$ 
  
```

---

As a different batch of  $N$  samples is used at each iteration, the DASF algorithm produces the filtered signal

$$\mathbf{z}^{i+1} \triangleq X^{i+1T} \mathbf{y} = \sum_k X_k^{i+1T} \mathbf{y}_k = \tilde{X}^{*T} \tilde{\mathbf{y}}^i \quad (3.23)$$

for each  $N$ -samples block, while at the same time improving the estimate of the optimal spatial filter  $X^*$ , such that each new block of the filtered signal is closer to the desired filtered signal (under the stationarity assumption 1.2). In other words, DASF acts as a time-recursive block-adaptive filter that continuously adapts itself over time to the (possibly changing) statistics of  $\mathbf{y}(t)$  [58]. Note that the last equality implies that each updating node can locally produce the new estimate  $\mathbf{z}^{i+1}$  without additional data exchange.

In order to get an intuitive understanding of the benefits of DASF over plain Gauss-Seidel, let us consider again the unconstrained problem (3.1). The cost

function associated with the local problem in the case of Gauss-Seidel was

$$\varphi \left( X_q^T \mathbf{y}_q + \sum_{k \neq q} \hat{\mathbf{y}}_k^i, X_q^T B_q + \sum_{k \neq q} \hat{B}_k^i \right), \quad (3.24)$$

and the optimization variable had  $QM_q$  degrees of freedom. In the case of DASF, it would be

$$\varphi \left( X_q^T \mathbf{y}_q + \sum_{k \neq q} \tilde{X}_k^T \hat{\mathbf{y}}_k^i, X_q^T B_q + \sum_{k \neq q} \tilde{X}_k^T \hat{B}_k^i \right), \quad (3.25)$$

with  $QM_q + Q(K - 1)$  degrees of freedom, which can in addition be expressed as

$$\varphi(\tilde{X}^T \tilde{\mathbf{y}}, \tilde{X}^T \tilde{B}) \quad (3.26)$$

thanks to the parametrization introduced by (3.20). We therefore expect DASF to converge faster than plain Gauss-Seidel, as it is able to update multiple blocks of the optimization variable at each iteration. In addition, in the case of constrained problems, Gauss-Seidel's convergence is only guaranteed if the constraints are per-node separable [99], a restriction that is not present with DASF.

**Simulated example** Figure 3.2 depicts a simulated convergence example for DASF, Gauss-Seidel, and ADMM. The network consists of 10 nodes equipped with 10 channels each, collaborating to compute a multichannel Wiener filter expressed as

$$X^* = \underset{X}{\operatorname{argmin}} \mathbb{E} \left\{ \|X^T \mathbf{y} - \mathbf{d}\|^2 \right\}, \quad (3.27)$$

where  $\mathbf{d}$  is a known 2-dimensional signal.  $\mathbf{y}$  follows the data model

$$\mathbf{y} = A\mathbf{d} + \mathbf{n}. \quad (3.28)$$

where  $\mathbf{n}$  is a 10-dimensional white gaussian noise signal with a per-channel variance of 0.1.  $A$  is a random mixing matrix whose entries are sampled from a zero-mean gaussian distribution with unit variance. The entries of  $\mathbf{d}$  are sampled from the same distribution as  $A$ . The distributed ADMM implementation follows the algorithm in [40, Section 8.3] (further described in Subsection 3.4), with a learning rate of 1. Note that no meta-parameter search was performed for the learning rate. We count one unit of transmitted data every time a node sends a  $Q$ -dimensional signal to another node (i.e. no broadcasting is assumed).

We can observe that, by a simple algorithmic change and no significant additional data transmission, DASF significantly outperforms Gauss-Seidel. Although it

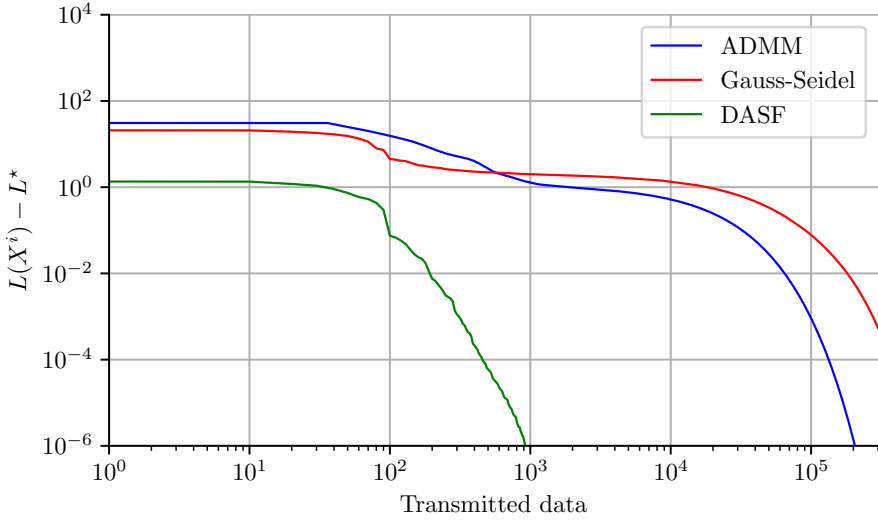


Figure 3.2: Simulation of the distributed computation of a Wiener filter based on a 1000 samples from  $\mathbf{y}$  and  $\mathbf{d}$ .  $L$  denotes the objective in (3.27) and  $L^*$  its optimal value.

performs better, ADMM requires more data transmission per iteration and suffers from the same drawback as Gauss-Seidel, as it does not fully use the degrees of freedom available at each local problem. Note that the first plateau displayed by all three algorithms corresponds to one full-round of network updates, such that every node has taken the role of updating node at least once. Progress is slower in those first few iterations, as the influence of the random initialization is still present in some blocks of the filter.

**Remark 3.2.** Note that the above simulations were performed in “batch-mode”, meaning that the same set of samples was used at each iteration. In a practical setting, we would use a different set of samples at each iteration. Due to the fact that the intermediate signals exchanged by ADMM are not guaranteed to be in the subspace spanned by the channels of  $\mathbf{y}$ , ADMM does not allow the iterations to be spread over multiple batches of samples. As a consequence, the same batch must be iterated over multiple times, resulting in a much higher number of data exchanges for ADMM (see Subsection 3.4 for details). This ability to spread iterations across multiple batches is one of the fundamental advantages of DASF over ADMM.

### 3.2.2 Arbitrary network topologies

A fully-connected network topology allows the updating node to receive the compressed data of every other node directly. In an arbitrary network topology, the updating node can only receive data from its neighbors. Applying the same procedure as in the fully-connected case, i.e. requiring the nodes to relay the compressed data of their neighbors to the updating node, would result in a significant communication overhead, most extreme in the case of line topologies. To avoid this, it was proposed in [58] to construct at each iteration a spanning tree that is rooted at the updating node, and let each node fuse (i.e. sum) the compressed data of its neighbors before relaying it to the neighbor closest to the updating node. In other words, the updating node will receive some linear combination of the compressed data of the rest of the network. With this interpretation in mind, the procedure described for fully connected networks can be readily applied, considering each branch at a given iteration *as if it were a single node*. The local variables  $X_k$  of each node in a branch are therefore updated with the same linear transformation (i.e. there is a single matrix  $\tilde{X}_{(\cdot)}$  per branch). Note that, because a new ad-hoc spanning tree is constructed at each iteration of the algorithm, the procedure is resilient to changes in topology and link failures, as long as they do not occur during the data aggregation phase and the underlying connectivity graph stays connected (i.e. there always exists a path between any two nodes). The procedure in arbitrary topologies is as follows.

**(i) Data aggregation** A spanning tree rooted at the updating node and preserving the links with its neighbors is computed in a distributed fashion, using, e.g., Algorithm 2.4. We denote the set of nodes in the subtree (i.e. branch) containing  $k$  and obtained by removing the link between  $k$  and  $q$  as

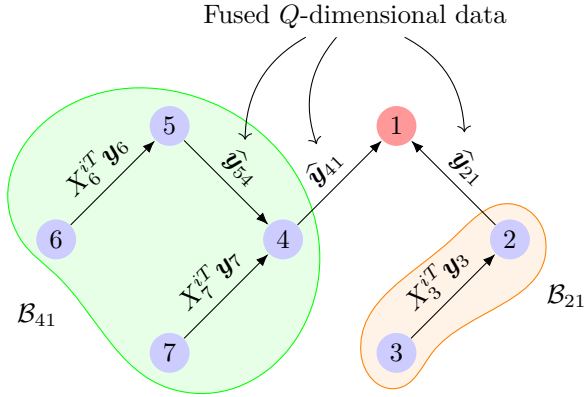


Figure 3.3: Example of an aggregation scheme in a spanning tree rooted at node 1. Transmission of  $\widehat{R}_k^i$ ,  $\widehat{B}_k^i$ ,  $\widehat{D}_k^i$  and  $\widehat{A}_k^i$  omitted.

$\mathcal{B}_{kq}$  (see Figure 3.3 for an example). We denote the compressed data associated with subtree  $\mathcal{B}_{kq}$  as

$$\begin{aligned} \widehat{\mathbf{y}}_{kq}^i &= \sum_{l \in \mathcal{B}_{kq}} \widehat{\mathbf{y}}_l^i, \\ \widehat{B}_{kq}^i &= \sum_{l \in \mathcal{B}_{kq}} \widehat{B}_l^i, \text{ and} \\ \widehat{D}_{j,kq}^i &= \sum_{l \in \mathcal{B}_{kq}} \widehat{D}_{j,l}^i. \end{aligned} \tag{3.29}$$

This can be computed recursively by having each node  $k$  in the subtree sum the data it receives from its children, and then forward the result to its parent. This ensures that the dimension of the data sent by each node stays equal to  $Q$ , independently of the network size and topology, making the communication of the compressed  $N$ -sample batches fully scalable. The full aggregation procedure is formally described by Algorithm 3.3 and an illustrative example is shown in Figure 3.3. Upon completion, the updating node  $q$  has access to the compressed data  $\widehat{\mathbf{y}}_{kq}^i$ ,  $\widehat{B}_{kq}^i$  and  $\widehat{D}_{j,kq}^i$  of each of its neighbors  $n \in \mathcal{N}_q$ .

**(ii) Local solution** The updating node constructs the local data  $\tilde{\mathbf{y}}^i$  as

$$\tilde{\mathbf{y}}^i \triangleq \begin{bmatrix} \mathbf{y}_q \\ \tilde{\mathbf{y}}_{n_1 k}^i \\ \vdots \\ \tilde{\mathbf{y}}_{n_L k}^i \end{bmatrix}, \quad \tilde{B}^i \triangleq \begin{bmatrix} B_q \\ \hat{B}_{n_1 k}^i \\ \vdots \\ \hat{B}_{n_L k}^i \end{bmatrix} \quad \text{and} \quad \tilde{D}_j^i \triangleq \begin{bmatrix} D_{j,q} \\ \hat{D}_{j,n_1 k}^i \\ \vdots \\ \hat{D}_{j,n_L k}^i \end{bmatrix} \quad (3.30)$$

where  $\{n_1, \dots, n_L\} = \mathcal{N}_q$  are the neighbors of node  $q$ .

It then obtains  $\tilde{X}^*$  by solving  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i)$ . In the case of multiple solution, the one closest to  $\tilde{X}^i$  as defined in (3.19) is selected.

**(iii) Parameters update**  $\tilde{X}^*$  is now partitioned correspondingly to (3.30) as

$$\tilde{X}^* \triangleq \begin{bmatrix} X_q^* \\ \tilde{X}_{n_1}^* \\ \vdots \\ \tilde{X}_{n_L}^* \end{bmatrix}. \quad (3.31)$$

Similarly to the fully-connected case, the updating node updates its filter with  $X_q^*$  and for each branch  $\mathcal{B}_{n_i q}$ , the nodes all update their filters according to  $X_k^{i+1} \leftarrow X_k^i \tilde{X}_{n_i}^*$  for all  $k \in \mathcal{B}_{n_i q}$  (Note that  $\tilde{X}_{n_i}^*$  is the same for every node in the branch). The full algorithmic procedure is described by Algorithm 3.4.

**Remark 3.3.** The dimension of the local problem  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i)$  directly depends on the number of neighbors of the updating node kept when building the spanning tree. Indeed, the local optimization variable will have dimension  $(Q|\mathcal{N}_q| + M_q) \times Q$ . We could in practice build a spanning tree ignoring some of the links between the updating node and its neighbors, but this would not yield any savings in required bandwidth (every node still needs to forward its data to some node), and it would be at the expense of convergence speed, as the available degrees of freedom available when minimizing the local problems  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i)$  would then be lower.

---

**Algorithm 3.3:** Recursive aggregation procedure in a tree-topology network rooted at node  $q$ .

---

**input :** Parent node  $p_k$  and set of childrens  $\mathcal{C}_k$  for each node  $k \neq q$  (the parent node is the neighbor closest to the updating node  $q$ ,  $\mathcal{C}_k = \emptyset$  for leaf nodes)

**begin**

**At node  $k$**

        Collect a new batch of samples of  $\mathbf{y}_k(t)$

        Wait for the aggregate compressed data  $\hat{\mathbf{y}}_{l_k}^i, \hat{B}_{kq}^i$  and  $\hat{D}_{j,kq}^i$  received from children.

        Send  $\hat{\mathbf{y}}_{kp_k}^i = \hat{\mathbf{y}}_k^i + \sum_{l \in \mathcal{C}_k} \hat{\mathbf{y}}_{lk}^i$  to  $p_k$  and similarly for  $\hat{B}_{kp_k}^i$  and  $\hat{D}_{j,kq}^i$ .

---



---

**Algorithm 3.4:** DASF algorithm in arbitrary network topologies

---

**begin**

$i \leftarrow 0, q \leftarrow 1$ , Randomly initialize  $X^0$

**loop**

        Construct a spanning tree rooted at node  $q$ .

        Aggregate the data according to Alg. 3.3.

**At node  $q$**

            Obtain  $\tilde{X}^*$  by solving  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i)$ . Select the solution closest to  $\tilde{X}^i$  in case of multiple solutions.

            Extract the  $\tilde{X}_k^*$ 's from  $\tilde{X}^*$  according to (3.31).

            Update the filter of node  $q$  with  $\tilde{X}_q^*$ .

**for  $n \in \mathcal{N}_q$  do**

                Send  $\tilde{X}_n^*$  to node  $n$ .

**for  $l \in \mathcal{B}_{nq}$  do**

**At node  $l$**

                        Wait for  $\tilde{X}_n^*$  and forward it to its children.

$X_l^{i+1} \leftarrow X_l^i \tilde{X}_n^*$

$i \leftarrow i + 1, q \leftarrow (q + 1) \bmod K$

---

### 3.3 Convergence and optimality of DASF

This section contains a proof of the convergence and optimality of DASF in fully-connected networks. We purposely skip the case of arbitrary network topologies, as this case is treated more in-depth for the non-smooth case in Chapter 4. This section only describes a subset of the results of [62], to which we refer the reader for more details.

For notational convenience, we define the following

$$L(X) \triangleq \varphi(X^T \mathbf{y}, X^T B), \quad (3.32a)$$

$$h_j(X) \triangleq \eta_j(X^T \mathbf{y}, X^T D_j). \quad (3.32b)$$

We also denote the global constraint set defined by (3.8b)–(3.8c) as

$$\begin{aligned} \mathcal{X} \triangleq \{X \mid \forall j \in \mathcal{J}_I, \eta_j(X^T \mathbf{y}(t), X^T D_j) \leq 0, \\ \forall j \in \mathcal{J}_E, \eta_j(X^T \mathbf{y}(t), X^T D_j) = 0\}. \end{aligned} \quad (3.33)$$

allowing us to express the original global problem (3.1) as

$$\min_{X \in \mathbb{R}^{M \times Q}} L(X) \quad \text{s.t.} \quad X \in \mathcal{X}. \quad (3.34)$$

#### 3.3.1 From global to local problems and back

Before delving into the actual proof, we give some intuition about the relationship between global and local problems in the case of fully-connected networks.

From (3.14)–(3.15), it can be observed that  $\tilde{\mathbf{y}}^i$ ,  $\tilde{B}^i$  and  $\tilde{D}_j^i$  are linear (compressive) transformations of  $\mathbf{y}$ ,  $B$  and  $D_j$ . Therefore, these variables can be related via a matrix transformation  $C_q^i$ , such that  $\tilde{\mathbf{y}}^i = C_q^{iT} \mathbf{y}$ ,  $\tilde{B}^i = C_q^{iT} B$ , and  $\tilde{D}_j^i = C_q^{iT} D_j$ . From (3.14)–(3.15), it is clear that the matrix  $C_q^i$  is constructed from the entries in  $X^i$ , such that we can define  $C_q^i$  as the result of a matrix-valued function  $C_q(\cdot)$  such that

$$C_q^i \triangleq C_q(X^i). \quad (3.35)$$

More specifically, we define

$$\begin{aligned} \Theta_{-q}(X) \triangleq \text{BlockDiag}(X_1, \dots, X_{q-1}) \quad \text{and}, \\ \Theta_{+q}(X) \triangleq \text{BlockDiag}(X_{q+1}, \dots, X_K) \end{aligned} \quad (3.36)$$

from which we construct the linear matrix-to-matrix map

$$C_q(X) = \left[ \begin{array}{c|c|c} O & \Theta_{-q}(X) & O \\ \hline I_{M_q} & O & O \\ \hline O & O & \Theta_{+q}(X) \end{array} \right] \quad (3.37)$$

where the  $O$ 's denote all-zero matrices of appropriate dimensions. With this notation (3.14)–(3.15) can be written as

$$\begin{aligned} \tilde{\mathbf{y}}^i &= C_q(X^i)^T \mathbf{y}, \\ \tilde{B}^i &= C_q(X^i)^T B, \text{ and} \\ \tilde{D}_j^i &= C_q(X^i)^T D_j. \end{aligned} \quad (3.38)$$

An update rule naturally emerges from the parametrizations introduced by (3.38), as by simple associativity,

$$\begin{aligned} \tilde{X}^{*T} \tilde{\mathbf{y}}^i &= \tilde{X}^{*T} (C_q(X^i)^T \mathbf{y}) \\ &= \left( C_q(X^i) \tilde{X}^* \right)^T \mathbf{y} \\ &= X^{i+1T} \mathbf{y} \end{aligned} \quad (3.39)$$

(and similarly for  $B$  and  $D_j$ ), which means that we can also re-express (3.21)–(3.22) in matrix form as

$$X^{i+1} \leftarrow C_q(X^i) \tilde{X}^*. \quad (3.40)$$

This shows that the local problem  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i)$  at node  $q$  can be interpreted as a parametrized version of the centralized problem (3.8), where the optimization variable is now constrained to a smaller linear subspace defined by the column space of  $C_q(X^i)$ . Based on (3.39)–(3.40), we find that the variable  $X^{i+1}$  that is obtained by solving the local problem followed by the updates (3.21)–(3.22) must be a solution of the following problem:

$$\begin{aligned} X^{i+1} &\in \underset{X}{\operatorname{argmin}} L(X) \\ \text{s.t.} \quad &X \in \mathcal{X} \\ &X \in \operatorname{range} C_q(X^i), \end{aligned} \quad (3.41)$$

with  $\text{range}$  denoting the range or column space operator<sup>5</sup>. The block structure of  $C_q(X^i)$  allows us to further express this constraint set as

$$\begin{aligned} \mathcal{S}_q(X) \triangleq \text{range } C_q(X) = \\ \text{range } X_1 \times \cdots \times \text{range } X_{q-1} \times \mathbb{R}^{I_{\mathcal{M}_q} \times Q} \\ \times \text{range } X_{q+1} \times \cdots \times \text{range } X_K. \end{aligned} \quad (3.42)$$

The new constraint thus simply restricts each block  $X_k$  of the optimization variable  $X$  to stay within the range of the corresponding block  $X_k^i$  of  $X^i$  (at the previous iteration), except for the block associated with the updating node  $q$ , which can move freely within the original constraint set. Solving this equivalent problem and updating  $X^{i+1}$  with its solution is effectively equivalent to performing the *Local solution* and *Parameters update* steps described earlier in Subsection 3.2.1. Figures 3.4 and 3.5 illustrate the relationship between global and local spaces schematically.

We define the map producing the solutions of the local problem, expressed in the global space of  $X$ , as

$$\mathcal{F}_q(X) \triangleq \underset{U \in \mathcal{X} \cap \mathcal{S}_q(X)}{\text{argmin}} L(U). \quad (3.43)$$

We also introduce the map<sup>6</sup>

$$\mathcal{M}_q(X) \triangleq \underset{U \in \mathcal{F}_q(X)}{\text{argmin}} \|U - X\|. \quad (3.44)$$

which allows us to summarize the full procedure as

$$X^{i+1} = \mathcal{M}_{q^i}(X^i). \quad (\text{DASF})$$

In the rest of this section, we study the properties of this map, and show that they lead to a sequence  $(X^i)_{i \in \mathbb{N}}$  that converges to interesting points, i.e. stationary points of problem (3.8).

---

<sup>5</sup>With a slight abuse of notation, as  $X$  has  $Q$  columns. We thus actually mean the  $Q$ -th Cartesian power of the column space.

<sup>6</sup>In [62] and Algorithm 3.2, we minimize the distance in the local space with  $\tilde{X}^i$  rather than with  $X^i$ . Doing so here lightens the notation, and is still conceptually correct, but from a practical algorithmic point of view,  $X^i$  should also be transmitted to the updating node for the selection to be possible in this setting.

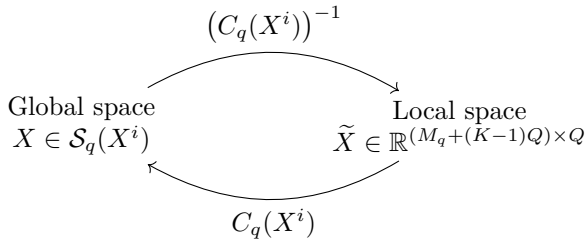


Figure 3.4: Equivalence between the global and local spaces. The linear map  $C_q(X^i) : \mathcal{S}_q(X^i) \rightarrow \mathbb{R}^{(M_q+(K-1)Q) \times Q}$  and its inverse map the local space to the global space and vice-versa. The inverse is well defined over  $\mathcal{S}_q(X^i)$  when  $C_q(X^i)$  has full column rank, as then the global and local space are homeomorphic to one another.

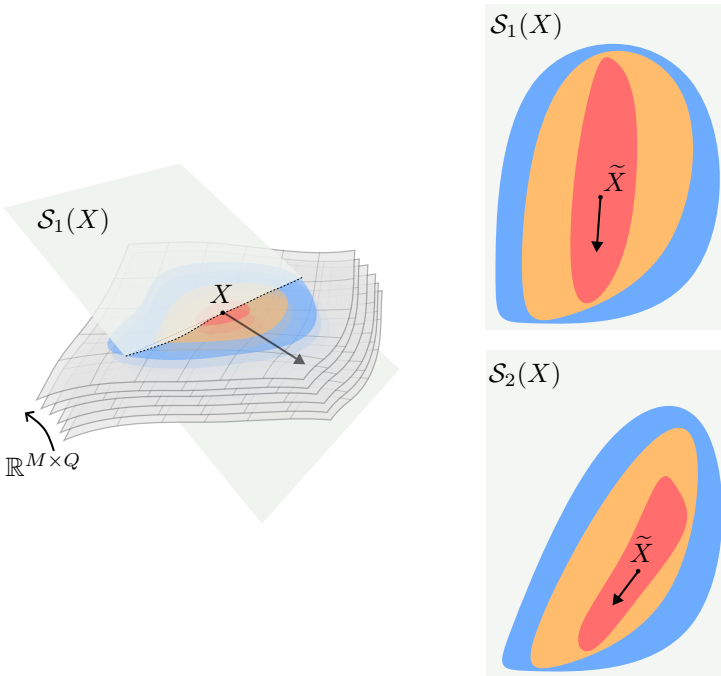


Figure 3.5: Schematic illustration of the relationship between global and local spaces. The local space is the result of the intersection of a linear space, whose orientation depends on  $X^i$ , with the original space. In other words, the updating node can only observe the data and cost function by looking through this linear space. Similarly, it can only manipulate the optimization variable  $X$  within the limits of this linear space.

**Remark 3.4** (Relationship with majorization-minimization schemes). In essence, DASF can be seen as a majorization-minimization (MM) scheme [101]–[103], where at each iteration, we minimize a *surrogate* function

$$s(U|X) = L(U) + \delta_{S_q(X)}(U) \quad (3.45)$$

with the properties that

$$s(U|X) \geq L(U) \quad \forall U \in \mathcal{X} \quad (3.46)$$

$$s(X|X) = L(X) \quad \forall X \in \mathcal{X}. \quad (3.47)$$

Similarly to (DASF), the optimization procedure can be summarized as

$$X^{i+1} \in \underset{U \in \mathcal{X}}{\operatorname{argmin}} s(U|X^i). \quad (3.48)$$

The existing convergence results for such schemes unfortunately do not apply here, as they typically require the surrogate function  $s$  to be smooth, or at least continuous with regards to both its arguments, and the problem to have a convex constraints set [104] (although problem specific variants lifting some of those requirements exist [105], [106]).

### 3.3.2 Key assumptions

The following assumptions capture our intuitions about a practical problem setting. They strive to be as general as possible, while being sufficiently restrictive to limit the necessary mathematical derivations in the proofs to a reasonable level.

**Assumption 3.1** (Compactness).  $L$  has compact sub-level sets when restricted to  $\mathcal{X}$ , i.e.

$$\{X \in \mathcal{X} \mid L(X) \leq m\} \quad (3.49)$$

is closed and bounded for any  $m \in \mathbb{R}$ .

This technical assumption captures some notion of finiteness about the sub-level sets of  $L$ . It ensures that a sequence does not drift away endlessly, and ensures that it always eventually visits back part of the sets it has already “seen”<sup>7</sup>.

<sup>7</sup>More specifically, this intuition captures the idea of a *sequentially compact* space, which in the case of a metric space, is equivalent to compactness.

**Assumption 3.2** (Well-posedness). *The solution set of  $\mathbb{P}(\cdot, \cdot, \cdot)$  varies continuously with the problem parameters, and therefore the set-valued maps  $\mathcal{F}_q : \mathbb{R}^{M \times Q} \rightrightarrows \mathcal{X}$  are continuous<sup>8</sup>. Furthermore, the maps  $\mathcal{M}_q$  are well-defined, in the sense that they are single-valued.*

This assumption is quite strong, but will be relaxed in later chapters. Still, by imposing the proper assumptions on the problems of interest, it can be shown to hold. For example, the Wiener filtering problem (3.9) and PCA (3.11) both satisfy this assumption if we can guarantee that  $R_{\mathbf{y}}$  keeps full rank and has non-degenerate eigenvalues (for the local problems as well). In a real-world setting, the presence of noise is unavoidable, and two given signals will be independent with overwhelming probability, ensuring that the associated covariance matrix is non-singular. The second part of this assumption regarding  $\mathcal{M}_q$  is also reasonable, as if it were not true, it would mean that the algorithm reaches a point which is exactly equally spaced from two solutions. Such a scenario should not happen with “regular” problems, and is in addition unstable, i.e., the next iteration would most likely not suffer from the same issue.

### 3.3.3 Convergence

In this first part of the proof, we show that under the two above assumptions, any sequence generated by the DASF algorithm converges to a fixed point of the procedure, i.e., a point  $X$  such that for any node  $q$ ,  $X = \mathcal{M}_q(X)$ .

We first prove the monotonic decrease of the sequence of costs  $L(X^i)$ .

**Lemma 3.1.** *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by (DASF). Then  $(L(X^i))_{i \in \mathbb{N}}$  is a monotonically decreasing sequence, and every point  $X^i$  is feasible, i.e.  $X^i \in \mathcal{X}$  for every  $i$ .*

*Proof.* By the definition (3.43) of  $\mathcal{F}_q$ , any point in  $\mathcal{F}_q(X)$  is in  $\mathcal{X}$  for any  $X$ ,  $X^i$  is therefore feasible. By the same definition,  $X^{i+1} \in \mathcal{F}_q(X^i)$  implies that for any  $X \in \mathcal{X} \cap \mathcal{S}_q(X^i)$ ,  $L(X^{i+1}) \leq L(X)$ . As  $X^i$  is feasible, and as by (3.42), it is also in  $\mathcal{S}_q(X^i)$  (each block of  $X^i$  is trivially in its own range), it must be that  $L(X^{i+1}) \leq L(X^i)$ .  $\square$

<sup>8</sup>The continuity of  $\mathcal{F}_q$  must be understood in the sense described in Subsection 1.2.2.

We use the above result to show that any accumulation point is a fixed point for the local update of at least one node  $q$ . This sub-result requires only a milder version of Assumption 3.2, which is the outer-semicontinuity of  $\mathcal{F}_q$ , rather than the full continuity required by Assumption 3.2.

**Lemma 3.2.** *Let Assumptions 3.1 hold, let  $(\bar{X}, q)$  be an accumulation point of the sequence  $(X^i, q^i)_{i \in \mathbb{N}}$  and  $\mathcal{F}_q$  be an outer-semicontinuous map, then*

$$\bar{X} = \mathcal{M}_q(\bar{X}). \quad (3.50)$$

*Proof.* Let  $\mathcal{I}$  be an index set such that the sequence  $(X^i, X^{i+1}, q^i)_{i \in \mathcal{I}}$  converges to some  $(\bar{X}, \bar{X}^{+1}, q)$  (such an index set always exists in a compact set<sup>a</sup>). By the definition of outer-semicontinuity,

$$X^{i+1} \in \mathcal{M}_q(X^i) \subseteq \mathcal{F}_q(X^i) \quad \forall i \in \mathcal{I} \quad (3.51)$$

implies that

$$\bar{X}^{+1} \in \mathcal{F}_q(\bar{X}). \quad (3.52)$$

We now show that, as  $\bar{X}^{+1}$ ,  $\bar{X}$  is also in the local solution set. We thus need to show that (a) it is in the constraint set of the local problem, and (b) that it achieves the minimum value. Because the sequence  $(X^i, q^i)_{i \in \mathbb{N}}$  lives in the sub-level sets of  $L$  restricted to  $\mathcal{X}$ , it must be that  $\bar{X} \in \mathcal{X}$  (as compactness implies closedness). Furthermore, by the definition of (3.42),  $\bar{X} \in \mathcal{S}_q(\bar{X})$ . Therefore

$$\bar{X} \in \mathcal{X} \cap \mathcal{S}_q(\bar{X}), \quad (3.53)$$

proving the first point. Because the sub-level sets of  $L$  are compact and  $L$  is continuous,  $L$  is lower-bounded in  $\mathcal{X}$  [70, Theorem 2.54]. As from Lemma 3.1,  $(L(X^i))_{i \in \mathbb{N}}$  is a lower-bounded monotonically decreasing sequence, it must converge to some value  $\bar{L}$  (Weierstrass' theorem). Therefore it must be that

$$L(\bar{X}) = \bar{L} = L(\bar{X}^{+1}) = \min_{U \in \mathcal{X} \cap \mathcal{S}_q(\bar{X})} L(U), \quad (3.54)$$

where the last equality flows from (3.52), proving the second point. We conclude that

$$\bar{X} \in \mathcal{F}_q(\bar{X}). \quad (3.55)$$

As  $\bar{X}$  is by definition closest to itself, this also implies that

$$\bar{X} = \mathcal{M}_q(\bar{X}). \quad (3.56)$$

□

---

<sup>a</sup>and the cartesian product of compact sets is compact as well.

We will now use the previous result, and show that this fixed-point property must be true for *any* node  $q$ , rather than only a single one (as was the case for Lemma 3.2).

**Proposition 3.1.** *Let Assumptions 3.1 and 3.2 hold. Then for any sequence  $(X^i)_{i \in \mathbb{N}}$  generated by (DASF), the following is true*

- (i) *Every accumulation point of  $(X^i)_{i \in \mathbb{N}}$  is a fixed point of  $\mathcal{M}_q$  for every  $q$ .*
- (ii)  $\lim_{i \rightarrow \infty} \|X^{i+1} - X^i\| = 0$ .

*Proof.* Let  $\mathcal{I}$  be an index set such that the sequence  $(X^i, X^{i+1}, q^i)_{i \in \mathcal{I}}$  converges to some  $(\bar{X}, \bar{X}^{+1}, q)$ . From Assumption 3.2 and Berge's maximum theorem [107], the continuity of  $\|\cdot\|$  and  $\mathcal{F}_q$  imply that  $\mathcal{M}_q$  is continuous<sup>a</sup>. Therefore,

$$X^{i+1} = \mathcal{M}_q(X^i) \quad \forall i \in \mathcal{I} \quad (3.57)$$

implies that

$$\bar{X}^{+1} = \mathcal{M}_q(\bar{X}). \quad (3.58)$$

As from Lemma 3.2, we have  $\bar{X} = \mathcal{M}_q(\bar{X})$ , it must be that  $\bar{X} = \bar{X}^{+1}$ . In addition, this last equality implies that

$$\bar{X} = \mathcal{M}_{q^1}(\bar{X}), \quad (3.59)$$

with  $q^n = q + n \pmod K$ . We can inductively apply the above reasoning to show that

$$\bar{X} = \mathcal{M}_{q^n}(\bar{X}), \quad (3.60)$$

for any  $n > 0$ , proving (i).

Consider that (ii) is false. Then we can find an index set  $\mathcal{I}'$  such that

$$\lim_{i \in \mathcal{I}'} \|X^{i+1} - X^i\| > 0, \quad (3.61)$$

as otherwise, 0 would be the only accumulation point, and the sequence would then be convergent (see Lemma F.2). But we can find another index set  $\mathcal{I}'' \subseteq \mathcal{I}'$  such that  $X^i \rightarrow \bar{X}$  and  $X^{i+1} \rightarrow \bar{X}^{+1}$  (not necessarily the same as above), and therefore

$$\lim_{i \in \mathcal{I}''} \|X^{i+1} - X^i\| = \|\bar{X} - \bar{X}^{+1}\| = 0, \quad (3.62)$$

where the last equality results from the fact that  $\bar{X} = \bar{X}^{+1}$ . This contradicts (3.61) and (ii) must therefore be true. □

---

<sup>a</sup>Berge's maximum theorem would imply that  $\mathcal{M}_q$  seen as a singleton-valued map is outer-semicontinuous (in the set analysis sense), which is equivalent to continuity when  $\mathcal{M}_q$  is seen as a function [70, Lemma 17.6].

Note that (i) and (ii) are equivalent when  $\mathcal{M}_q$  is a continuous single-valued map, but as will be seen in Chapter 4, it is not true anymore when we drop the single-value assumption.

### 3.3.4 Optimality

As we have shown that accumulation points are fixed points, it now remains to show that such fixed points are optimal in some sense. Without further assumption, we can only show stationarity.

This next result states that if some “compressed” version of the linear independence constraint qualification (LICQ) holds at the fixed point, then they are stationary points.

**Theorem 3.2.** *Let  $\bar{X}$  be a fixed point of (DASF) that also satisfies the following qualification: the matrices defined by*

$$\text{BlockDiag}(\bar{X}_1, \dots, \bar{X}_K)^T \nabla h_j(X) \quad (3.63)$$

*for any  $j$  corresponding to equality constraints or active<sup>9</sup> inequality constraints, are linearly independent. Then  $\bar{X}$  is a stationary point of the global problem (3.8) and it therefore satisfies its KKT conditions, i.e., there exist some  $\lambda_j$  such that*

$$\nabla L(\bar{X}) + \sum_{j \in \mathcal{J}_E \cap \mathcal{A}(\bar{X})} \lambda_j \nabla h_j(\bar{X}) = 0 \quad (3.64a)$$

$$\bar{X} \in \mathcal{X} \quad (3.64b)$$

$$\lambda_j \geq 0 \quad \forall j \in \mathcal{A}(\bar{X}) \quad (3.64c)$$

*where  $\mathcal{A}(\bar{X})$  denotes the set of indices of the active inequality constraints at  $\bar{X}$ .*

*Proof.* As  $\bar{X}$  is a fixed point of  $\mathcal{M}_q$  for every  $q$ , it is a solution of (3.43) for every  $q$ , i.e.

$$\bar{X} \in \underset{U \in \mathcal{X} \cap \mathcal{S}_q(X)}{\text{argmin}} L(U). \quad (3.65)$$

Noting that  $U \in \mathcal{S}_q(\bar{X})$  is equivalent to

$$\exists \tilde{X} : U = C_q(\bar{X})\tilde{X} \quad (3.66)$$

we can express the local problem in terms of the local variable  $\tilde{X}$  as

$$\tilde{X}^* \in \underset{\tilde{X}}{\text{argmin}} L(C_q(\bar{X})\tilde{X}) \quad (3.67a)$$

$$\text{s.t.} \quad \forall j \in \mathcal{J}_E : h_j(C_q(\bar{X})\tilde{X}) = 0 \quad (3.67b)$$

$$\forall j \in \mathcal{J}_I : h_j(C_q(\bar{X})\tilde{X}) \leq 0 \quad (3.67c)$$

and because  $\bar{X}$  is a fixed-point,  $\bar{X} = C_q(\bar{X})\tilde{X}^*$ . Using the chain rule, we can write down the KKT conditions associated with the above local

<sup>9</sup>An inequality constraint is said to be active if it is equal to 0.

problem as

$$C_q(\bar{X})^T \nabla L(\bar{X}) + \sum_{j \in \mathcal{J}_E \cap \mathcal{A}(\bar{X})} \lambda_j^q C_q(\bar{X})^T \nabla h_j(\bar{X}) = 0 \quad (3.68a)$$

$$\bar{X} \in \mathcal{X} \quad (3.68b)$$

$$\lambda_j \geq 0 \quad \forall j \in \mathcal{A}(\bar{X}), \quad (3.68c)$$

where the  $\lambda_j^q$  are the multipliers associated with the local problem at a particular node  $q$ . Note that because

$$\text{range BlockDiag}(\bar{X}_q, \bar{X}_1, \dots, \bar{X}_{q-1}, \bar{X}_{q+1}, \dots, \bar{X}_K) \subseteq \text{range } C_q(\bar{X}) \quad (3.69)$$

the independence condition of (3.63) implies the independence of the  $C_q(\bar{X})^T \nabla h_j(\bar{X})$  for any  $q$ , therefore the local problem satisfy the linear independence constraint qualification, which ensure that if  $\bar{X}$  is a minimizer of the local problem, then it satisfies the above KKT conditions. Let us look at the rows of (3.68a) corresponding to  $\bar{X}_q$ , and associated with the unconstrained block  $q$ , and thus corresponding to the rows with the identity block in (3.37), we have

$$\nabla_q L(\bar{X}) + \sum_{j \in \mathcal{J}_E \cap \mathcal{A}(\bar{X})} \lambda_j^q \nabla_q h_j(\bar{X}) = 0 \quad (3.70)$$

Because  $\bar{X}$  is a local solution for any  $q$ , we can stack the above equations across  $q$  and obtain

$$-\nabla L(\bar{X}) = \sum_{j \in \mathcal{J}_E \cap \mathcal{A}(\bar{X})} \begin{bmatrix} \lambda_j^1 \nabla_1 h_j(\bar{X}) \\ \vdots \\ \lambda_j^K \nabla_K h_j(\bar{X}) \end{bmatrix}. \quad (3.71)$$

We now need to prove that  $\lambda_j^1 = \dots = \lambda_j^K$  for every  $j$ , as therefore the above equation will be equivalent to the condition (3.64a). Let us first note that because of (3.69), (3.68a) implies that<sup>a</sup>.

$$\Gamma^T \nabla L(\bar{X}) + \sum_{j \in \mathcal{J}_E \cap \mathcal{A}(\bar{X})} \lambda_j^q \Gamma^T \nabla h_j(\bar{X}) = 0, \quad (3.72)$$

where  $\Gamma$  denotes  $\text{BlockDiag}(\bar{X}_1, \dots, \bar{X}_K)$  for conciseness. Because of the linear independence of the  $\Gamma^T \nabla h_j(\bar{X})$ , the  $\lambda_j^q$  are uniquely defined and  $\lambda_j^1 = \dots = \lambda_j^K$ . We conclude that  $\bar{X}$  satisfies the KKT conditions of the global problem.  $\square$

<sup>a</sup>Note that  $\Gamma$  and the block matrix in (3.69) are equal up to a permutation of two blocks of rows.

The qualification can be shown to hold for many kinds of constraints [62], such as the generalized Stiefel manifold

$$\{X \in \mathbb{R}^{M \times Q} \mid X^T A X = I_Q\}, \quad (3.73)$$

for some  $A \in \mathbb{R}^{M \times M}$ , and linear constraints

$$\{X \in \mathbb{R}^{M \times Q} \mid X^T D = H\}, \quad (3.74)$$

with  $H \in \mathbb{R}^{Q \times L}$  when  $H$  has full column-rank (see [62] for details).

We can now summarize the above results in a single theorem stating the convergence and optimality of DASF.

**Theorem 3.3.** *Let the qualification of Theorem 3.2 be satisfied for any feasible point of problem (3.8), and Assumptions 3.1 and 3.2 hold. Then any sequence generated by Algorithm 3.2 converges to the set of stationary points of problem (3.8). Furthermore, if the problem has finitely many stationary points, then the sequence converges to a single point.*

*Proof.* From Proposition 3.1, the accumulation points of a sequence generated by Algorithm 3.2 are fixed point of the full procedure. From Theorem 3.2, the fixed points are stationary points of problem (3.8). The last statement flows directly from Lemma F.3.  $\square$

Note that we can make a statement similar to Theorem 2.3 regarding the stability of accumulation points that are not local or global minima. We therefore expect that in the presence of noise, a sequence generated by DASF would converge to a minimum (local or otherwise) of problem (3.8) (see [62, Theorem 7] for details). See [62] for worked-out examples showing that the qualification and assumptions of Theorem 3.3 do hold for many common problems.

As mentioned earlier, all the results in this section can be extended to the case of arbitrary network topologies. See [62] for details.

### 3.3.5 Convergence issues

In order to prove convergence of the DASF algorithm, we assumed that the solution set of (3.8) varied continuously with the data  $\mathbf{y}$ . Formally, we required the set-valued map defined by (3.43) to be continuous (see e.g. [70] for details), which means that for an infinitesimal perturbation of the data  $\mathbf{y}$ , any point in the solution set can be made arbitrarily close from at least one point in the perturbed solution set. Unfortunately, there are some edge cases where this assumption does not hold, as described in the examples hereafter.

**Example 1: Least Squares Regression** Consider the case of least squares regression or Wiener filtering, where the objective function is given by

$$\mathbb{E} \{ \|X^T \mathbf{y} - \mathbf{d}\|_F^2 \} \quad (3.75)$$

and the solution set is given by

$$X^* \in \mathbb{E} \{ \mathbf{y}\mathbf{y}^T \}^\dagger \mathbb{E} \{ \mathbf{y}\mathbf{d} \} + \text{Null}(\mathbb{E} \{ \mathbf{y}\mathbf{y}^T \}). \quad (3.76)$$

where  $\text{Null}(\mathbb{E} \{ \mathbf{y}\mathbf{y}^T \})$  denotes the null space of the covariance matrix of  $\mathbf{y}$  and  $(\cdot)^\dagger$  the Moore-Penrose pseudo-inverse. Here, a slight perturbation of the data  $\mathbf{y}$  can make an entire dimension of the null space disappear, making the solutions corresponding to that dimension finitely far from the new solution set. This particular example has an easy work-around: to always select the least norm solution.

**Example 2: Rayleigh Quotient Minimization** Rayleigh quotient minimization (the underlying optimization problem associated with PCA (3.11)) is typically performed by solving

$$\min_{X^T X = I} \text{Tr} (X^T \mathbb{E} \{ \mathbf{y}\mathbf{y}^T \} X). \quad (3.77)$$

where a solution corresponds to the eigenvectors of the covariance matrix of  $\mathbf{y}$  associated with the smallest eigenvalues. Now consider some  $\mathbf{y}^\diamond$  such that the  $Q$  and  $Q + 1$  smallest eigenvalues are equal. Then, a slight perturbation of  $\mathbf{y}$  around  $\mathbf{y}^\diamond$  can result in the  $Q + 1$  smallest eigenvalue becoming smaller than the  $Q$  smallest eigenvalue, and vice-versa, making the solution correspond to an altogether different set of eigenvectors, and the solution set seen as a set-valued function of  $\mathbf{y}$  would have a discontinuity at  $\mathbf{y}^\diamond$ . Although the algorithm would still converge in terms of objective function values, such lack of continuity can lead to oscillations of the filters which could make the algorithm impractical.

Both those examples are a consequence of the fact that, when Assumption 3.2 does not hold, we cannot anymore guarantee the validity of Proposition 3.1, and in particular point (ii) that states that

$$\lim_{i \rightarrow \infty} \|X^{i+1} - X^i\| = 0. \quad (3.78)$$

A possible solution to this problem could be to embed the post-hoc step associated with the selection of the solution minimizing  $\|\tilde{X} - \tilde{X}^i\|_F$  inside the local problem. That is, instead of solving the local problem

$$\min_{\tilde{X} \in \tilde{\mathcal{X}}} \varphi(\tilde{X}^T \tilde{\mathbf{y}}, \tilde{X}^T \tilde{B}), \quad (3.79)$$

the updating node would solve

$$\min_{\tilde{X} \in \tilde{\mathcal{X}}} \varphi(\tilde{X}^T \tilde{\mathbf{y}}, \tilde{X}^T \tilde{B}) + \lambda \|\tilde{X} - \tilde{X}^i\|_F^2, \quad (3.80)$$

where we denote the constraint set of the local problem by  $\tilde{\mathcal{X}}$ , and  $\lambda$  is a strictly positive regularization parameter, which must be selected a priori.

This modification comes with the significant drawback that problem (3.80) cannot be solved with the same solver as the global problem (3.8), and a dedicated solver is therefore required for the local problems<sup>10</sup>.

In Chapter 5, we prove the convergence of this scheme, along with much more efficient methods avoiding the convergence issues described in the present subsection.

**Simulated example** We illustrate the convergence issues arising from Example 2 with a hand-picked case. We applied both DASF and the suggested regularization, which we denote ‘‘DASF-R’’, to problem (3.77), with  $M = 100$ ,  $Q = 2$ ,  $K = 10$  and  $\lambda = 2$ . The data  $\mathbf{y}$  was specifically crafted to ensure that the second and third eigenvalues are equal. As can be seen on Figure 3.6, once close enough to the solution, DASF oscillates between two solutions, different in only a single column (the chordal distance therefore alternates between 0 and 1). DASF-R, on the other hand, converges to one of the solutions, without oscillations. But, as can be observed in Figure 3.7, this comes at the cost of a reduced convergence speed, caused by the damping introduced by the regularizing term. Note that by selecting the appropriate value for  $\lambda$ , one can obtain a trade-off between stability of the solution and convergence speed.

<sup>10</sup>The operator producing the set of minimizers of (3.80) is in fact the proximal operator associated with  $\tilde{X} \mapsto \varphi(\tilde{X}^T \tilde{\mathbf{y}}, \tilde{X}^T \tilde{B}) + \delta_{\tilde{\mathcal{X}}}(\tilde{X})$ , and, if some function are said to be ‘‘proximable’’, i.e. their prox-operator can be efficiently computed, this is not the case for most functions [108].

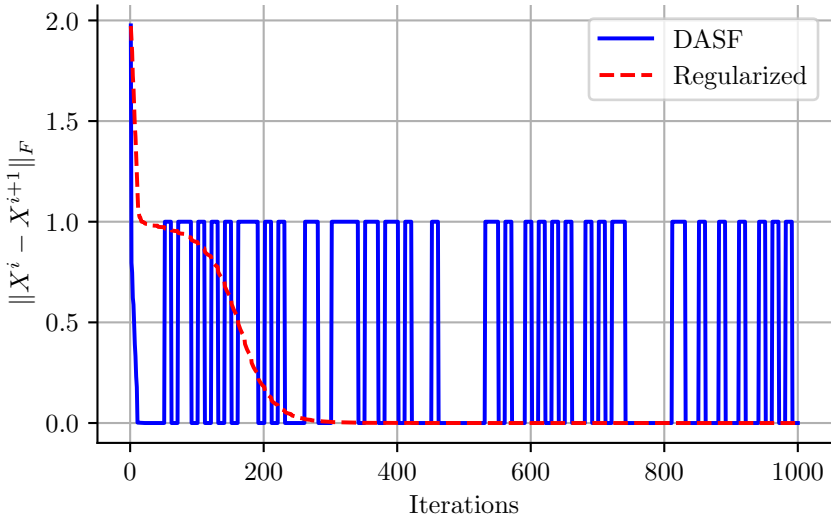


Figure 3.6: Chordal distance between the span of two successive iterates for a single instance of problem (3.77). A distance of 1 indicates that the subspaces have one orthogonal component.

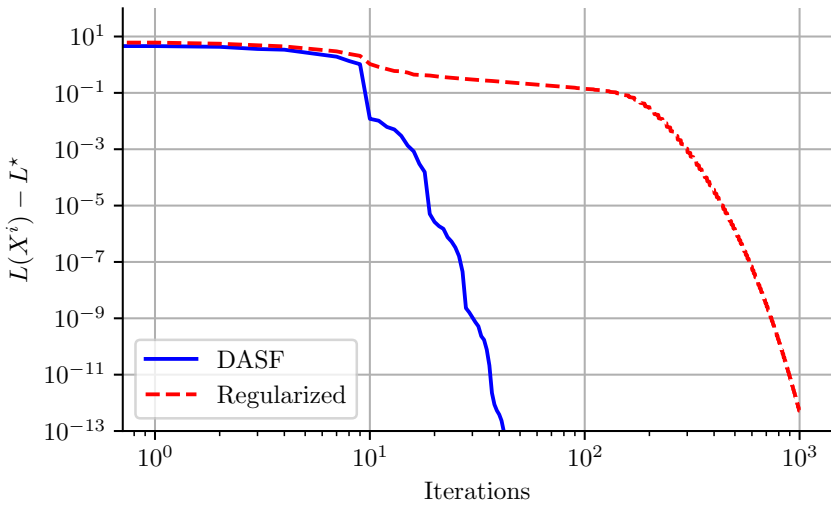


Figure 3.7: Convergence in objective values for a single instance of problem (3.77).  $L$  denotes the objective in (3.77), and  $L^*$  its minimum value.

### 3.4 A note on ADMM

The alternating direction method of multipliers (ADMM) [40] is a well-known and well-studied method to solve convex problems of the same shape as (3.1). The algorithm resulting from the application of ADMM usually admits a “natural” distributed formulation [109]. More specifically, ADMM solves problems of the shape of (4.4), studied in Chapter 4 and summarized hereafter

$$\begin{aligned}
 X^* \in \operatorname{argmin}_{X \in \mathbb{R}^{M \times Q}} \varphi(X^T \mathbf{y}(t), X^T B) + \sum_k \gamma_k(X_k) \\
 \text{s.t. } X_k \in \mathcal{X}_k \quad \forall k.
 \end{aligned} \tag{3.81}$$

Note here that, contrarily to the original DASF formulation (3.8), the constraint set is now required to be per-node separable (and is summarized here by the abstract sets  $\mathcal{X}_k$ ). The problem also allows the addition of a per-node non-smooth regularizer  $\gamma_k$  (see Chapter 4 for an in-depth description of this problem formulation). A naive implementation of ADMM can be described as follows.

**(i) Data collection** Each node  $k \neq q$  collects a batch of samples of  $\mathbf{y}_k$ .

**(ii) Main subroutine** The nodes iterate over the following substeps until convergence is practically achieved.

**(ii.i) Data aggregation and parallel update** Each node  $k \neq q$  computes in parallel

$$X_k^{i+1} = \operatorname{argmin}_{X_k \in \mathcal{X}_k} \gamma_k(X_k) + \frac{\rho}{2} \mathbb{E} \left\{ \left\| (X_k - X_k^i)^T \mathbf{y}_k + \frac{1}{K} \mathbf{z}^i + \mathbf{u}^i - \mathbf{v}^i \right\|_F^2 \right\},$$

where  $\rho$  is an appropriately chosen step-size, and transmits  $X_k^{i+1T} \mathbf{y}_k$  to the updating node  $q$ . The updating node only needs access to  $\mathbf{z}^{i+1} = \sum_k \hat{\mathbf{y}}_k$ , and a recursive aggregation procedure can therefore be used.

**(ii.ii) Local update** Node  $q$  computes

$$\mathbf{v}^{i+1} = \operatorname{argmin}_{\mathbf{v}} \varphi \left( \frac{1}{K} \mathbf{v} \right) + \frac{K\rho}{2} \mathbb{E} \left\{ \left\| \frac{1}{K} \mathbf{z}^{i+1} + \mathbf{u}^i - \mathbf{v} \right\|_F^2 \right\}$$

and

$$\mathbf{u}^{i+1} = \frac{1}{K} \mathbf{z}^{i+1} + \mathbf{u}^i - \mathbf{v}^{i+1}$$

and transmits back  $\mathbf{u}^{i+1}$  and  $\mathbf{v}^{i+1}$  to the other nodes.

Note that the substeps (ii.i) and (ii.ii) are performed using the same batch of samples, until convergence is achieved for that particular batch.

Although ADMM has been shown to converge for convex instances of problem (3.81), its use in an adaptive setting would be very inefficient. With DASF, the inter-iteration memory is completely stored in the iterates  $X^i$ , which are not tied to a particular sensor batch, i.e., the next iteration can make use of  $X^i$  with a new sample batch to compute the next iterate  $X^{i+1}$ , therefore spreading the computations of the filter over multiple batches. In the case of ADMM, the inter-iteration memory is embedded in the iterates  $X^i$ , but also in the signals  $\mathbf{v}^i$  and  $\mathbf{u}^i$  associated with a particular batch of samples. As soon as a new batch of samples is used to compute  $\mathbf{z}^i$  and estimate the statistics of  $\mathbf{y}_k$ , the information contained in  $\mathbf{v}^i$  and  $\mathbf{u}^i$  becomes irrelevant as both signals correspond to different samples, and the cross-covariance between  $\mathbf{y}_k$  and  $\mathbf{u}^i$  or  $\mathbf{v}^i$  cannot be estimated prior to the parallel update step. This procedure therefore only converges if the same batch of samples is iterated over (and transmitted via  $\hat{\mathbf{y}}_k$ ) multiple times.

### 3.5 Discussion

In this chapter, we have given an overview of the DASF algorithm, and showed its benefits relative to a naive Gauss-Seidel approach. In order to prove its convergence, we have assumed that the problems of interest were well-posed, in the sense their solution set varies continuously with their input parameters. If this assumption generally holds, it can become prohibitive for certain problems, of which we have given examples. Furthermore, we assumed that the objective function in (3.8) was smooth, precluding the use of some particularly useful regularizers, such as sparsity-inducing norms. In the next chapter, we lift this limitation of smoothness, and show that DASF can, with some restrictions, be applied to non-smooth spatial filtering problems. In Chapter 5, we will lift the well-posedness assumption, and therefore make DASF readily applicable to a broader class of problems.



## Chapter 4

---

# Non-smooth DASF

---

This chapter is based on

- \* **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems”, in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5
- \* **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems in wireless sensor networks”, *IEEE Transactions on Signal Processing*, vol. 72, pp. 4682–4697, 2024 (in second round of review for IEEE’s Transactions on Signal Processing)

## Chapter abstract

In this chapter, we show that the DASF algorithm can be extended to compute the filters associated with a certain class of non-smooth optimization problems. This extension makes the addition of sparsity-inducing norms to the problem's cost function possible, allowing sensor selection to be performed in a distributed fashion, alongside the filtering task of interest, thereby further reducing the network's energy consumption compared to the original DASF algorithm. We provide a description of the algorithm, prove its convergence, and validate its performance and solution tracking capabilities with numerical experiments.

## 4.1 Introduction

This chapter extends the distributed adaptive signal fusion (DASF) algorithm [58], described in the previous chapter, which was originally designed to solve distributed features problems in the particular context of spatial filtering in WSNs, where the sensor channels are distributed across the nodes of a WSN. As a reminder, the DASF framework provides a generic “meta-algorithm” that computes adaptive spatial filters whose coefficients are the solutions of some smooth optimization problem, where the latter defines the optimal centralized spatial filter based on the network-wide signal statistics (which are assumed to be unknown at start-up). The DASF algorithm relies on the exchange of linearly compressed views of the nodes’ data, which are then used by each node to locally solve a subproblem preserving the original problem structure, and iteratively producing a better estimate of the optimal filter coefficients. By spreading iterations of the algorithm over different sample batches, DASF is able to adaptively track the optimal filters based on their evolving statistics, which are estimated online using the most recently collected samples. The DASF framework is applicable to a wide class of spatial filtering problems, including the trace ratio problem [110], principal component analysis [30], generalized eigenvalue problems [78], minimum mean squared error filtering, minimum variance beamforming [36], and single and multi-view canonical correlation analysis [80]. However, the DASF algorithm requires the optimization objective to be smooth, which notably prevents the use of the  $\ell_1$  norm, which is often used in signal processing to encourage sparsity in the solutions. Our focus in this chapter, is the extension of the DASF algorithm to non-smooth problems, with the side goal of performing  $\ell_1$ -induced node or sensor selection alongside a given filtering task (rather than performing an a priori node selection based on some task-agnostic criterion as was suggested in Chapter 2).

Our contribution is three-fold. Firstly, we propose an extension of the original DASF algorithm to certain classes of *non-smooth* adaptive spatial filtering problems, referred to as non-smooth DASF (NS-DASF). Secondly, we provide a convergence and optimality proof for the NS-DASF algorithm based on milder assumptions than the original DASF algorithm. The new assumptions and optimality in the case of non-smooth problems lead to a different proof strategy compared to the convergence proof of the smooth version of DASF. Finally, we apply our algorithm to the problem of node selection in WSNs, where only a subset of the nodes is required to contribute to the filtering task. In this chapter, we show via numerical experiments that the problem of (distributed) node selection can be solved concurrently with the filtering task, by the addition of an appropriate regularizer to the optimization problem.

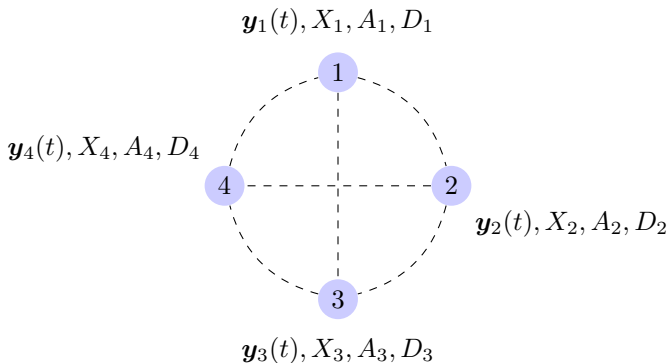


Figure 4.1: An example 4 nodes fully-connected network with associated filters and data.

The outline of this chapter is as follows. In Section 4.2 we re-formalize the scope of DASF and NS-DASF in a WSN context. In Section 4.3 we describe the NS-DASF algorithm. We first introduce the simpler case of fully-connected networks, before extending the description to arbitrary network topologies. In Section 4.4, we prove the convergence and optimality of NS-DASF. Section 4.5 describes several numerical experiments, and Section 4.6 concludes the chapter with a brief discussion.

## 4.2 Problem scope

As a reminder, we reiterate the general WSN setting hereafter.

We consider a network of  $K$  nodes with labels in  $\mathcal{K} = \{1, \dots, K\}$ . Each node  $k$  senses an  $M_k$ -channel stochastic signal  $\mathbf{y}_k(t)$  with values in  $\mathbb{R}^{M_k}$  for each sample  $t \in \mathbb{Z}$ . We denote the network-wide  $M$ -channel signal with values in  $\mathbb{R}^M$  as

$$\mathbf{y}(t) \triangleq \begin{bmatrix} \mathbf{y}_1(t), \\ \vdots \\ \mathbf{y}_K(t) \end{bmatrix}, \quad (4.1)$$

where  $M \triangleq \sum_k M_k$ . In this chapter, we focus on the distributed computation of an adaptive  $M$ -inputs  $Q$ -outputs spatial filter  $X \in \mathbb{R}^{M \times Q}$ , that is optimal in

some sense, and structured as

$$X \triangleq \begin{bmatrix} X, \\ \vdots \\ X_K \end{bmatrix}, \quad (4.2)$$

where  $X_k$  is the  $k$ -th block of  $X$ , corresponding to the filter associated with  $\mathbf{y}_k$  in the product  $X^T \mathbf{y}$ .

The original smooth version of DASF [58], applies to filtering problems of the form given by (3.8):

$$\begin{aligned} X^* \in \operatorname{argmin}_{X \in \mathbb{R}^{M \times Q}} \quad & \varphi(X^T \mathbf{y}(t), X^T B) \\ \text{s.t.} \quad & \forall j \in \mathcal{J}_I, \eta_j(X^T \mathbf{y}(t), X^T D_j) \leq 0, \\ & \forall j \in \mathcal{J}_E, \eta_j(X^T \mathbf{y}(t), X^T D_j) = 0. \end{aligned} \quad (4.3)$$

which we denote here by  $\mathbb{P}_S(\mathbf{y}, B, \mathcal{D})$ , where the subscript indicates that we are referring to a smooth problem, and where  $B$  and the  $D_j$  are deterministic matrices known a-priori.

## 4.2.1 Extended scope of the NS-DASF framework

The NS-DASF algorithm deals with filters that are solutions of optimization problems of the form

$$\begin{aligned} X^* \in \operatorname{argmin}_{X \in \mathbb{R}^{M \times Q}} \quad & \varphi(X^T \mathbf{y}(t), X^T B) + \gamma(X^T A) \\ \text{s.t.} \quad & \forall k \in \mathcal{K}, \forall j \in \mathcal{J}_I^k, \eta_j(X_k^T \mathbf{y}_k(t), X_k^T D_{j,k}) \leq 0, \\ & \forall j \in \mathcal{J}_E^k, \eta_j(X_k^T \mathbf{y}_k(t), X_k^T D_{j,k}) = 0 \end{aligned} \quad (4.4)$$

where the matrices  $A$  and  $D_{j,k}$ <sup>1</sup> are, similarly to  $B$ , deterministic matrices,  $\gamma$  is a possibly non-smooth but convex real-valued function encoding some soft-constraints on the filter coefficients, and the  $\eta_j$ 's are smooth functions now describing *per-node* constraints on the filter. Note that the block-separability of the constraints is specific to the non-smooth version of DASF, and is not required for its smooth counterpart. The  $\eta_j$ 's in (4.3) can thus possibly introduce

---

<sup>1</sup>Most problems are usually formulated with  $\gamma(AX)$  rather than  $\gamma(X^T A)$ . This notation was chosen to stay consistent with the  $X^T$  structure of the remaining functions involved in the problem.

multiplicative coupling between the filter coefficients of different nodes, but the  $\eta_j$ 's in (4.4) can only depend on a single block  $X_k$ .

In order to ensure optimality (see Section 4.4), we also require  $\gamma$  to be *per-node* block separable, i.e. there exist functions  $\gamma_k$  such that

$$\gamma(X^T A) = \sum_{k \in \mathcal{K}} \gamma_k(X_k^T A_k) \quad (4.5)$$

This implies that  $A$  must be a block diagonal matrix, whose blocks we denote  $A_k$ . Typical examples of functions satisfying this property are the weighted  $\ell_1$  and  $\ell_{2,1}$  norms, where the later is typically used to introduce group-sparsity in an optimization model. Note that the smooth function  $\varphi$  is not required to be block-separable.

We denote the parametric optimization problem defined in (4.4) as  $\mathbb{P}_{NS}(\mathbf{y}, A, B, D)$ , where  $\mathcal{D}$  denotes the collection (i.e. set) of matrices  $D_{j,k}$ .

The problem structure described by (4.4) now encompasses new problems which were not within the scope of DASF. For example, the problem

$$\begin{aligned} \max_X \quad & \text{Tr}(X^T \mathbb{E}\{\mathbf{y}\mathbf{y}^T\} X) \\ \text{s.t.} \quad & X_k^T \mathbb{E}\{\mathbf{y}_k \mathbf{y}_k^T\} X_k = I_Q \quad \forall k \in \mathcal{K} \end{aligned} \quad (4.6)$$

where  $I_Q$  is the  $Q$ -dimensional identity matrix, is typically referred to as the SUMCORR formulation of generalized canonical correlation analysis (GCCA) [44], [81], [111] and can be cast in the form (4.4). It can be extended by adding  $\gamma(X) = \sum_k \|X_k\|_F$  to the objective function of (4.6) to obtain a sparse version of SUMCORR, which encourages a subset of the channels to be used. As another example, the following problem can be viewed as a sparse Wiener filtering problem [100] with additional power constraints on the per-node filter outputs:

$$\begin{aligned} \min_X \quad & \mathbb{E}\left\{\|X^T \mathbf{y}(t) - \mathbf{d}(t)\|_F^2\right\} + \sum_k \|X_k\|_F \\ \text{s.t.} \quad & \mathbb{E}\left\{\|X_k^T \mathbf{y}_k(t)\|_F^2\right\} \leq P_k \quad \forall k \in \mathcal{K} \end{aligned} \quad (4.7)$$

where  $P_k$  are scalars denoting some power constraint on the filter output, and  $\mathbf{d}(t)$  is a known target signal taking values in  $\mathbb{R}^Q$ . Note that these are only two examples of the many problems that fit in the proposed non-smooth DASF framework.

Our goal is to efficiently track a solution  $X^* \in \mathbb{R}^{M \times Q}$  of (4.4) and the corresponding filter output  $X^{*T} \mathbf{y}(t)$ . Although the optimal filter  $X^*$  is required,

we are typically more interested in the output signals of the spatial filter, i.e., in the  $Q$ -dimensional filtered output  $\mathbf{z}(t) \triangleq X^{*T} \mathbf{y}(t)$  for each sample time  $t$ , which could, for example, correspond to a denoised speech signal that must be available to the network at any time. The DASF algorithm must therefore be such that both the optimal filter and the filtered signal  $\mathbf{z}(t)$  can be computed in a bandwidth efficient manner.

The NS-DASF algorithm assumes that a centralized solver that would be able to find the solution of (4.4) if all data would be known at a fusion center is available. Similarly to the original DASF algorithm, the NS-DASF algorithm will use this solver to compute the solution of lower-dimensional versions of (4.4) that are available at individual nodes. We also assume that computing a solution of this local lower dimensional problem is cheap in comparison to the cost of sharing the data  $\mathbf{y}(t)$ , which motivates the design of a distributed algorithm that can compute  $X^*$  and  $\mathbf{z}(t)$  while also limiting the amount of data that needs to be exchanged between the nodes, by relying on local computations instead. This is a reasonable assumption, as it is well known that the wireless data exchange is typically an energy bottleneck in WSNs [6], [112].

## 4.3 The non-smooth DASF algorithm

In this section, we describe the non-smooth DASF (NS-DASF) algorithm in earnest, i.e. we describe an iterative procedure to solve (4.4) in a distributed fashion, while also tracking the filtered output  $\mathbf{z}(t)$  at each node. The procedure relies on each node sending a compressed view of its local data to a given node, which we call the *updating node*, and whose role is assumed by a different node at each iteration. Based on the compressed data it received, the updating node will update the current estimate of  $X^*$ . For the sake of an easier exposition, we first introduce the algorithm in fully-connected networks before extending the description to arbitrary network topologies at the end of the section.

### 4.3.1 NS-DASF in fully-connected networks

The state of the algorithm at any iteration  $i$  is characterized by the (initially random) current estimate of the optimal solution  $X^i$  and the updating node index  $q^i$  (we use the node index  $q$  without any iteration index to refer to the updating node when the iteration is clear from the context). The algorithmic procedure is essentially the same as the one described in [58] and Chapter 3 for its smooth counterpart, except for the handling of the term  $\gamma$  and the resulting local problems solved by each updating node. However, the convergence analysis (and

in particular the optimality results) is substantially impacted by the addition of this non-smooth term (see Section 4.4). Each iteration is divided into three phases:

**(i) Data aggregation** Each node  $k$  collects  $N$  new samples of  $\mathbf{y}_k$  and sends a block of  $N$   $Q$ -dimensional *compressed* samples of

$$\hat{\mathbf{y}}_k^i \triangleq X_k^{iT} \mathbf{y}_k \quad (4.8)$$

along with

$$\hat{D}_{j,k}^i \triangleq X_k^{iT} D_{j,k} \quad (4.9)$$

$$\hat{A}_k^i \triangleq X_k^{iT} A_k \quad (4.10)$$

$$\hat{B}_k^i \triangleq X_k^{iT} B_k \quad (4.11)$$

to the updating node  $q$ , where  $B_k$  is the block-row of  $B$  associated with  $X_k$ . Note that  $X_k^i$  acts both as the compression matrix and as the current estimate of the optimal filter. Upon reception of the compressed data, the updating node  $q$  constructs the *local* data

$$\tilde{\mathbf{y}}^i = \begin{bmatrix} \mathbf{y}_q \\ \hat{\mathbf{y}}_1^i \\ \vdots \\ \hat{\mathbf{y}}_{q-1}^i \\ \hat{\mathbf{y}}_{q+1}^i \\ \vdots \\ \hat{\mathbf{y}}_K^i \end{bmatrix}, \quad \tilde{B}^i = \begin{bmatrix} B_q \\ \hat{B}_1^i \\ \vdots \\ \hat{B}_{q-1}^i \\ \hat{B}_{q+1}^i \\ \vdots \\ \hat{B}_K^i \end{bmatrix} \quad \text{and} \quad (4.12)$$

$$\tilde{A}^i = \text{BlockDiag}(A_q, \hat{A}_1^i, \dots, \hat{A}_{q-1}^i, \hat{A}_{q+1}^i, \dots, \hat{A}_K^i)$$

adding node  $q$ 's own data to the aggregated data. Similarly to  $\mathcal{D}$ , we denote the collection of the  $\hat{D}_{j,k}^i$ 's and the  $D_{j,q}$  as  $\tilde{\mathcal{D}}^i$ . As was described in Subsection 3.3.1, one can again observe that the “local” data  $\tilde{\mathbf{y}}^i$ ,  $\tilde{A}^i$ ,  $\tilde{B}^i$  and  $\tilde{\mathcal{D}}^i$  live in a subspace of the original data  $\mathbf{y}$ ,  $A$ ,  $B$  and  $\mathcal{D}$ , and can be interpreted as a low-dimensional “view” (i.e. linear combination of the channels/rows) of the original data, where the view of the node's own data is unaltered (i.e. uncompressed).

**(ii) Local solution** In order to update the current estimate of the optimal filter  $X^i$ , the updating node  $q$  computes a solution of the original problem

(4.4), but using the aggregated data (4.12) received in the previous step instead of the original, global, data  $\mathbf{y}$ ,  $A$ ,  $B$ , and  $\mathcal{D}$ . More specifically, it solves  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{A}^i, \tilde{B}^i, \tilde{\mathcal{D}}^i)$ :

$$\begin{aligned} \tilde{X}^* &\in \underset{\tilde{X}}{\operatorname{argmin}} \varphi(\tilde{X}^T \tilde{\mathbf{y}}^i, \tilde{X}^T \tilde{B}^i) + \gamma(\tilde{X}^T \tilde{A}^i) \\ \text{s.t.} \quad \forall k \in \mathcal{K}, \forall j \in \mathcal{J}_I^k, \eta_j(\tilde{X}_k^T \tilde{\mathbf{y}}_k^i, \tilde{X}_k^T \tilde{D}_{j,k}^i) &\leq 0, \\ \forall j \in \mathcal{J}_E^k, \eta_j(\tilde{X}_k^T \tilde{\mathbf{y}}_k^i, \tilde{X}_k^T \tilde{D}_{j,k}^i) &= 0 \end{aligned} \quad (4.13)$$

where we denote  $\tilde{\mathbf{y}}_q^i \triangleq \mathbf{y}_q$ ,  $\tilde{B}_q^i \triangleq B_q$  and  $\tilde{D}_q^i \triangleq D_q$  for the special case of the updating node, and where  $\tilde{X}^*$  is again partitioned as

$$\tilde{X}^* = \begin{bmatrix} \tilde{X}_q^* \\ \tilde{X}_1^* \\ \vdots \\ \tilde{X}_{q-1}^* \\ \tilde{X}_{q+1}^* \\ \vdots \\ \tilde{X}_K^* \end{bmatrix}. \quad (4.14)$$

with each block associated with the corresponding blocks of the local data (4.12). Note that  $\tilde{X}_q$  is an  $M_q \times Q$  matrix, whereas all the other  $\tilde{X}_k$ 's are  $Q \times Q$  matrices. Also note that because the node receives blocks of samples, it will not solve  $\mathbb{P}_{NS}(\tilde{\mathbf{y}}^i, \tilde{A}^i, \tilde{B}^i, \tilde{\mathcal{D}}^i)$  exactly, but an approximation thereof, where the implicit statistics involved in  $\varphi$  and the  $\eta_j$ 's are approximated using sample averages across a batch of  $N$  samples.

As was also the case with the original DASF algorithm, (4.13) shares the structure of the original problem (4.4), it can be solved using the same solver. It should be noted that the dimension of this local problem is much smaller than the original problem, and therefore cheaper to solve, making it amenable to run on devices with limited computing capabilities.

**(iii) Parameters update** Following the partitioning defined in (4.14), the updating node  $q$  updates its own filter according to

$$X_q^{i+1} \leftarrow \tilde{X}_q^* \quad (4.15)$$

and sends the appropriate blocks of  $\tilde{X}^*$  to the other nodes, such that they can update their local filters according to

$$X_k^{i+1} \leftarrow X_k^i \tilde{X}_k^*. \quad (4.16)$$

---

**Algorithm 4.1:** NS-DASF algorithm in fully-connected networks.
 

---

```

begin
   $i \leftarrow 0, q \leftarrow 1$ , Randomly initialize  $X^0$ 
  loop
    for  $k \in \mathcal{K} \setminus \{q\}$  do
      At node  $k$ 
      Collect a new batch of  $N$  samples of  $\mathbf{y}_k(t)$  and send the
      compressed samples  $\tilde{\mathbf{y}}_k^i(t) = X_k^{iT} \mathbf{y}_k(t)$  along with
       $\hat{A}_k^i = X_k^{iT} A_k$ ,  $\hat{B}_k^i = X_k^{iT} \hat{B}_k$  and  $\hat{D}_k^i = X_k^{iT} D_k$  to node  $q$ .
      At node  $q$ 
      Obtain  $\tilde{X}^*$  by solving and selecting any solution of
       $\mathbb{P}_{NS}(\tilde{\mathbf{y}}^i(t), \hat{A}^i, \hat{B}_k^i, \hat{D}^i)$  (see (4.13)).
      Extract the  $\tilde{X}_k^*$ 's from  $\tilde{X}^*$  according to (4.14).
       $X_q^{i+1} \leftarrow \tilde{X}_q^*$ 
      for  $k \in \mathcal{K} \setminus \{q\}$  do
        Send  $\tilde{X}_k^*$  to node  $k$ .
        At node  $k$ 
         $X_k^{i+1} \leftarrow X_k^i \tilde{X}_k^*$ 
       $i \leftarrow i + 1, q \leftarrow (q + 1) \bmod K$ 
  
```

---

The above rule is related to (4.8)–(4.9): At each node, the signal  $\mathbf{y}_k$  of a node can only be manipulated by the updating node “through” the current estimate of the filter  $X_k^i$ , via the parametrization introduced by  $\tilde{X}_k^*$ .

Upon completion of the above steps, the updating node role is passed-on to another node and another iteration begins, using another batch of  $N$  samples. At the end of each iteration, the updating node has access to an estimate of the output filtered signal for the latest  $N$ -samples block:

$$\mathbf{z} \approx \tilde{X}^{*T} \tilde{\mathbf{y}}^i = X^{i+1T} \mathbf{y}. \quad (4.17)$$

The full algorithm description is given by Algorithm 4.1.

### 4.3.2 NS-DASF in arbitrary network topologies

As we have required the constraints to be block separable, we would in practice still need to forward all the compressed data  $X_k^{Ti} \mathbf{y}_k$  in order for the updating

node to be able to evaluate each  $\eta_j(X_k^{T^i} \mathbf{y}_k, X^{T^i} D_k)$ . As this would require expensive data relaying, we require the constraints to depend only on the first or second order statistics of  $X_k^{T^i} \mathbf{y}_k$  in the arbitrary topology case. This allows the nodes to relay the compressed data  $X_k^{T^i} D_k$  along with  $X_k^{T^i} \mathbb{E} \{ \mathbf{y}_k \mathbf{y}_k^T \} X_k$  or  $X_k^{T^i} \mathbb{E} \{ \mathbf{y}_k \}$  instead of a batch of  $N$  samples of  $X_k^{T^i} \mathbf{y}_k$ , which is of much larger dimension than the compressed statistics (which have a dimension of the same order of magnitude as the parameter update matrices  $\tilde{X}$ ). For example, to handle constraints of the form

$$X_k^T \mathbb{E} \{ \mathbf{y}_k \mathbf{y}_k^T \} X_k = I_Q,$$

the nodes would relay (sample-averaged estimates of)  $\hat{R}_k^i \triangleq X_k^{iT} \mathbb{E} \{ \mathbf{y}_k \mathbf{y}_k^T \} X_k^i$  itself and still be able to evaluate the constraints, as then the local constraints in (4.13) would be of the form

$$\tilde{X}_k^T \hat{R}_k^i \tilde{X}_k = I_Q. \quad (4.18)$$

In what follows, we denote the per-node covariance matrix  $\mathbb{E} \{ \mathbf{y}_k \mathbf{y}_k^T \}$  as  $R_k$ , and the per-node first order statistics  $\mathbb{E} \{ \mathbf{y}_k \}$  as  $r_k$ . The procedure in arbitrary topologies is as follows.

**(i) Data aggregation** We build a spanning tree rooted at the updating node and preserving the links with its neighbors using, e.g. Algorithm 2.4. As a reminder, we denote the set of nodes in the subtree containing  $k$  and obtained by removing the link between  $k$  and  $q$  as  $\mathcal{B}_{kq}$  (see Figures 2.2 and 3.3). The compressed data  $\hat{D}_{j,k}^i$  and  $\hat{A}_k$  are forwarded to the updating node. Similarly, each node  $k$  computes and forwards its compressed first and second-order statistics

$$\hat{R}_k^i = X_k^{T^i} R_k X_k^i, \quad (4.19)$$

$$\hat{r}_k^i \triangleq X_k^{T^i} r_k \quad (4.20)$$

based on the latest available batch of samples. The compressed data received by node  $q$  from each of its neighbors  $k$  is

$$\hat{\mathbf{y}}_{kq}^i = \sum_{l \in \mathcal{B}_{kq}} \hat{\mathbf{y}}_l^i, \text{ and} \quad (4.21)$$

$$\hat{B}_{kq}^i = \sum_{l \in \mathcal{B}_{kq}} \hat{B}_l^i.$$

The aggregation procedure for NS-DASF is described by Algorithm 4.2. At the end of the aggregation procedure, the updating node  $q$  has received the

compressed data  $\widehat{\mathbf{y}}_{kq}^i$  of each of its neighbors  $n \in \mathcal{N}_q$ , and the set of compressed matrices  $\widehat{R}_k^i$ ,  $\widehat{r}_k^i$ ,  $\widehat{D}_{j,k}^i$  and  $\widehat{A}_k$  for every node in the network.

The updating then organizes the received data according to

$$\widetilde{\mathbf{y}}^i \triangleq \begin{bmatrix} \mathbf{y}_q \\ \widehat{\mathbf{y}}_{n_1 k}^i \\ \vdots \\ \widehat{\mathbf{y}}_{n_L k}^i \end{bmatrix}, \text{ and } \widetilde{B}^i \triangleq \begin{bmatrix} B_q \\ \widehat{B}_{n_1 k}^i \\ \vdots \\ \widehat{B}_{n_L k}^i \end{bmatrix} \quad (4.22)$$

where  $\{n_1, \dots, n_L\} = \mathcal{N}_q$  are the neighbors of node  $q$ . The matrix  $\widetilde{A}^i$ , is constructed as in the fully connected case (4.12).

**(ii) Local solution** The updating node solves

$$\begin{aligned} \widetilde{X}^* \in \underset{\widetilde{X}}{\operatorname{argmin}} \quad & \varphi(\widetilde{X}^T \widetilde{\mathbf{y}}^i, \widetilde{X}^T \widetilde{B}^i) + \gamma(\widetilde{X}^T \widetilde{A}^i) \\ \text{s.t.} \quad & \forall j \in \mathcal{J}_I^q, \eta_j(\widetilde{X}_q^T \mathbf{y}_q^i, \widetilde{X}_n^T D_{j,q}^i) \leq 0 \\ & \forall j \in \mathcal{J}_E^q, \eta_j(\widetilde{X}_q^T \mathbf{y}_q^i, \widetilde{X}_n^T D_{j,q}^i) = 0, \\ & \forall n \in \mathcal{N}_q, \forall l \in \mathcal{B}_{nq}, \\ & \forall j \in \mathcal{J}_I^l, \eta_j(\widetilde{X}_n^T \widehat{\mathbf{y}}_l^i, \widetilde{X}_n^T \widehat{D}_{j,l}^i) \leq 0, \\ & \forall j \in \mathcal{J}_E^l, \eta_j(\widetilde{X}_n^T \widehat{\mathbf{y}}_l^i, \widetilde{X}_n^T \widehat{D}_{j,l}^i) = 0, \end{aligned} \quad (4.23)$$

where the dependence on the  $\widehat{R}_k^i$  and  $\widehat{r}_k^i$  is implicitly embedded in the  $\eta_j$ .

**(iii) Parameters update** The update procedure is the same as DASF, that is  $\widetilde{X}^*$  is partitioned as

$$\widetilde{X}^* \triangleq \begin{bmatrix} X_q^* \\ \widetilde{X}_{n_1}^* \\ \vdots \\ \widetilde{X}_{n_L}^* \end{bmatrix}, \quad (4.24)$$

and the filters are updated according to

$$X_q^{i+1} \leftarrow X_q^* \quad (4.25a)$$

$$X_k^{i+1} \leftarrow X_k^i \widetilde{X}_{n_l}^* \quad \forall k \in \mathcal{B}_{n_l q}, \forall n_l \in \mathcal{N}_q. \quad (4.25b)$$

The full algorithmic procedure is described by Algorithm 4.3.

## 4.4 Convergence and optimality

In this section, we provide convergence and optimality results for the proposed NS-DASF algorithm. The addition of the non-smooth term and the different technical assumptions under which convergence and optimality are obtained do not allow for a straightforward extension of the convergence proofs of the original DASF algorithm (see [62] and Chapter 3). These proofs are the main contribution of this chapter. In order to keep them accessible, the convergence and optimality analyses are first derived for fully-connected networks in Sections 4.4.2 and 4.4.3, while Section 4.4.4 briefly describes how the results obtained for fully-connected networks can be extended to arbitrary network topologies.

Note that the proofs are still implicitly described under the stationarity and perfect statistical estimation hold (Assumptions 1.1 and 1.2).

### 4.4.1 Notation and proof outline

We define

$$L(X) \triangleq \varphi(X^T \mathbf{y}, X^T B) + \gamma(X^T A) \quad (4.26)$$

and the set  $\mathcal{X}$  as the set of feasible points of (4.4), allowing us to express the original global problem (4.4) as

$$\min_{X \in \mathbb{R}^{M \times Q}} L(X) \quad \text{s.t.} \quad X \in \mathcal{X}. \quad (4.27)$$

We restate here the definition of the local space first introduced in (3.42),

$$\begin{aligned} \mathcal{S}_q(X) \triangleq \text{range } C_q(X) = \\ \text{range } X_1 \times \cdots \times \text{range } X_{q-1} \times \mathbb{R}^{J_{M_q} \times Q} \\ \times \text{range } X_{q+1} \times \cdots \times \text{range } X_K \end{aligned} \quad (4.28)$$

where  $C_q(X)$  was defined in (3.37). As was done in Subsection 3.3.1 for DASF, we can summarize the local problem at any node  $q$  by a single set-valued map

$$\mathcal{F}_q(X) \triangleq \underset{U \in \mathcal{X} \cap \mathcal{S}_q(X)}{\text{argmin}} L(U), \quad (4.29)$$

---

**Algorithm 4.2:** Recursive aggregation procedure in a tree-topology network rooted at node  $q$ .

---

**input :** Parent node  $p_k$  and set of childrens  $\mathcal{C}_k$  for each node  $k \neq q$  (the parent node is the neighbor closest to the updating node  $q$ ,  $\mathcal{C}_k = \emptyset$  for leaf nodes)

**begin**

**At node  $k$**

Collect a new batch of samples of  $\mathbf{y}_k(t)$

Wait for the aggregate compressed signals received from children  $\widehat{\mathbf{y}}_{lk}^i$  along with the sets of compressed matrices  $\{\widehat{D}_{j,m}^i\}_{m \in \mathcal{B}_{lk}}$ ,

$\{\widehat{A}_m^i\}_{m \in \mathcal{B}_{lk}}$ ,  $\{\widehat{R}_m^i\}_{m \in \mathcal{B}_{lk}}$  and  $\{\widehat{r}_m^i\}_{m \in \mathcal{B}_{lk}}$  for  $l \in \mathcal{C}_k$

Send  $\widehat{\mathbf{y}}_{kp_k}^i = \widehat{\mathbf{y}}_k^i + \sum_{l \in \mathcal{C}_k} \widehat{\mathbf{y}}_{lk}^i$  to  $p_k$  and similarly for  $\widehat{B}_{kp_k}^i$

Send  $\{\widehat{D}_{j,k}^i\} \cup_l \{\widehat{D}_{j,m}^i\}_{m \in \mathcal{B}_{lk}}$ ,  $\{\widehat{A}_k^i\} \cup_l \{\widehat{A}_m^i\}_{m \in \mathcal{B}_{lk}}$ ,

$\{\widehat{R}_k^i\} \cup_l \{\widehat{R}_m^i\}_{m \in \mathcal{B}_{lk}}$  and  $\{\widehat{r}_k^i\} \cup_l \{\widehat{r}_m^i\}_{m \in \mathcal{B}_{lk}}$  to  $p_k$

---



---

**Algorithm 4.3:** NS-DASF algorithm in arbitrary networks

---

**begin**

$i \leftarrow 0, q \leftarrow 1$ , Randomly initialize  $X^0$

**loop**

Construct a spanning tree rooted at node  $q$ .

Aggregate the data according to Alg. 4.2.

**At node  $q$**

Obtain  $\widetilde{X}^*$  by solving and selecting any solution of (4.23).

Extract the  $\widetilde{X}_k^*$ 's from  $\widetilde{X}^*$  according to (4.24).

Update the filter of node  $q$  with  $\widetilde{X}_q^*$ .

**for  $n \in \mathcal{N}_q$  do**

Send  $\widetilde{X}_n^*$  to node  $n$ .

**for  $l \in \mathcal{B}_{nq}$  do**

**At node  $l$**

Wait for  $\widetilde{X}_n^*$  and forward it to its children.

$X_l^{i+1} \leftarrow X_l^i \widetilde{X}_n^*$

$i \leftarrow i + 1, q \leftarrow (q + 1) \bmod K$

---

such that the full algorithmic procedure can be summarized by

$$X^{i+1} \in \mathcal{F}_{q^i}(X^i). \quad (4.30)$$

We will show the convergence of the algorithm by studying the properties of the map (4.29). We will first show that the successive application of the map converges to the set of its fixed points, defined as

$$\{X \mid \forall k \in \mathcal{K}, X \in \mathcal{F}_k(X)\}. \quad (4.31)$$

We will then show that the fixed points are solutions, or at least stationary points, of the original problem (4.4). In order to obtain the most general result, we do not assume any particular order for the sequence of updating nodes  $q^i$ , but simply that the number of iterations during which a node does not act as the updating node is bounded above. We register this requirement in the following assumption:

**Assumption 4.1.** *There is some  $S < \infty$  such that it is guaranteed that a node becomes updating node at least every  $S$  iterations.*

#### 4.4.2 Subsequential convergence (in fully-connected networks)

This first section contains a proof that the NS-DASF algorithm converges to the set of its fixed points, i.e., for any distance  $\delta$ , we can find an index  $T$ , such that for any  $i > T$  there is some fixed point  $X_F$  of NS-DASF such that  $\|X_F - X^i\|_F < \delta$ .

This first part of the proof is organized as follows.

- (i) We prove that the procedure results in a monotonic decrease of the objective, and a sequence of feasible points.
- (ii) We prove that the optimal function value of the local problem is an upper semicontinuous function of  $X^i$ .
- (iii) Based on the above two results, we show that if we select a subsequence of  $(X^i)_{i \in \mathbb{N}}$  over which the updating node  $q$  remains the same (i.e.  $q^i = q$  for any  $i$  in the subsequence), the accumulation points of such a sequence are fixed points of  $\mathcal{F}_q$ .
- (iv) We finally show that this must also be true for every node  $q$ .

**Proposition 4.1** (Monotonic decrease). *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence of iterates generated by (4.30). Then, the sequence of objective function values  $(L(X^i))_{i \in \mathbb{N}}$  is monotonically decreasing. In addition, all the points in  $(X^i)_{i \in \mathbb{N}}$  are feasible, i.e.  $X^i \in \mathcal{X}$ .*

*Proof.* As by the definition of the algorithm (4.30)  $X^{i+1}$  is a solution of (4.29), it must be feasible (i.e.  $X^i \in \mathcal{X}$  for any  $i$ ). By the definition (4.28) of  $\mathcal{S}_q$ , we have that  $X \in \mathcal{S}_q(X)$  for every  $X$  (as each block of  $X$  is trivially in its own range). Combining these two facts yields that  $X^i \in \mathcal{X} \cap \mathcal{S}_q(X^i)$ , i.e. it is feasible for the local problems. As  $X^{i+1} \in \mathcal{F}_q(X^i)$ , it must be that  $L(X^{i+1}) \leq L(X)$  for every  $X$  in  $\mathcal{X} \cap \mathcal{S}_q(X^i)$ . Since we just showed that  $X^i$  is also in this set, we find that  $L(X^{i+1}) \leq L(X^i)$ .  $\square$

The remainder of the convergence analysis relies on three technical assumptions, satisfied by a broad class of problems.

**Assumption 4.2** (Continuity, compactness and CCP regularizer). *The function  $L : \mathcal{X} \rightarrow \mathbb{R}$  is continuous and has compact sublevel sets. In addition,  $\gamma$  is closed, convex and proper (CCP).*

The monotonic decrease property from Proposition 4.1 implies that the compactness of all sublevel sets can be relaxed to the compactness of at least the sublevel set of  $L(X^0)$  in  $\mathcal{X}$ , where  $X^0$  is the initialization point of the algorithm.

**Assumption 4.3** (Similar solutions). *The solutions of  $\mathbb{P}_{NS}(\cdot, \cdot, \cdot, \cdot)$  are unique up to a full-rank transformation, i.e. they share the same column space.*

This is trivially satisfied by strictly convex problems, or problems that can be cast as subspace problems, i.e., optimization problems where the solution set is a linear subspace<sup>2</sup>. This assumption ensures that in the case of local problems

<sup>2</sup>Of which principal component analysis is a notable example.

with multiple solutions, all the solutions yield the same local problem at the next iteration, that is the value of  $\mathcal{S}_q(X^{i+1})$  is independent of any particular choice of local solution  $\tilde{X}^*$ . Indeed, one may notice from the definition of  $\mathcal{S}_q$  in (4.28) that for any full-rank matrix  $R$ ,  $\mathcal{S}_q(XR) = \mathcal{S}_q(X)$ .

The next assumption relies on a parametric description of the constraint set. We define the set-valued map

$$\begin{aligned} \mathcal{X}(\mathbf{y}, \mathcal{D}) &\triangleq \{X \in \mathbb{R}^{M \times Q} \mid \forall k \in \mathcal{K}, \\ &\forall j \in \mathcal{J}_I^k, \eta_j(X_k^T \mathbf{y}_k, X_k^T D_{j,k}) \leq 0, \\ &\forall j \in \mathcal{J}_E^k, \eta_j(X_k^T \mathbf{y}_k, X_k^T D_{j,k}) = 0\} \end{aligned} \quad (4.32)$$

which describes the feasible set  $\mathcal{X}$  as a parametric set depending on the data  $\mathbf{y}$  and  $\mathcal{D}$ . We keep using  $\mathcal{X}$  without any argument to denote the feasible set of the global problem (4.4).

**Assumption 4.4** (Feasible set continuity). *The set-valued map  $(\mathbf{y}, \mathcal{D}) \mapsto \mathcal{X}(\mathbf{y}, \mathcal{D})$  is continuous, i.e., it is both outer and inner semicontinuous.*

Set continuity can intuitively be understood as requiring that the minimum distance between any point in the set resulting from an infinitesimal change in the inputs of  $\mathcal{X}$ , i.e., perturbations in the distribution of  $\mathbf{y}$  or in the entries of  $\mathcal{D}$ , and the points of the original set can always be made arbitrarily small by choosing a sufficiently small input perturbation (i.e. the set grows and shrinks “smoothly”). To further illustrate this property, we give some examples of continuous parametric constraint sets.

**Linear constraints** Linear constraints of the form

$$\mathcal{X}(R) = \{X \mid RX = P\} \quad (4.33)$$

vary continuously with  $R$  on trajectories where  $R$  has constant rank. Indeed, the solution of this linear system can be expressed via the pseudo-inverse of  $R$  (used to express the projection on the null-space of  $R$ ), which is continuous on the set of matrices with constant rank [113], [114]. Intuitively, if the range of  $R$  would suddenly lose a dimension, then a new dimension would appear in the solution space of  $RX = P$  (corresponding to  $R$ 's null-space having gained a dimension).

**Orthogonality constraints** Quadratic orthogonality constraints of the form

$$\mathcal{X}(\mathbf{y}) = \{X \mid X^T \mathbb{E} \{\mathbf{y}\mathbf{y}^T\} X = I\} \quad (4.34)$$

are continuous on trajectories where  $\mathbb{E} \{\mathbf{y}\mathbf{y}^T\}$  has full rank. A proof of this fact can be found in Appendix C.1.

**Convex Inequality Constraints** Convex inequality constraints of the form

$$\mathcal{X}(\mathbf{y}, \mathcal{D}) = \{X \mid G(\mathbf{y}, X, \mathcal{D}) \preceq 0\} \quad (4.35)$$

where  $\preceq$  denotes the element-wise  $\leq$ , are continuous if the function  $G$  is convex and continuous in  $X$  for any  $\mathcal{D}$  and distribution of  $\mathbf{y}$ , and if for any  $\mathbf{y}$  and  $\mathcal{D}$  there is a strictly feasible  $X$ . A proof can be found in [68].

Continuity of  $\mathcal{X}$  also ensures the continuity of the map  $\mathcal{R}_q : X \mapsto \mathcal{X} \cap \mathcal{S}_q(X)$ , describing the local constraint set, as stated in the following lemma.

**Lemma 4.1** (Continuity of Local Feasible Sets). *Let  $(\mathbf{y}, \mathcal{D}) \mapsto \mathcal{X}(\mathbf{y}, \mathcal{D})$  be a continuous map. Then the map  $\mathcal{R}_q : X \mapsto \mathcal{X} \cap \mathcal{S}_q(X)$  is also continuous.*

*Proof.* See Appendix C.2. □

**Remark 4.1.** In the two first constraint set examples, the continuity of the constraint set requires the data to have constant rank. Keeping in mind that

$$\mathcal{X} \cap \mathcal{S}_q(X^i) = \mathcal{X}(\tilde{\mathbf{y}}^i, \tilde{\mathcal{D}}^i), \quad (4.36)$$

this constant rank assumption must also hold for the subblocks of  $X$ , as if any  $X_k^i$  loses rank, the compressed  $\tilde{\mathbf{y}}_k$  will have dependent channels. When the continuity of the constraint set is only valid for full-rank data, some additional mechanism must ensure that this condition is met. Slight algorithmic modifications to DASF have been proposed in [62] for the cases where this could not be guaranteed. Similar modifications can be applied to NS-DASF (we refer to [62] for further details).

Note that the combination of Assumptions 4.3 and 4.4 allows us to drop Assumption 3.2, used in the convergence proof of DASF.

We now turn our attention to the actual proof. As the sublevel sets of  $L$  are compact (Assumption 4.2), any sequence  $(X^i)_{i \in \mathbb{N}}$  satisfying the update rule (4.30) has at least one accumulation point  $\bar{X}$  [69, Theorem 3.6]. In other words, there is some index set  $\mathcal{I} \subseteq \mathbb{N}$  such that  $(X^i)_{i \in \mathcal{I}}$  converges to  $\bar{X}$ . Because of Assumption 4.1, we can in particular select  $\mathcal{I}$  such that  $(q^i)_{i \in \mathcal{I}}$  is a constant subsequence with value  $q$ . Our first objective is to show that the accumulation point  $\bar{X}$  associated with such a sequence is a fixed point of the map  $\mathcal{F}_q$ . Before doing so, we first need to show that the value function

$$m_q(X) \triangleq \min_{U \in \mathcal{R}_q(X)} L(U) \quad (4.37)$$

is upper semicontinuous, as stated in the following lemma.

**Lemma 4.2** (Semicontinuity of the Value Function). *Under Assumptions 4.2 and 4.4, the value function  $m_q : \mathcal{X} \rightarrow \mathbb{R}$  is upper semicontinuous.*

*Proof.* See Appendix C.3. □

We can now relate the accumulation points of  $(X^i)_{i \in \mathbb{N}}$  to the fixed points of the maps  $\mathcal{F}_q$ .

**Lemma 4.3.** *Let  $(X^i)_{i \in \mathbb{N}}$  and  $(q^i)_{i \in \mathbb{N}}$  be sequences related by (4.30) and let  $\mathcal{I} \subseteq \mathbb{N}$  be such that the subsequence  $(X^i, q^i)_{i \in \mathcal{I}}$  converges to  $(\bar{X}, q)$ . Then under Assumptions 4.2 and 4.4,  $\bar{X}$  is a fixed point of  $\mathcal{F}_q$ .*

*Proof.* Since  $q^i$  is part of the finite set  $\mathcal{K}$ , its convergence implies that it eventually becomes constant. We can therefore proceed as if it was a constant sequence. By (4.30) and (4.37), we have  $L(X^{i+1}) = m_q(X^i)$ . As  $L$  is continuous with compact sublevel sets, it attains its minimum value in  $\mathcal{X}$  (extreme value theorem) [70], and the sequence  $L(X^{i+1})$  is therefore bounded below by  $L(X^*)$  and, as a consequence of its monotonic decrease (Proposition 4.1), it must therefore converge to some value  $\bar{L}$  [69]. From the continuity of  $L$ , it must be that  $\bar{L} = L(\bar{X})$ . Since  $L(X^{i+1})$  converges

to  $L(\bar{X})$ , and since  $m_q(X^i) = L(X^{i+1})$ , it must hold that  $m_q(X^i)$  also converges to  $L(\bar{X})$ . From Lemma 4.2 (and in particular (C.6)), it must be that

$$L(\bar{X}) \leq m_q(\bar{X}). \quad (4.38)$$

Since  $\bar{X}$  is the accumulation point of a sequence  $(X^i)_i$  living in the closed<sup>a</sup> set  $\mathcal{X}$ , it must be that  $\bar{X} \in \mathcal{X}$  (by the definition of closedness), and since  $\bar{X} \in \mathcal{S}_q(\bar{X})$  by definition (see (4.28)), we conclude that  $\bar{X} \in \mathcal{R}_q(\bar{X}) = \mathcal{X} \cap \mathcal{S}_q(\bar{X})$ . Hence, we have by definition of the value function,

$$m_q(\bar{X}) \leq L(\bar{X}). \quad (4.39)$$

Combining this fact with (4.38), we conclude that

$$m_q(\bar{X}) = L(\bar{X}), \quad (4.40)$$

and hence  $\bar{X} \in \mathcal{F}_q(\bar{X}) = \operatorname{argmin}_{U \in \mathcal{R}_q(\bar{X})} L(U)$ .  $\square$

<sup>a</sup>The closedness of  $\mathcal{X}$  follows from the continuity of the  $\eta_j$ .

It now remains to show that accumulation points of the full sequence  $(X^i)_{i \in \mathbb{N}}$  are fixed points of the map  $\mathcal{F}_q$  for any  $q$ .

**Proposition 4.2** (Subsequential convergence). *Let  $(X^i)_{i \in \mathbb{N}}$  be any sequence generated by (4.30). Then under Assumptions 4.1–4.4, any accumulation point of  $(X^i)_{i \in \mathbb{N}}$  is a fixed point of the map  $\mathcal{F}_q$  for any  $q \in \mathcal{K}$ .*

*Proof.* Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence satisfying (4.30), by Assumption 4.2, there must be an index set  $\mathcal{I} \subseteq \mathbb{N}$  such that the subsequence  $(X^i, X^{i+1}, q^i, q^{i+1})_{i \in \mathcal{I}}$  converges to some  $(\bar{X}, \bar{X}^{+1}, q, q')$  (we also used the fact that the cartesian product of compact sets is compact [70]). We first prove that  $\bar{X}^{+1} \in \mathcal{F}_q(\bar{X})$ . From Assumption 4.4 and Lemma 4.1, we have<sup>a</sup> that

$$\lim_{i \in \mathcal{I} \rightarrow \infty} X^{i+1} = \bar{X}^{+1} \in \mathcal{R}_q(\bar{X}) = \mathcal{R}_q \left( \lim_{i \in \mathcal{I} \rightarrow \infty} X^i \right). \quad (4.41)$$

From the continuity of  $L$  (Assumption 4.2), we know that

$$L(\bar{X}^{+1}) = L(\bar{X}) = \lim_{i \rightarrow \infty} L(X^i), \quad (4.42)$$

and by Lemma 4.3 (see (4.40))

$$L(\bar{X}^{+1}) = L(\bar{X}) = m_q(\bar{X}),$$

we find that  $\bar{X}^{+1} \in \mathcal{F}_q(\bar{X})$  (as  $\bar{X}^{+1}$  is in the local feasible set and minimizes the local objective). We will use this result to show that  $\bar{X}^{+1}$  is also a fixed point of  $\mathcal{F}_q$  (we already know that  $\bar{X}$  is a fixed point from Lemma 3). From Assumption 4.3 and since both  $\bar{X}^{+1}, \bar{X} \in \mathcal{F}_q(\bar{X})$ , we have  $\mathcal{S}_k(\bar{X}) = \mathcal{S}_k(\bar{X}^{+1})$  and thus

$$\bar{X}^{+1} \in \mathcal{F}_k(\bar{X}) = \mathcal{F}_k(\bar{X}^{+1}) \quad (4.43)$$

for any  $k$ , because, by Assumption 4.3, they share the same column space. As any pair of successive accumulation points  $(\bar{X}, \bar{X}^{+1})$  share the same column space, we can inductively deduce that any sequence of accumulation points  $(\bar{X}, \dots, \bar{X}^{+Q})$  (for the right selection of  $\mathcal{I}$ ) share the same column space. In addition, by (4.42), we also have

$$L(\bar{X}) = L(\bar{X}^{+1}) = \dots = L(\bar{X}^{+Q}). \quad (4.44)$$

Because of Assumption 4.1, we can find some  $Q < \infty$  such that the sequence of accumulation points associated with  $(X^i, \dots, X^{i+Q})_{i \in \mathcal{I}}$  is such that all nodes are selected at least once between iterations. For any  $q$  we can therefore find some  $\bar{X}^{+a}$ , with  $a \leq Q$ , such that  $\mathcal{S}_q(\bar{X}) = \mathcal{S}_q(\bar{X}^{+a})$  and hence because  $L(\bar{X}) = L(\bar{X}^{+a})$ ,

$$\bar{X} \in \mathcal{F}_q(\bar{X}), \quad (4.45)$$

where  $q$  was selected arbitrarily in  $\mathcal{K}$ . We therefore find that  $\bar{X}$  is a fixed point for any  $q$  and therefore a fixed point of the full algorithm.  $\square$

<sup>a</sup>This is the definition of set outer semicontinuity.

### 4.4.3 Optimality of fixed points (in fully-connected networks)

It now remains to show that the aforementioned fixed points the algorithm converges to are optimal in some sense. We first describe the optimality result for the case of fully-connected networks, before describing it for arbitrary network topologies. For ease of notation let us define the functions

$$f(X) \triangleq \varphi(X^T \mathbf{y}(t), X^T B) \text{ and } g(X) \triangleq \gamma(X^T A). \quad (4.46)$$

In the non-smooth case, a stationary point  $X^\circ$  is defined as a point satisfying the following inclusion<sup>3</sup>:

$$0 \in \nabla f(X^\circ) + \partial g(X^\circ) + N_{\mathcal{X}}(X^\circ) \quad (4.47)$$

where  $\nabla f(X^\circ)$  is the gradient of  $f$  at  $X^\circ$ ,  $\partial g(X^\circ)$  is the set of subgradients of  $g$  at  $X^\circ$  and  $N_{\mathcal{X}}(X^\circ)$  is the normal cone to  $\mathcal{X}$  at  $X^\circ$  [68], [72]. The sum of sets must be understood as a Minkowski sum<sup>4</sup>. Intuitively, the normal cone to a set at a particular point can be understood as the set of directions having no component “pointing inwards” the set (it therefore only contains 0 at a point in the set’s interior). The set of subgradient can be roughly<sup>5</sup> understood as the set of slopes of all the linear under-approximators tangent to the graph of  $g$  at a particular point. This inclusion translates the fact that at a stationary point, any direction of improvement must exit the constraint set, and is known as Fermat’s rule. In the smooth case, (4.47) is strictly equivalent to the usual KKT conditions [73], [74].

In order to prove that the fixed points of the algorithm satisfy (4.47), we will show that under the proper assumption (or qualification, in optimization parlance), the local optimality at each node (defined by (4.29)) implies global stationarity (i.e. being a stationary point of the centralized problem (4.4)).

We first treat the case of fully-connected networks before moving on to arbitrary topologies.

#### Fully-connected networks

We start with a technical lemma that ensures that the normal cone of  $\mathcal{R}_q(X)$  is a subset of the sum of the normal cones of  $\mathcal{X}$  and  $\mathcal{S}_q(X)$ .

<sup>3</sup>A fairly complete and self-contained introduction to the notions of stationarity in the non-smooth case is available in [115]. See also Subsection 1.2.2.

<sup>4</sup> $\mathcal{A} + \mathcal{B} \triangleq \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\}$ .

<sup>5</sup>See section 1.2.2 for a formal definition.

**Lemma 4.4.** *Let  $X$  satisfy the following constraint qualification*

$$\forall U \in N_{\mathcal{X}}(X), \quad C_k(X)^T U = 0 \Rightarrow U = 0 \quad \forall k \quad (4.48)$$

*Then for every  $k$ ,*

$$N_{\mathcal{R}_k(X)}(X) \subseteq N_{\mathcal{X}}(X) + N_{S_k(X)}(X).$$

*Proof.* See Appendix C.4. □

Later on, we will provide a more interpretable sufficient condition for the qualification (4.48), see Proposition 4.5.

**Proposition 4.3** (Optimality in Fully-Connected Networks). *Under Assumption 4.2, fixed points  $X^\circ$  of the algorithm (4.30) satisfying the qualification (4.48) are stationary points of problem (4.4).*

*Proof.* Since  $X^\circ$  is a fixed point, it is a solution of the local problem

$$\min_{X \in \mathcal{R}_q(X^\circ)} f(X) + g(X) \quad (4.49)$$

at any node  $q$ . This local optimality of  $X^\circ$  along with the fact that  $\gamma$  and hence  $g$  is CCP (Assumption 4.2) implies that [72, Theorem 4.75] for every  $k$

$$0 \in \nabla f(X^\circ) + \partial g(X^\circ) + N_{\mathcal{R}_k(X^\circ)}(X^\circ). \quad (4.50)$$

From (4.50) and Lemma 4.4, we find that the following must also hold:

$$0 \in \nabla f(X^\circ) + \partial g(X^\circ) + N_{\mathcal{X}}(X^\circ) + N_{S_k(X^\circ)}(X^\circ). \quad (4.51)$$

From [72, Prop. 4.44 & 6.43], the block separability of  $g$  and  $\mathcal{X}$  implies that

$$\begin{aligned} \partial g(X^\circ) + N_{\mathcal{X}}(X^\circ) &= (\partial g_1(X_1^\circ) + N_{\mathcal{X}_1}(X_1^\circ)) \times \\ &\quad \cdots \times (\partial g_K(X_K^\circ) + N_{\mathcal{X}_K}(X_K^\circ)), \end{aligned} \quad (4.52)$$

where  $\mathcal{X}_k$  refers to the constraint-set related to a single block of  $X$ . Therefore, as  $N_{\mathcal{S}_k(X^\circ)}$  also has a block structure, and in particular, the block  $[N_{\mathcal{S}_k(X^\circ)}]_k = \{0\}$  (as the updating block is unconstrained, any direction is feasible, i.e.  $[\mathcal{S}_k]_k = \mathbb{R}^{M_k \times Q}$ ), we have that for any  $k$ ,

$$-\nabla_k f(X^\circ) \in \partial g_k(X_k^\circ) + N_{\mathcal{X}_k}(X_k^\circ) \quad (4.53)$$

and therefore

$$-\nabla f(X^\circ) \in \partial g(X^\circ) + N_{\mathcal{X}}(X^\circ), \quad (4.54)$$

which is equivalent to (4.47), hence completing the proof.  $\square$

We defer the statement of our main result for fully connected networks to Section 4.4.6, after giving a more interpretable condition for the qualification of Lemma 4.4 to hold.

#### 4.4.4 Extention to arbitrary network topologies

In this subsection, we explain how all previous results, which were derived only for the case of a fully-connected network, can be extended to arbitrary network topologies. It only requires a minor change in the description of the in-network fusion process. In the fully-connected case, this fusion process is mathematically described by the matrix  $C_q$ . For the case of arbitrary

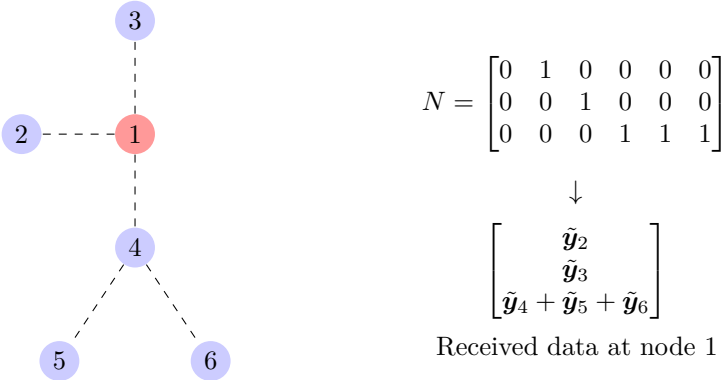


Figure 4.2: An example aggregation matrix and received data for a tree rooted at node 1.

topologies, we have to modify this fusion matrix to describe the per-branch fusion within each of the trees constructed around each updating node  $q$ . Indeed, instead of receiving the compressed data  $\widehat{\mathbf{y}}_k^i = X_k^{iT} \mathbf{y}_k$  from all other nodes, the updating node  $q$  will receive the linear combination  $T_q C_q(X^i)^T \mathbf{y}$  where  $T_q \in \mathbb{R}^{(M_q + |\mathcal{N}_q|Q) \times (K-1)Q + M_q}$  has the structure

$$T_q = \text{BlockDiag}(I_{M_q}, N \otimes I_Q) \tag{4.55}$$

where  $\otimes$  denotes the Kronecker product, and  $N$  is a topology-dependent matrix encoding the aggregation procedure. More specifically, we define some tree graph with the updating node as the root. The matrix  $N$  groups the nodes based on the branches  $\mathcal{B}_{n_q}$  for  $n \in \mathcal{N}_q$  (see Figure 4.2 for an illustrative example). It has as many rows as the updating node has neighbors, and  $K - 1$  columns corresponding to all nodes with the column for node  $q$  omitted. The element  $N_{i,j}$  is 1 if node  $j$  is in the  $i$ -th branch, and 0 otherwise. The received data  $T_q C_q(X^i)^T \mathbf{y}^i$  is then a block matrix, where the first block of  $M_q$  rows correspond to the updating node's uncompressed data  $\mathbf{y}_q$ , and where each subsequent block of  $Q$  rows can be interpreted as the compressed data of a full branch (instead of a single node in the fully-connected setting). Note that in the case of fully-connected networks,  $T_q = I$  for any  $q$ .

### Convergence

The results of Section 4.4.2 can be straightforwardly extended to arbitrary network topologies by replacing  $C_q(X)$  by  $C_q(X)T_q^T$  everywhere it appears in those sections, and thus update the definition of the local range constraint set  $\mathcal{S}_q$  in (4.28) with

$$\mathcal{S}'_q(X) \triangleq \text{range } C_q(X)T_q^T. \tag{4.56}$$

$C_q(X)T_q^T$  shares with  $C_q$  the two properties that matter to the proof: its continuity and invariance to full-rank transformations of  $X$ . The continuity ensures that the continuity of the local constraint set  $\mathcal{X} \cap \mathcal{S}'_q(\cdot)$  in Lemma 4.1 still applies to arbitrary topologies, and the invariance to full-rank transformation is a key property used in the proof of Proposition 4.2. The proof of convergence is otherwise identical to the fully-connected case.

### Optimality

Proposition 4.3 must be slightly modified to ensure that the proper qualification is met at each of the local problems, as described hereafter.

**Proposition 4.4** (Optimality in Arbitrary Network Topologies). *Fixed points  $X^\circ$  of the procedure (4.30) satisfying for every  $k$  the qualification*

$$\forall U \in N_{\mathcal{X}}(X^\circ), \quad T_k C_k(X^\circ)^T U = 0 \Rightarrow U = 0 \quad (4.57)$$

*are stationary points of the problem (4.4).*

*Proof.* Let  $\mathcal{R}'_k(X) \triangleq \mathcal{X}(D) \cap \mathcal{S}'_k(X)$ . The qualification is sufficient to ensure the inclusion [68]:

$$N_{\mathcal{R}'_k(X)}(X) \subseteq N_{\mathcal{X}}(X) + N_{\mathcal{S}'_k(X)}(X). \quad (4.58)$$

The reasoning for the above inclusion is the same as Lemma 4.4, but where we consider  $T_k C_k(X)^T$  instead of  $C_k(X)^T$ . The rest of the proof is identical to the proof of Proposition 4.3 with  $\mathcal{R}_k$  replaced with  $\mathcal{R}'_k$  and  $\mathcal{S}_k$  replaced with  $\mathcal{S}'_k$ .  $\square$

Note that Proposition 4.3 is a special case of Proposition 4.4 where  $T_k = I$ .

#### 4.4.5 Constraint qualifications and an upper bound on the number of constraints

The qualifications in Propositions 4.3 and 4.4 simply ensure that the local solutions satisfy the optimality conditions of the local problems (see [72] for examples failing to meet the qualification). The following proposition gives a sufficient condition akin to the familiar linear independence constraint qualifications (LICQ) for the qualification to hold in the case of equality and inequality constraints in arbitrary topology networks. In what follows, we define  $\vartheta_j^k : X_k \mapsto \eta_j(X_k^T \mathbf{y}_k, X_k^T D_k)$ .

**Proposition 4.5** (Constraint Qualification). *Let  $\mathcal{A}^k(X) \triangleq \{j \in \mathcal{J}_I^k \mid \vartheta_j^k(X_k) = 0\}$  denote the set of active inequality constraints for node  $k$  and*

$$\mathcal{D}_{nk} \triangleq \{X_l^T \nabla \vartheta_j^l(X_l) \mid j \in \mathcal{J}_E^l \cup \mathcal{A}^l(X), l \in \mathcal{B}_{nk}\}.$$

*Then if the elements of  $\mathcal{D}_{nk}$  are linearly independent matrices for every pair of neighboring nodes  $n, k$ , then it holds that*

$$\forall U \in N_{\mathcal{X}}(X), \quad T_k C_k(X)^T U = 0 \Rightarrow U = 0 \quad (4.59)$$

*for any  $k$ .*

*Proof.* See Appendix C.5. □

The independence of the elements of  $\mathcal{D}_{nk}$  implies that the “compressed” gradients  $X_l^T \nabla \vartheta_j^l(X_l)$  of all the constraints, associated with all the nodes of a branch  $\mathcal{B}_{nq}$  (for some updating node  $q$ ), must be independent. In order to ensure that  $\mathcal{D}_{nq}$  has independent elements, all the nodes in a single branch can (in total) have at most  $Q^2$  constraints (the dimension of the compressed gradients). In the case of fully-connected networks, each branch contains a single node, and the bound can be relaxed to a maximum of  $Q^2$  constraints per node. The difference with the similar bound found for the original smooth DASF algorithm [58], [62] can be attributed to the imposed separability of the constraints in the non-smooth problem (4.4).

#### 4.4.6 Main result

We finally summarize the convergence and optimality of NS-DASF with the following theorem:

**Theorem 4.1** (Convergence and optimality). *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by Algorithm 4.1 or 4.3. Assume that the qualification described in Proposition 4.5 holds at the accumulation points of  $(X^i)_{i \in \mathbb{N}}$ , along with Assumptions 4.1–4.4. Then  $(X^i)_{i \in \mathbb{N}}$  converges to the set of stationary points of problem (4.4).*

*Proof.* By virtue of Proposition 4.2 (which can be extended to arbitrary topologies by considering  $C_q(X)T_q^T$  instead of  $C_q(X)$ ), the accumulation points of  $(X^i)_{i \in \mathbb{N}}$  are fixed points of the algorithm, and the sequence therefore converges to the set of fixed points of the algorithm (see Lemma F.1 for the proof that a sequence converges to the set of its accumulation points). Finally, Proposition 4.5 ensures that the qualification (4.48) is met, which in turns ensures by Proposition 4.3 the stationarity of fixed points.  $\square$

Although we only guarantee convergence to a set, convergence to a single point can be guaranteed in several common scenarios, some of which can be enforced by the proper modification of the procedure, as described hereafter.

**The solution of the local problems is uniquely defined** In this case, the output of the set-valued maps  $\mathcal{F}_q$  is a singleton, and the map is therefore a function in the usual sense.

**Proposition 4.6.** *Let the solution of the local problems (4.29) be uniquely defined. Then under Assumptions 4.2 and 4.4,*

$$\lim_{i \rightarrow \infty} \|X^i - X^{i+1}\|_F = 0. \quad (4.60)$$

*Proof.* In the proof of Proposition 4.2, we have shown that for any subsequence generated by the procedure, there is some index set  $\mathcal{I}$  such that  $(X^i, X^{i+1}, q^i)_{i \in \mathcal{I}}$  converges to some  $(\bar{X}, \bar{X}^{+1}, q)$ , and such that both  $\bar{X}, \bar{X}^{+1} \in \mathcal{F}_q$ . But because  $\mathcal{F}_q$  is single-valued, it must be that  $\bar{X} = \bar{X}^{+1}$ . Thus

$$\lim_{i \in \mathcal{I} \rightarrow \infty} \|X^i - X^{i+1}\|_F = 0. \quad (4.61)$$

As this is true for any subsequence, 0 is the sole accumulation point of  $(\|X^i - X^{i+1}\|_F)_{i \in \mathbb{N}}$ , and by virtue of Lemma F.2, it must converge to 0.  $\square$

It seems reasonable that, in the context of our target problems (4.4), the regularizer will most likely favor a particular solution of the local problems to be selected (which is, in the end, the purpose of a regularizer). If that would not be the case, once the algorithm is sufficiently stable in objective values, the weight of the regularization parameter can be progressively increased across iterations until a single solution is obtained.

**The solution set of the problem is continuous and a selection step is added**

We fall back in the scenario described in the original dasf algorithm (Algorithm 3.2, and the convergence of residuals is guaranteed. Note that the continuity assumption (Assumption 3.2) is much stronger than the continuity of the constraint set (Assumption 4.4) considered here.

In both those cases, it can be shown that if the problem has a finite number of stationary points, then the algorithm always converges to a single point as a consequence of Lemma F.3. Note that even in the case where the algorithm would have an infinite number of stationary points, the convergence of  $\|X^i - X^{i+1}\|_F$  ensures that the output filter is stable, and it is therefore not a concern in practice.

## 4.5 Numerical experiments

This section provides some numerical results supporting the theoretical claims of Section 4.4. We consider the problem of computing a sparse Wiener filter. The target filter is the solution of

$$x^* \triangleq \underset{x}{\operatorname{argmin}} \mathbb{E} \left\{ \|x^T \mathbf{y}(t) - \mathbf{d}(t)\|^2 \right\} + \lambda \sum_k \|x_k\|_2, \tag{4.62}$$

where  $\mathbf{d}(t)$  is a single channel target signal known by every node. Note that the per-node regularization term can be written as  $\|x_k^T A_k\|_2$  with  $A_k = I$ , such that it fits (4.4). We use a lowercase  $x$  to emphasize that the filter has a single output channel (i.e.  $Q = 1$ ).  $\lambda$  acts here as a meta-parameter allowing to tune the trade-off between solution accuracy and bandwidth (more nodes will eventually become inactive with a higher value of  $\lambda$ ). For the following experiments, we

considered a fully-connected network where  $M = K = 10$ . Each entry of a sample of  $\mathbf{y}(t)$  is sampled from a zero-mean unit-variance gaussian distribution.  $\mathbf{d}(t)$  is constructed as

$$\mathbf{d}(t) = a^T \mathbf{y}(t) + 10\mathbf{n}(t), \quad (4.63)$$

where the entries in  $\mathbf{n}(t)$  and  $a$  are also sampled from a zero-mean unit-variance gaussian distribution.

The local problems associated with (4.62) will be of the form

$$\min_{\tilde{x}} \mathbb{E} \left\{ \left\| \tilde{x}^T \tilde{\mathbf{y}}(t) - \mathbf{d}(t) \right\|_F^2 \right\} + \lambda \sum_k \left\| \tilde{x}_k^T \tilde{A}_k \right\|_2. \quad (4.64)$$

As  $x \mapsto \sum_k \left\| x_k^T A_k \right\|_2$  is not proximable, using proximal gradient descent [108] for solving the local problems would be very inefficient. We use the Chambolle-Pock [116] algorithm instead, as it was designed specifically for problems such as (4.64). See Appendix D for details on the use of Chambolle-Pock algorithm for solving problem (4.64).

Figure 4.3 depicts the typical convergence behavior of NS-DASF applied to (4.62), i.e. error convergence when starting from a random filter. In order to better observe the adaptivity of the algorithm, the solution  $x^*$  was randomly changed at iteration 40. The top plot depicts the relative excess cost computed as

$$1 - \frac{L(x^i)}{L(x^*)}, \quad (4.65)$$

and the bottom plot depicts the hamming distance between the set of active nodes, that is

$$d_H(x_A, x_B) \triangleq H(|x_A| > 0, |x_B| > 0), \quad (4.66)$$

where  $H$  denotes the actual hamming distance,  $|\cdot|$  and  $>$  are the element-wise absolute value and “greater than” operators.

NS-DASF applied to (4.62) appears to exhibit a linear convergence rate. A first dip in excess cost can be observed after the tenth iteration, as it is only then that each node has had the opportunity to freely update its corresponding block of  $x^i$ . The set of active channels is correctly identified after at most two full rounds of iterations (i.e. 20 iterations). Figure 4.3 depicts this same behavior when changing the solution to a new random point.

Figure 4.4 depicts the adaptive property of NS-DASF in a scenario where the solution changes over time for different rates of change. At each iteration, we apply a perturbation with zero-mean gaussian entries  $e^i$  to the ground-truth solution  $x^{i*}$ , with the ratio  $\left\| e^i \right\|_F / \left\| x^{i*} \right\|_F$  kept constant across iterations (this

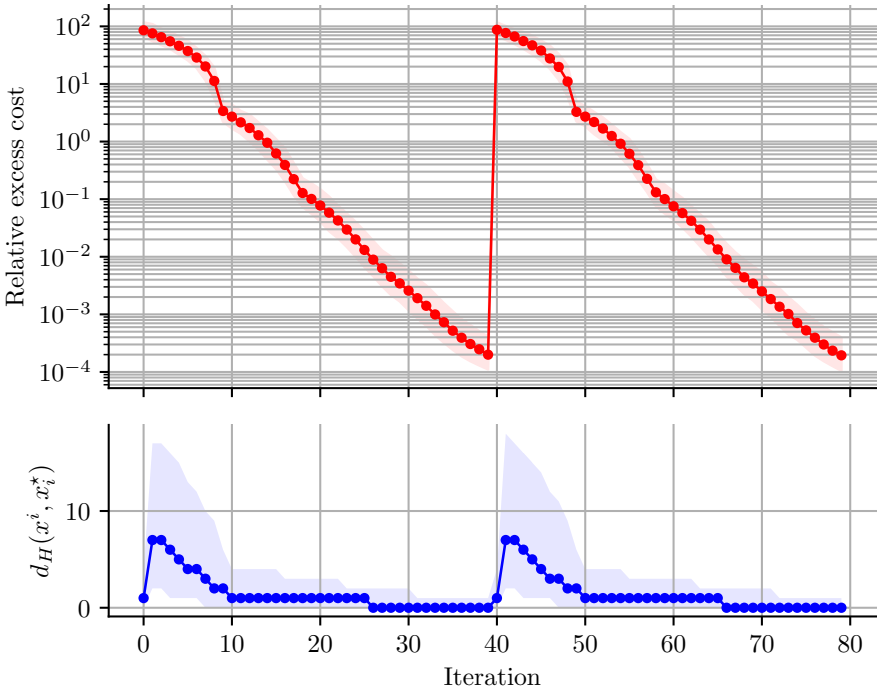


Figure 4.3: Transient behavior of NS-DASF. The optimal solution changes at iteration 40. Solid curve depicts the median performance, the shaded area depicts the 5%-95% percentile region and were computed over 10000 Monte-Carlo runs.

ratio reflects the amount of change in  $x^*$  across iterations, i.e., a higher ratio is more challenging). The residual excess cost is defined as

$$\lim_{i \rightarrow \infty} 1 - \frac{L(x^i)}{L(x^{i^*})}, \tag{4.67}$$

and is estimated in the simulation by computing the relative excess cost at iteration  $i = 200$ . Figure 4.4 shows a mostly linear relationship between the rate of change of the solution and the residual error. This indicates that the tracking error can be decreased by reducing the time between iterations, either by iterating multiple times on the same batch of samples using the technique proposed in [117], or by collecting more samples, such that iterations can be performed more frequently, without reducing the number of samples per batch.

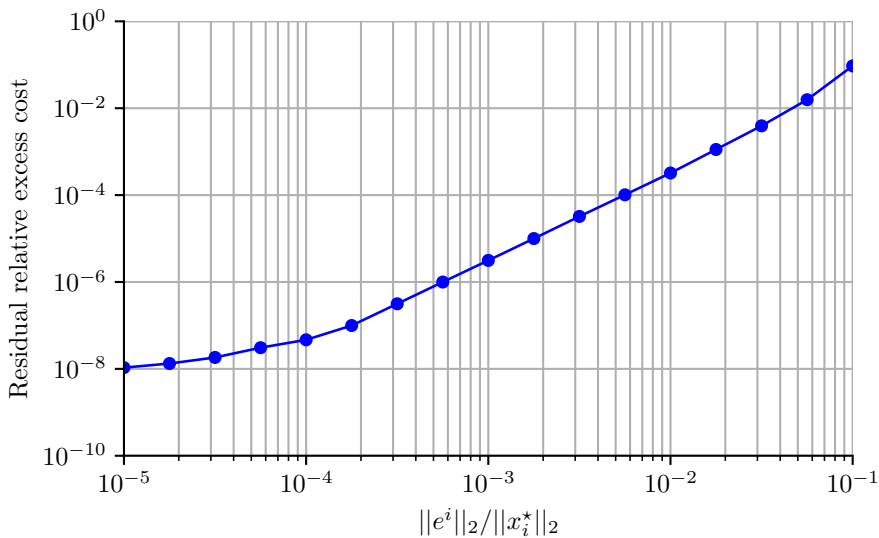


Figure 4.4: Tracking performance of NS-DASF. Each data point corresponds to the mean relative excess cost at iteration 200 of 100 hundred Monte-Carlo runs.

## 4.6 Discussion

In this chapter, we have described an extension of the DASF algorithm for a particular class of non-smooth spatial filtering problems. We have provided theoretical results ensuring the convergence and optimality of this NS-DASF algorithm. Furthermore, we have replaced the most constraining assumption required for the convergence of DASF by two milder and easier to check assumptions. Still, there are still cases where these assumptions could possibly not hold, as in the contrived examples given in Subsection 3.3.5. We address this fact in the next chapter by dropping both assumptions altogether.

Using Monte-Carlo simulations, we have demonstrated the algorithm's transient and stationary behavior when applied to a sparse form of the multichannel Wiener filtering problem. In particular, this indicates that NS-DASF can be used to perform channel or node selection alongside a given filtering task, allowing energy savings compared to DASF used without channel selection, or with an apriori ad-hoc channel selection procedure (such as the one described in Chapter 2), by omitting the transmission from nodes with 0-norm filters. Note that the actual energy savings would in part depend on the possible increase in computational power associated with the use of a local solver able to handle

non-smooth problems. However, we do not expect a significant increase compared to the smooth case, as efficient first-order solvers with similar computational requirements as their smooth counterparts are typically available for such problems [108], [118].



## Chapter 5

---

# DASF with finite-time solvers

---

This chapter is based on

- \* **C. Hovine** and A. Bertrand, “Distributed adaptive spatial filtering with inexact local solvers”, *arXiv preprint arXiv:2405.03277*, 2024 (in first round of review for IEEE’s Transactions on Signal Processing)

## Chapter abstract

The Distributed Adaptive Signal Fusion (DASF) framework is a meta-algorithm for computing data-driven spatial filters in a distributed sensing platform with limited bandwidth and computational resources, such as a wireless sensor network. The convergence and optimality of the DASF algorithm has been extensively studied under the assumption that an exact, but possibly impractical solver for the local optimization problem at each updating node is available. In this work, we provide convergence and optimality results for the DASF framework when used with an inexact, finite-time solver such as (proximal) gradient descent or Newton's method. We provide sufficient conditions that the solver should satisfy in order to guarantee convergence of the resulting algorithm, and a lower bound for the convergence rate. We also provide numerical simulations to validate these theoretical results.

## 5.1 Introduction

The distributed adaptive signal fusion (DASF) framework [58], introduced in Chapter 3 and later extended in Chapter 4, provides a scalable and bandwidth-efficient algorithm that adaptively computes the outputs of data-driven spatial filters satisfying some optimality criterion, in a WSN or other distributed setting with constrained resources. Given an existing solver for the particular optimization problem associated with the filter of interest, the DASF algorithm can adaptively compute the desired filter along with the filtered signal by relying on the exchange of low-dimensional compressed samples between the nodes, and the repeated computation of the solution of a lower-dimensional version of the original centralized optimization problem at each node. The DASF framework is mostly problem-agnostic, and most traditional linear spatial filtering or estimation problems can be solved by plugging-in the appropriate (and unmodified) solver for the corresponding centralized problem. However, the original DASF framework was shown to converge under the assumption that the solver produces an (near-)exact solution every time it is called. In practice though, an exact solver might not be available, either due to the nature of the problem (e.g. non-convexity), or simply because obtaining an exact solution would be too expensive or time consuming, hindering the adaptivity of the algorithm. In addition, the convergence of DASF relies on hard-to-check assumptions regarding the parametric continuity of the problem. In this chapter, we show that an exact solution is not required for optimality, and that a few iterations of an iterative solver are in practice sufficient to ensure convergence to an “interesting” filter (i.e. satisfying some relaxed optimality condition such as mere stationarity with regards to the optimality criterion). We are also able to eliminate some of the original convergence assumptions, making the algorithm applicable to a broader set of problems.

This chapter is organized as follows. Section 5.2 describes the general objective, as well as the family of problems covered by DASF. Section 5.3 describes how DASF can be modified to work with an inexact iterative solver, rather than an exact one. This section describes our main contribution, including a proof of the convergence of DASF used with such a solver. Section 5.4 offers a validation of our theoretical results via numerical simulations. We conclude with a brief discussion in Section 5.5.

## 5.2 Problem statement

For convenience we reintroduce the general problem setting of DASF hereafter.

We consider a WSN consisting of  $K$  nodes, each sensing an  $M_k$ -dimensional stochastic signal  $\mathbf{y}_k(t)$ , where  $t$  denotes a sample index (usually related to a time index). We denote the  $M$ -dimensional network-wide sensor signal

$$\mathbf{y}(t) \triangleq \begin{bmatrix} \mathbf{y}_1(t), \\ \vdots \\ \mathbf{y}_K(t) \end{bmatrix} \quad (5.1)$$

with values in  $\mathbb{R}^M$ , with  $M = \sum_k M_k$ . We assume that  $\mathbf{y}$  is short-term stationary and ergodic, such that its statistics can be computed over short-term sample batches. The DASF framework assumes that the nodes in the network collaborate to compute an optimal linear spatial filter  $X \in \mathbb{R}^{M \times Q}$  in a bandwidth-efficient manner. Similarly to  $\mathbf{y}$ , we define the per-node partition of  $X$  as

$$X \triangleq \begin{bmatrix} X_1 \\ \vdots \\ X_K \end{bmatrix}, \quad (5.2)$$

where we refer to  $X_k$  as the ‘‘local’’ filter associated with node  $k$ . The  $Q$ -channel filtered signal  $\mathbf{z}(t) \triangleq X^T \mathbf{y}(t)$ , and hence the filter  $X$  should satisfy some optimality criterion, which can be expressed as [58]

$$\begin{aligned} X^* \in \operatorname{argmin}_{X \in \mathbb{R}^{M \times Q}} \varphi(X^T \mathbf{y}(t), X^T B) \\ \text{s.t.} \quad \forall j \in \mathcal{J}_I, \eta_j(X^T \mathbf{y}(t), X^T D_j) \leq 0, \\ \forall j \in \mathcal{J}_E, \eta_j(X^T \mathbf{y}(t), X^T D_j) = 0 \end{aligned} \quad (5.3)$$

where the definitions of the various terms are the same as for (3.8).

As was done in the previous chapters, we assume that they implicitly contain an operator turning the random argument  $X^T \mathbf{y}(t)$  into a deterministic one. The stationarity and perfect statistical estimation assumptions (Assumptions 1.1 and 1.2) are still assumed to hold. In order to also cover the treatment of non-smooth objective functions, we also allow the optimality criterion to be expressed as [59], [63]

$$\begin{aligned} X^* \in \operatorname{argmin}_{X \in \mathbb{R}^{M \times Q}} \varphi(X^T \mathbf{y}(t), X^T B) + \gamma(X^T A) \\ \text{s.t.} \quad \forall k \in \mathcal{K}, \\ \forall j \in \mathcal{J}_I^k, \eta_j(X_k^T \mathbf{y}_k(t), X_k^T D_{j,k}) \leq 0, \\ \forall j \in \mathcal{J}_E^k, \eta_j(X_k^T \mathbf{y}_k(t), X_k^T D_{j,k}) = 0 \end{aligned} \quad (5.4)$$

where the main differences with (5.3) are the restriction of the constraints to be per-node block-separable, i.e., the constraints index sets  $\mathcal{J}_I^k$  and  $\mathcal{J}_E^k$  describe the constraints for a specific node  $k$ , and the functions  $\eta_j$  associated with a particular node  $k$  can only depend on the local filter  $X_k$  associated with that node, and the addition of  $\gamma$ , a possibly non-smooth function. The function  $\gamma$  is also required to be convex and per-node block-separable, i.e., there exist matrices  $A_k$  and functions  $\gamma_k$  such that

$$\gamma(X^T A) = \sum_{k \in \mathcal{K}} \gamma_k(X_k^T A_k), \quad (5.5)$$

which also implies that  $A$  in (5.5) must be a block-diagonal matrix with blocks  $A_k$ . This allows for most of the typical regularizers, including the  $\ell_1$ ,  $\ell_2$ , and the  $\ell_{1,2}$  norm.

We refer to the parametric optimization problem defined by (5.3) as  $\mathbb{P}_S(\mathbf{y}(t), B, \mathcal{D})$ , and  $\mathbb{P}_{NS}(\mathbf{y}(t), B, \mathcal{D}, A)$  for (5.4). Here the set  $\mathcal{D}$  denotes the collection of all the matrices  $D_j$  or  $D_{j,k}$ .

### 5.3 DASF with inexact local solvers

The algorithms described in Chapters 3 and 4 might in practice be difficult to implement, as the computational cost to find a global minimizer of  $\mathbb{P}(\hat{\mathbf{y}}^i, \hat{B}^i, \hat{\mathcal{D}}^i, \hat{A}^i)$  at the updating node might be prohibitive, requiring either too much computations or too much memory. In addition, an exact global solver for the problem of interest might not even exist. For example, many solvers for non-convex problems often produce a stationary point or local optimum rather than a global optimum. Furthermore, their convergence was only guaranteed under relatively strict assumptions. In this section, we show how the local solution step can be significantly relaxed to allow for common iterative solvers, not necessarily converging to optimal values, while still guaranteeing convergence to a stationary point of the centralized problem (5.3) or (5.4), without the need for strict and hard-to-check assumptions.

In what follows, we express (5.3)–(5.4) in the equivalent form

$$\min_{\mathbf{X}} L(X) \quad (5.6)$$

where

$$L(X) \triangleq \varphi(X^T \mathbf{y}(t), X^T B) + \gamma(X^T A) + \delta_{\mathcal{X}}(X) \quad (5.7)$$

with  $\delta_{\mathcal{X}}(\cdot)$  the indicator function of some set  $\mathcal{X}$  described by the equality and inequality constraints, i.e.,  $\delta_{\mathcal{X}}(X) = 0$  if  $X \in \mathcal{X}$  and  $\delta_{\mathcal{X}}(X) = \infty$  otherwise<sup>1</sup>. In the case of the generic problem (5.3) we have

$$\begin{aligned} \mathcal{X} \triangleq \{X \mid \forall j \in \mathcal{J}_I, \eta_j(X^T \mathbf{y}(t), X^T D_j) \leq 0, \\ \forall j \in \mathcal{J}_E, \eta_j(X^T \mathbf{y}(t), X^T D_j) = 0\}, \end{aligned} \quad (5.8)$$

and in the non-smooth case (5.4),

$$\begin{aligned} \mathcal{X} \triangleq \{X \mid \forall k, \forall j \in \mathcal{J}_I^k, \eta_j(X_k^T \mathbf{y}_k(t), X_k^T D_{j,k}) \leq 0, \\ \forall j \in \mathcal{J}_E^k, \eta_j(X_k^T \mathbf{y}_k(t), X_k^T D_{j,k}) = 0\}. \end{aligned} \quad (5.9)$$

From the update rules (3.21) and (3.22) of DASF, one could show that solving  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i, \tilde{A}^i)$  in the space where  $\tilde{X}$  lives is equivalent to solving  $\mathbb{P}(\mathbf{y}, B, D, A)$  in the original space of  $X$  with the additional constraints that

$$\forall k \neq q, \exists \tilde{X}_k \in \mathbb{R}^{Q \times Q} : X_k = X_k^i \tilde{X}_k, \quad (5.10)$$

or equivalently

$$\forall k \neq q, X_k \in \text{range } X_k^i. \quad (5.11)$$

To each local  $\tilde{X}$  is associated a unique  $X$  in the original problem space, a fact that we can express as the linear relationship

$$X = C_q^i \tilde{X} \quad (5.12)$$

where  $C_q^i \in \mathbb{R}^{M \times (M_q + (K-1)Q)}$  was defined in (3.35), leading in particular to

$$X^i = C_q^i [X_q^{iT}, I_Q, \dots, I_Q]^T. \quad (5.13)$$

It is important to note that there is a linear relationship between the *local* variable  $\tilde{X}$  and the corresponding point  $X$  in the *global* space, and that this relationship depends on the current iteration  $i$  (via the previous estimate of the solution  $X^i$ ) and the current updating node  $q$ . This relationship *itself* linearly depends on  $X^i$  via (3.37). The linearity, and hence continuity of the map  $C_q$  is one of the key properties used to show convergence, and replacing  $C_q$  by any other continuous map would lead to the same convergence result (although optimality would not anymore be guaranteed). In particular, in the case of arbitrary network topologies,  $C_q^i(X)$  is replaced by the map  $C_q^i(X)T_q^T$  described

<sup>1</sup> $L$  is thus not a real-valued function, but takes its values in the *extended* real number line  $\mathbb{R} \triangleq \mathbb{R} \cup \{-\infty, \infty\}$ .

in Subsection 4.4.4, and the generalization to arbitrary network topologies can thus be done as it was done in Chapter 4.

Using the notation (5.7) and (5.12), we can now compactly express the local problem at iteration  $i$  as

$$\min_{\tilde{X}} L(C_q^i \tilde{X}). \tag{5.14}$$

We note that –by construction– (5.14) has the same solutions as the local problem  $\mathbb{P}(\tilde{\mathbf{y}}^i, \tilde{B}^i, \tilde{D}^i, \tilde{A}^i)$ . This equivalent problem formulation allows us to offer a unifying treatment of the smooth and non-smooth cases via the compact problem formulation (5.6) (or (5.14) for the local problem at the updating node).

### 5.3.1 DASF algorithm with inexact solver

In what follows, we define the local objective at iteration  $i$

$$\tilde{L}_q^i : \tilde{X} \mapsto L(C_q^i \tilde{X}), \tag{5.15}$$

which we use to construct the following definition of an *inexact iterative solver*.

**Definition 5.1** (Inexact iterative solver). *Given some current estimate  $X^i$ , an inexact iterative local solver running on node  $q$  produces a sequence of iterates  $(\tilde{X}^{i,j})_{j \in \mathbb{N}}$  such that for each iteration  $i$*

(i)

$$\tilde{X}^{i,0} = \begin{bmatrix} X_q^i \\ I_Q \\ \vdots \\ I_Q \end{bmatrix}, \tag{C1}$$

(ii) *there is some  $R^i > 0$  such that for any  $j$ ,*

$$\tilde{L}_q^i(\tilde{X}^{i,j}) - \tilde{L}_q^i(\tilde{X}^{i,j+1}) \geq R^i \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F^2, \tag{C2}$$

(iii) *there is some  $c^i > 0$  such that for any  $j$  there is some  $W \in \partial \tilde{L}_q^i(\tilde{X}^{i,j+1})$  such that*

$$c^i \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F \geq \|W\|_F. \tag{C3}$$

(iv) In addition, the sequences of constants  $R^i$  and  $c^i$  must be such that

$$\liminf_{i \rightarrow \infty} R^i > 0 \quad \text{and} \quad \limsup_{i \rightarrow \infty} c^i < \infty. \quad (\text{C4})$$

Flowing directly from (5.13), condition (C1) ensures consistency with the current global estimate of the solution  $X^i$ , and can be enforced as long as the solver accepts an arbitrarily chosen starting point. Condition (C2) is often referred to as a *sufficient decrease* condition [108], [119]. It ensures that an update of the optimization variable results in a corresponding decrease of the cost, and produces a feasible point (as, by the definition of the indicator function  $\delta_{\mathcal{X}}$ , the cost would otherwise have infinite value in the case of a non-feasible point). Condition (C3) ensures that progressively smaller updates of the optimization variable result in a higher degree of stationarity (the distance of the set  $\partial\tilde{L}(\tilde{X}^{j+1})$  to 0 plays here the role of optimality criterion, and summarizes how “close” we are to a stationary point). This third condition is crucial, as without it, the procedure could wander away from a stationary point during the first few steps, but still eventually converge to some stationary point, in which case a single step of the solver would not be guaranteed to improve on the current iterate. The last requirements on  $R^i$  and  $c^i$  in condition (C4) ensure that those constants do not become arbitrarily small or large as the algorithm progresses, as that would render the two previous conditions meaningless. Those constants are usually related to a parameter of the solver that can be controlled, such as a learning rate or step-size<sup>2</sup>, a lower or upper bound can therefore be enforced explicitly. Although these requirements might seem restrictive, they have been shown to hold for many commonly used descent methods [120], if some additional technical conditions are satisfied. Here are some common examples, some of which are further elaborated on in Appendix E:

**Gradient Descent** (see Appendix E, and [120], [121]) provided that the problem is unconstrained and the objective has Lipschitz gradients.

**Projected/Proximal Gradient Descent** (see [108], [120]) provided that the smooth part of the objective has Lipschitz gradients.

**Newton’s Method** (see Appendix E) provided that the Hessian is positive definite and bounded, and that the objective has Lipschitz gradients.

**Regularized Exact Solver** (see Appendix E) By definition, an exact solver obtains a solution in a single iteration and the properties of Definition 5.1 are almost satisfied. If there are multiple solutions, some mechanism is

<sup>2</sup>See appendix E for examples.

required to ensure that the solver does not “jump” from one solution to the next, by for example explicitly penalizing the distance from the starting point. This was the regularization strategy suggested in Subsection 3.3.5.

**Power Method** (see Appendix E) An intuitive argument is that the power method can be expressed as an instance of projected gradient descent applied to a specific constrained quadratic problem.

Note that the meta-parameters of the subsolvers should be chosen with the same care as if the solver was used centrally. For example, in the case of gradient-descent, the learning rate should be upper-bounded by a constant depending on the Lipschitz constant of the local problem (which can be computed from the data).

**Remark 5.1** (Comparison with the original DASF algorithm). An important, mostly theoretical, difference with the original DASF and NS-DASF algorithms, is the removal of the *well-posedness* assumption on the global problems (Assumptions 3.2, 4.3 and 4.4). Indeed, it was assumed in those algorithms that the solution set of the optimization problem varied “smoothly” with regards to the problem’s parameters  $(\mathbf{y}(t), B, \mathcal{D}, A)$ . This assumption can be hard to check, and even not hold in some cases. Property (C2), which is not satisfied by an exact solver, allows us to remove this assumption.

The inexact version of the DASF Algorithm simply consists in selecting  $\tilde{X}^* \triangleq \tilde{X}^{i, n_i}$  as the solution of an inexact solver applied to the local problem at node  $q$  at iteration  $i$  of the DASF algorithm, which we denote

$$\min_{\tilde{X}} \tilde{L}_q^i(\tilde{X}), \quad (5.16)$$

where the number of iterations  $n_i > 0$  of the inexact solver can be chosen arbitrarily.

### 5.3.2 Convergence and optimality

We first describe a general and easily satisfied assumption on  $L$ .

**Assumption 5.1.**  $L$  is continuous over its domain<sup>3</sup>, and it has compact sublevel-sets. In addition,  $\gamma$  is closed, convex and proper (CCP).

A sufficient condition is to require that  $\varphi$  and the  $\eta_j$  are continuous,  $\gamma$  is convex and continuous over its domain, that either  $\gamma$  and  $\varphi$  or at least one of the  $\eta_j$  have bounded sublevel sets, and that  $\mathbf{y}$  has independent channels (see Lemma F.5).

We show the convergence of this scheme by showing that (i)  $L(X^i)$  decreases monotonically, and (ii) every accumulation point of  $(X^i)_{i \in \mathbb{N}}$  is a stationary point (as defined by (1.21)) of the local problem associated with each node  $q$  (5.16). We then leverage the main results of [62] and Chapter 4 to show optimality with regards to the global optimization problem (5.6).

We first prove two useful subresults: the convergence in objective values, and of the residuals  $X^i - X^{i+1}$ .

**Lemma 5.1** (Monotonic decrease of the objective). *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by the DASF Algorithm with inexact local solver. Then  $(L(X^i))_{i \in \mathbb{N}}$  is a monotonically decreasing convergent sequence.*

*Proof.* Since by the definition (5.15)  $\tilde{L}_q^i(\tilde{X}) = L(C_q^i \tilde{X})$ , from the second property of the local solver (C2), we have

$$L(C_q^i \tilde{X}^{i,j}) - L(C_q^i \tilde{X}^{i,j+1}) \geq R^i \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F^2 \geq 0. \quad (5.17)$$

Summing from  $j = 0$  to  $j = n_i$  and telescoping the left-hand side yields

$$L(X^i) - L(X^{i+1}) \geq 0, \quad (5.18)$$

where we used (C1) to substitute  $X^i = C_q^i \tilde{X}^{i,0}$  and the definition of the inexact algorithm to substitute  $X^{i+1} = C_q^i \tilde{X}^{i,n_i}$ , hence proving the monotonic decrease. As  $L$  is continuous over its domain with compact sub-level sets, it has a minimum value, and the sequence  $(L(X^i))_{i \in \mathbb{N}}$  is therefore bounded-below [70] and must converge [69].  $\square$

<sup>3</sup>The domain of a function  $L$  is the set of points for which  $L(X) < \infty$ .

**Lemma 5.2** (Convergence of the Sequence of Residuals). *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by the DASF Algorithm with inexact local solver, and let  $(\tilde{X}^{i,j})_{j \in \mathbb{N}}$  be the subsequence generated by the local solver at each iteration  $i$ . Then ,*

$$\lim_{i \rightarrow \infty} \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F^2 = 0 \quad \forall j \quad (5.19)$$

and

$$\lim_{i \rightarrow \infty} \left\| X^i - X^{i+1} \right\|_F = 0. \quad (5.20)$$

*Proof.* Using the fact that

$$X^i = C_q^i \tilde{X}^{i,0} \text{ and } X^{i+1} = C_q^i \tilde{X}^{i,n_i}, \quad (5.21)$$

we have

$$L(X^i) - L(X^{i+1}) = \sum_{j=0}^{j=n_i} L(C_q^i \tilde{X}^{i,j}) - L(C_q^i \tilde{X}^{i,j+1}), \quad (5.22)$$

From the second property of the local solver (C2), for any  $i$  and  $j \leq n_i$

$$L(C_q^i \tilde{X}^{i,j}) - L(C_q^i \tilde{X}^{i,j+1}) \geq 0, \quad (5.23)$$

it must be that, for any  $i$  and  $j \leq n_i$

$$L(X^i) - L(X^{i+1}) \geq L(C_q^i \tilde{X}^{i,j}) - L(C_q^i \tilde{X}^{i,j+1}), \quad (5.24)$$

and hence from (C2) again

$$L(X^i) - L(X^{i+1}) \geq R^i \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F^2. \quad (5.25)$$

As a consequence of Lemma 5.1, the convergence of  $(L(X^i))_{i \in \mathbb{N}}$  in combination with (5.25) implies that, for any  $0 \leq j < n_i$ ,

$$\lim_{i \rightarrow \infty} L(X^i) - L(X^{i+1}) = \lim_{i \rightarrow \infty} R^i \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F^2 = 0. \quad (5.26)$$

From property (C4) of the local solver, the sequence  $R^i$  is eventually lower-bounded, and therefore

$$\lim_{i \rightarrow \infty} \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F^2 = 0 \quad \forall j, \quad (5.27)$$

proving the first statement. From (5.21) and the triangle inequality, we have

$$\begin{aligned} \|X^i - X^{i+1}\|_F &\leq \sum_{j=0}^{n_i-1} \left\| C_q^i \tilde{X}^{i,j} - C_q^i \tilde{X}^{i,j+1} \right\|_F \leq \\ &\|C_q^i\|_F \sum_{j=0}^{n_i-1} \left\| \tilde{X}^{i,j} - \tilde{X}^{i,j+1} \right\|_F, \end{aligned} \quad (5.28)$$

where the second inequality relies on the sub-multiplicative property of the Frobenius norm. As  $C_q^i = C_q(X^i)$ , and as the sub-level sets of  $L(X^0)$  are compact,  $\|X^i\|_F$  is bounded and by the continuity of the map  $C_q(X)$ , so is  $\|C_q^i\|$ . (5.26) and (5.28) therefore imply that

$$\lim_{i \rightarrow \infty} \|X^i - X^{i+1}\|_F = 0, \quad (5.29)$$

completing the proof.  $\square$

We now use the above result to show that any accumulation point is a stationary point of the local problem at every node.

**Lemma 5.3** (Node-specific Optimality of Accumulation Points). *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by the DASF Algorithm with inexact local solver. Then, every accumulation point  $\bar{X}$  of  $(X^i)_{i \in \mathbb{N}}$  is a stationary point of the local problem (5.16) for any updating node  $q$ .*

*Proof.* By the third and fourth property of the local solver (C3)–(C4), we can find sequences  $(W^i)_{i \in \mathbb{N}}$  and  $(c^i)_{i \in \mathbb{N}}$  (with  $c^i > 0$ ) such that

$$W^i \in \partial \tilde{L}_q^i(\tilde{X}^{i,n_i}), \quad (5.30)$$

and

$$c \left\| \tilde{X}^{i,n_i-1} - \tilde{X}^{i,n_i} \right\|_F \geq \|W^i\|_F, \quad (5.31)$$

where  $c = \limsup_i c^i$ . We can express equation (5.30) in the global domain of  $X$  by applying the generalized chain rule<sup>ab</sup> [68] to  $L(C_q^i \tilde{X}^{i,n_i})$ :

$$\begin{aligned} \partial \tilde{L}_q^i(\tilde{X}^{i,n_i}) &= \partial_{\tilde{X}} \left( L(C_q^i \tilde{X}^{i,n_i}) \right) \\ &= C_q^{iT} \partial L(C_q^i \tilde{X}^{i,n_i}) = C_q^{iT} \partial L(X^{i+1}) \end{aligned} \quad (5.32)$$

where we used the fact that  $C_q^i \tilde{X}^{i,n_i} = X^{i+1}$  by definition of the algorithm. If  $\bar{X}$  is an accumulation point, then there is (by definition) some index set  $\mathcal{I}$  such that  $\lim_{i \in \mathcal{I} \rightarrow \infty} X^i = \bar{X}$ . Furthermore, from Lemma 5.2 and in particular (5.20), we also have

$$\lim_{i \in \mathcal{I} \rightarrow \infty} X^{i+1} = \lim_{i \in \mathcal{I} \rightarrow \infty} X^i = \bar{X}. \quad (5.33)$$

As the number of nodes is finite, we can furthermore select  $\mathcal{I}$  such that the sequence of updating nodes  $(q^i)_{i \in \mathcal{I}}$  is a constant sequence, i.e. we only consider a single updating node  $q$ . From the continuity of the map  $C_q(X)$  it must also be that there is some  $\bar{C}_q \triangleq C_q(\bar{X})$  such that

$$\lim_{i \in \mathcal{I} \rightarrow \infty} C_q^i = \lim_{i \in \mathcal{I} \rightarrow \infty} C_q(X^i) = C_q(\bar{X}) = \bar{C}_q. \quad (5.34)$$

From (5.30), (5.32) and the outer-semicontinuity of the set of subgradients [68], we have

$$W^i \in C_q^{iT} \partial L(X^{i+1}) \forall i \Rightarrow \lim_{i \in \mathcal{I} \rightarrow \infty} W^i \in \bar{C}_q^T \partial L(\bar{X}). \quad (5.35)$$

From (5.19) and (5.31), it must be that

$$\lim_{i \rightarrow \infty} \|W^i\|_F = 0, \quad (5.36)$$

and thus

$$\lim_{i \in \mathcal{I} \rightarrow \infty} W^i = 0. \quad (5.37)$$

Combining (5.37) with (5.35) yields

$$0 \in C_q(\bar{X})^T \partial L(\bar{X}) \quad (5.38)$$

which is the condition for  $\bar{X}$  to be a stationary point of the local problem (5.16), based on the argument in (5.32). Note that the local problem

(5.16) is equal to (5.6) equipped with the additional constraint (5.10), i.e.  $\bar{X}$  is a stationary point of

$$\min_X L(X) \quad \text{s.t.} \quad X = C_q(\bar{X})\tilde{X}, \quad \tilde{X} \in \mathbb{R}^{((K-1)Q+M_q) \times Q}.$$

We have shown that an accumulation point is a stationary point for at least a single node  $q$ , it remains to show that it is the case for any  $q$ . From Lemma 5.2 and in particular (5.20), if  $\bar{X}$  is an accumulation point of  $(X^{i+1})_{i \in \mathcal{I}}$ , then it is also an accumulation point of  $(X^{i+n})_{i \in \mathcal{I}}$  for any  $n \geq 0$ . Indeed,

$$X^{i+n} = X^i - (X^i - X^{i+1}) - \dots - (X^{i+n-1} - X^{i+n}), \quad (5.39)$$

and therefore

$$\begin{aligned} \lim_{i \rightarrow \infty} X^{i+n} - X^i = \\ \lim_{i \rightarrow \infty} (X^i - X^{i+1}) - \dots - (X^{i+n-1} - X^{i+n}) = 0, \end{aligned} \quad (5.40)$$

and finally

$$\lim_{i \in \mathcal{I} \rightarrow \infty} X^i = \lim_{i \in \mathcal{I} \rightarrow \infty} X^{i+n} = \bar{X}. \quad (5.41)$$

As we have shown that  $\bar{X}$  is a stationary point for at least one node, it is a stationary point for all nodes as the sequence  $(X^{i+n})_{i \in \mathcal{I}}$  will be associated with node<sup>c</sup>  $(q+n) \bmod K$ .  $\square$

<sup>a</sup>We denote  $\partial_{\tilde{X}}(\cdot)$  the set of subgradients of the expression in argument, interpreted as a function of the local variable  $\tilde{X}$ .

<sup>b</sup>We define the product between a set and a matrix as the set whose elements are the elements of the original set multiplied by that matrix.

<sup>c</sup>assuming a sequential selection rule, but the conclusion stays valid as long as Assumption 4.1 holds.

Lemma 5.3 asserts that any accumulation point of DASF is a stationary point of every local problems, at every nodes. We will use this result to show that these accumulation points are also stationary points of the centralized problem (5.3)–(5.4). For this, we need a technical condition that is usually satisfied with high probability if the number of constraints is not too high (see below). The condition is the same as the one used in Theorem 3.2, and can actually be viewed as a compressed version of the standard linear independence constraint qualifications (LICQ) in numerical optimization [121], except that they apply

to the “compressed” gradients rather than the actual gradients of the constraint functions.

**Proposition 5.1 (Compressed LICQ).** *Let  $\vartheta_j : X \mapsto \eta_j(X^T \mathbf{y}, X^T D_j)$  in the smooth case, and  $\vartheta_j : X \mapsto \eta_j(X_k^T \mathbf{y}_k, X_k^T D_{j,k})$  in the non-smooth case, and define the active constraint set  $\mathcal{A}(X) \triangleq \{j \in \mathcal{J}_I \mid \vartheta_j(X) = 0\}$ . If the elements of the set*

$$\{\text{BlockDiag}(\bar{X}_1, \dots, \bar{X}_K)^T \nabla \vartheta_j(\bar{X}) \mid j \in \mathcal{A}(\bar{X}) \cup \mathcal{J}_E\} \quad (5.42)$$

*are linearly independent matrices and  $\bar{X}$  is a stationary point of the local problems for any node  $q$ , i.e.*

$$\forall q \in \mathcal{K}, 0 \in \bar{C}_q^T \partial L(\bar{X}), \quad (5.43)$$

*then it is also a stationary point of the global problem (5.3)-(5.4), i.e.*

$$0 \in \partial L(\bar{X}). \quad (5.44)$$

The proof is the same as for Theorem 3.2 in the smooth case and Proposition 4.3 in the non-smooth case. Note that this qualification is automatically violated if the number of active constraints exceeds  $KQ^2$  (which is the dimension of the vector space in which the compressed gradients live). See [58], [62] for details and an extension of this condition to the case of arbitrary network topologies.

We are now able to state our main result.

**Theorem 5.1 (Convergence and Optimality).** *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by the DASF algorithm with inexact solver. Then if Assumption 5.1 is satisfied and the qualification (5.42) is also satisfied at the accumulation points of  $(X^i)_{i \in \mathbb{N}}$ ,  $(X^i)_{i \in \mathbb{N}}$  converges to the set of stationary points of the global problem (5.3)-(5.4), that is*

$$\lim_{i \rightarrow \infty} \min_{X \in \Omega} \|X - X^i\|_F = 0, \quad (5.45)$$

*where  $\Omega$  denotes the set of stationary points of problem (5.3)-(5.4). Furthermore, if the number of stationary points of (5.3)-(5.4) is finite, then  $(X^i)_{i \in \mathbb{N}}$  converges to a single point.*

*Proof.* Under Assumption 5.1, Lemma 5.3 asserts that any accumulation point is a stationary point of the local problem at every node. Proposition 5.1 ensures that under the qualification (5.42), the local stationarity at every node implies stationarity for the global problem, proving the first part of the theorem (i.e., convergence to the set of stationary points). The second part (convergence to a single point) is a direct consequence of Lemma 5.2.  $\square$

As a consequence of the above theorem, interleaving the steps of DASF with a single descent step is sufficient to ensure convergence to a stationary point.

### 5.3.3 A bound on the convergence rate

Obtaining a convergence rate for the objective value would unfortunately require much more assumptions on the functions involved in (5.3)–(5.4), than what we assumed so far. Still, we can obtain a lower bound on the convergence rate of a local optimality measure at each node, that is

$$w^i \triangleq \text{dist}(0, \partial \tilde{L}_{q^i}^i(\tilde{X}^{i, n_i})) \triangleq \min_{W \in \partial \tilde{L}_{q^i}^i(\tilde{X}^{i, n_i})} \|W\|. \quad (5.46)$$

As our actual objective is to find stationary points rather than an actual minimum,  $w^i$  is an appropriate measure of optimality. We could also have considered the step-length  $\|X^i - X^{i+1}\|$  as a measure of optimality, which would yield a rate similar to the one presented hereafter.

**Proposition 5.2** (Convergence Rate Bound). *Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by the DASF algorithm with inexact solver, and let  $w^i$  be as defined in (5.46). Then there is some positive constant  $a$  such that*

$$\min_{j \leq i} w^j \leq a \frac{\sqrt{L(X^0) - L^*}}{\sqrt{i+1}}, \quad (5.47)$$

where  $L^*$  is the minimum value of  $L$ .

*Proof.* This follows from a well-known proof argument [108], [119]. From (5.25) and condition (C3), we have

$$L(X^i) - L(X^{i+1}) \geq \frac{R^i}{c^i} (w^i)^2. \quad (5.48)$$

Summing the inequality from 0 to  $i$  and telescoping yields

$$L(X^0) - L(X^{i+1}) \geq \sum_{j=0}^i \frac{R^j}{c^j} (w^j)^2. \quad (5.49)$$

From condition (C4), there is some  $r > 0$  such that

$$\liminf_{i \rightarrow \infty} \frac{R^i}{c^i} = r, \quad (5.50)$$

and hence, using the fact that  $L^* \leq L(X^{i+1})$ ,

$$L(X^0) - L^* \geq r \sum_{j=0}^i (w^j)^2 \geq r(i+1) \min_j (w_j)^2, \quad (5.51)$$

which can be rearranged to obtain (5.47) with  $a = r^{-1/2}$ . □

For a given level of optimality  $\min w^i$ , the bound gives a linear relationship between the required number of iterations and the current error  $L(X^i) - L^*$ . In other words, if the algorithm is set to stop after a sufficiently small norm of the subgradient is reached, we can guarantee that the algorithm will run for a number of iterations (hence time) that is at most proportional to the current error  $L(X^i) - L^*$ .

## 5.4 Numerical experiments

In order to qualitatively demonstrate the convergence of the DASF algorithm with inexact solver, we consider an SNR maximization (Max-SNR) problem [36] defined as

$$\max_x \mathbb{E} \{ \|x^T \mathbf{y}(t)\|^2 \} \quad \text{s.t.} \quad \mathbb{E} \{ \|x^T \mathbf{n}(t)\|_F^2 \} = 1 \quad (5.52)$$

where  $x$  denotes the  $M \times 1$  spatial filter ( $Q = 1$ ),  $\mathbf{y}(t)$  is modelled as  $\mathbf{y}(t) = \mathbf{a}d(t) + \mathbf{n}(t)$ ,  $\mathbf{n}(t)$  is an  $M$ -dimensional i.i.d white Gaussian noise signal,  $d(t)$  is a single-channel (unknown) target signal, and where  $\mathbf{a}$  is some unknown mixing vector with entries sampled from  $\mathcal{N}(0, 1)$ . The purpose of the multi-input single-output (MISO) filter  $x$  is to maximize the SNR of  $d$  in the output filtered signal  $z(t) = x^T \mathbf{y}(t)$ . It is assumed that the nodes of the WSN are able to

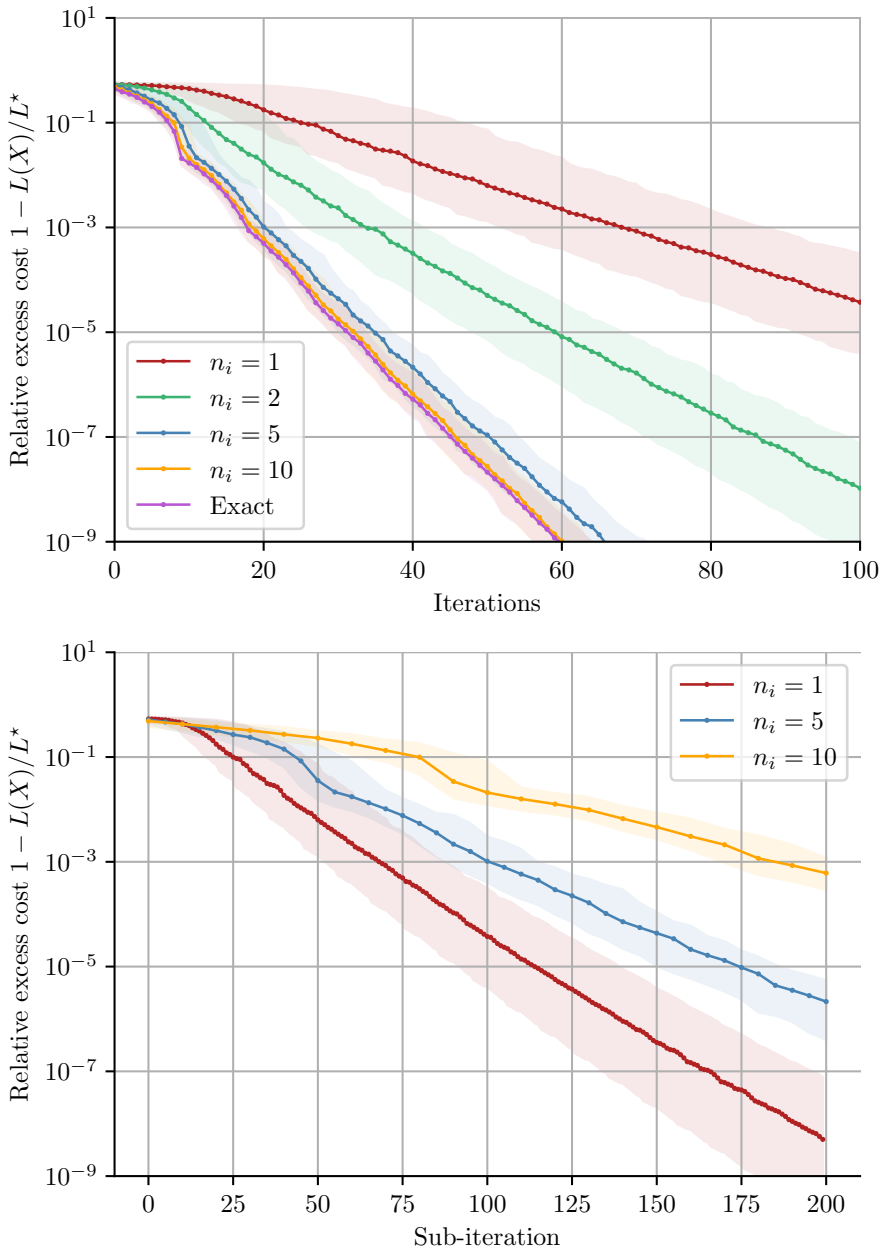


Figure 5.1: Convergence of the DASF algorithm with inexact solver for the Max-SNR problem. Each curve was generated by 1000 Monte-Carlo runs. Solid curves depict the median trajectories, while the shaded areas of corresponding color depict the the 5%-95% percentiles regions. The top plot depicts the excess cost as a function of the DASF iterations, while the bottom plot depicts it as a function of the total number of sub-solver iterations.

collect samples of both  $\mathbf{y}(t)$  and  $\mathbf{n}(t)$ . For our simulations we set  $M = 100$ ,  $M_k = 10$ ,  $K = 10$ , i.e., there are 10 nodes, with each node collecting 10 different channels of the 100-channel signals  $\mathbf{y}(t)$  and  $\mathbf{n}(t)$ . The noise variance is 10, while the signal variance is 1. Our figure of merit is the relative excess cost, defined as

$$1 - \frac{L(X)}{L^*}, \tag{5.53}$$

where  $L^*$  denotes the optimal objective value of (5.52). Note that the solution of (5.52) can be solved by computing a generalized eigenvalue decomposition (GEVD) on the covariance matrices  $\mathbb{E}\{\mathbf{y}(t)\mathbf{y}(t)^T\}$  and  $\mathbb{E}\{\mathbf{n}(t)\mathbf{n}(t)^T\}$  [95].

Figure 5.1 depicts the convergence of the inexact version of DASF applied to problem (5.52) with different numbers of sub-solver iterations ( $n^i$ ). The exact (GEVD) solver uses Cuppen’s divide and conquer algorithm [122], while the inexact solver relies on the generalized power method [123]. On the left part of the figure, we see that the algorithm converges to the optimal objective value with a single sub-solver iteration per-node, and that 10 sub-solver iterations are sufficient to reach a convergence rate at par with the exact solver. Although, based on this first plot, one might conclude that more sub-solver iterations is better, the right part of Figure 5.1 tells a very different story. The same figure of merit is depicted, but in terms of the total number of sub-solver iterations, rather than “global” DASF iterations. This second plot therefore depicts the algorithm’s convergence as a function of the actual number of computations and offers a fairer comparison. Indeed, the sub-solver with the lowest number of iterations ( $n_i = 1$ ) outperforms all the other ones. It seems that the smaller progress achieved at each step is largely compensated by the fact that the rate at which the updating node role is passed-on is much larger, resulting in the network being able to update the filter in more “directions” (i.e. with different subspace constraints (5.10)) within a given computational budget, albeit with smaller progress at each updating node.

The above simulation was performed under the assumption of perfect statistical estimation, and the full data (i.e. all samples) is used by the updating at each iteration, allowing it to estimate the required statistics, and factor out the effect of statistical estimation errors. Some more details are therefore required to interpret this second plot. In Algorithm 3.2, it is assumed that every time the updating node is switched, a new batch of samples should be sent by every node to the updating node. Performing a single sub-iteration therefore implies that less samples are received by the updating node (as less time has elapsed since the previous iteration). What we actually wish to know is whether DASF with a single-iteration would also be the fastest to converge to the optimal filter, as a function of “real” time, or number of samples processed, but the answer to this question largely depend on the number of samples necessary to

obtain reasonable estimations of the statistics, which itself depends on the data generating process. The second plot can therefore be interpreted as converging faster as a function of real time, only if we assume that even in the case of a single sub-solver iteration, the number of samples received is sufficient to estimate the signal statistics with negligible error. There is however a way out for the cases where this assumption would not hold. By applying the data-reuse technique described in [117], the same batch of samples can be iterated over by multiple updating nodes, making the number of samples available at each iteration independent of the rate at which computations are performed (that is as long as an iteration is much faster than the collection of a batch of samples). In this case, the second plot also depicts the convergence as a function of “real” time, and makes the DASF algorithm with inexact local solver an excellent option for adaptively tracking a spatial filter.

## 5.5 Conclusion

We have extended the convergence results of the DASF algorithm to a setting where the exact solver is replaced by an inexact, iterative one, therefore allowing significant computational gains. We have shown, that the convergence properties of DASF were preserved, even under relaxed technical conditions. Furthermore, we have shown via simulations that DASF used with an iterative solver could display better convergence properties, at the cost of a slightly increased bandwidth, assuming a data-reuse scheme as in [117].

# Chapter 6

---

## Conclusion

---

The objective of this thesis was to design signal processing algorithms that are amenable to run on the resource-constrained nodes of a wireless sensor network, or more specifically, to allow the computation of adaptive data-driven spatial filters within the constraints of a wireless sensor network. The research positions itself within the already established context of DASF and its ancestors within the DSF family of algorithms. As such, the focus was on leveraging and extending the existing algorithmic ideas where possible.

**Contributions** We explored two orthogonal directions in pursuit of possible energy-efficiency gains. We first challenged the assumption that all the signals within a sensor network need to be processed together by every node, for every task, and looked into ways to prune the network that allows the target task to be performed with a reduced node count, or split between non-communicating subsets of nodes. Indeed, due to the spatial configuration of the sensor nodes, it is likely that different subsets of nodes observe different, possibly uncorrelated phenomena. In order to achieve energy-reduction, it therefore makes sense to evaluate the correlation structure of the network a-priori, or periodically, to identify which nodes should exchange data. This was the focus of Chapter 2. The contributions of this chapter are first (i) a distributed algorithm for the MAXVAR problem, which is in itself useful, (ii) the introduction of the total squared correlation, a metric which can appropriately summarize the degree of overlap of the latent subspaces of two multi-channel signals and admits an efficient computational procedure, and (iii) the said procedure. In this chapter, we have shown via theoretical derivations and numerical simulations that our procedure converged towards a correct estimate of the total squared correlation.

This first attempt at network pruning is however task-agnostic, and relies on the assumption that inter-node correlation is an appropriate proxy for a node's utility. We addressed this lack in Chapter 4, where we extended the scope of the DASF algorithm to spatial filters than can be computed as the solution of non-smooth optimization problems. This notably allows the introduction of sparsity-inducing regularizers in the filter's formulation. The presence of such regularizers in the objective can encourage node to set their local filter to zero, and effectively stop participating to the filtering task. This allows a fine tuning of the trade-off between bandwidth and filter optimality, as the weights attributed to the main cost function and regularizer can be tuned for each application, even adaptively. We have shown the theoretical convergence of the procedure, and validated its pruning capabilities via simulations.

We also looked into algorithmic improvements that could lead to energy savings. Specifically, we have shown in Chapter 5 that the exact solver used by DASF to solve each local problem at the updating nodes can be replaced by a few iterations of an iterative solver, without sacrificing convergence speed. Even more, this leads to a more stable procedure, that converges under relaxed assumptions. We have described the technical conditions that such an iterative solver should satisfy to guarantee convergence, and have shown that they were met by most commonly used solvers, amongst them, all the well-known variants of gradient descent, such as proximal gradient descent, and hence the projected gradient method. We have shown that these technical conditions lead to the convergence of the filter.

The above contributions all relied on our initial work<sup>1</sup> on the general convergence result of the DASF algorithm in Chapter 3, which established the basic proof framework underpinning the rest of our work.

**Discussion and suggested future research** At the very beginning of this thesis, we introduced two key assumptions: (i) the signals are stationary, and (ii) each node is able to perfectly estimate the signal statistics based on the samples it received. These assumptions were not meant as statements about a real-world setting, but were merely a mean to decouple the effect of statistical phenomenons from the algorithmic properties of our procedures. It must be clear to anyone, that the world is anything but stationary, and it is therefore essential to study the interplay between adaptive signal processing procedures and non-stationary data. In that regard, some effort should be put in studying the convergence rate of the procedures in the case of a filter whose optimal value evolves over time, i.e. theoretically analyze the tracking capabilities. Furthermore, DASF is more efficient than a centralized procedure only

---

<sup>1</sup>joint work with C. Musluoglu and A. Bertrand in [62].

in an adaptive setting where the iterations of DASF are spread over different sample batches such that the filter updates happen in a time-recursive fashion. A stationary setting would allow for batch-mode computations where all the iterations of DASF are performed on a single batch of sensor observations, in which case the amount of transmitted data might become larger than the amount of raw sensor data within the batch (depending on the number of executed DASF iterations and the topology of the network). In this setting, sending the batch of data once to a fusion center, where the filter can be obtained by solving a single problem involving the full batch of samples would be more effective. Fully stationary data is therefore antithetic to the stated design goals of DASF.

The impact of statistical estimation errors should also be studied. Indeed, not only can it reinforce the result of this thesis by providing further convergence guarantees, it can open the door to stochastic variants of the algorithms introduced in this text. In particular, using stochastic gradient descent as the solver in our inexact DASF variant seems like a natural extension, and a similar proof can probably be derived, where the technical conditions are only required to hold in expectation.

Although we have provided the basic infrastructure to allow efficient network pruning, we did not study actual pruning strategies. This is an important step towards real applications, and practical systems.

Finally, if DASF is ever to be used in an engineering setting, it is essential to confront it to real hardware and real data. In particular, to study the true computational and transmission budgets of actual WSN components, and check whether DASF can be used to track real-world data within those budgets. The energy savings that could be achieved by DASF largely depend on the true energy consumptions associated with computations and data transmissions, this trade-off should therefore be studied for real systems as well.

By taking on all the above points, we believe that DASF has the potential to go from a mathematically interesting procedure to an actual tool on an engineer's belt.



# Appendix A

---

## Proofs of Chapter 2

---

### A.1 Proof of Theorem 2.1

*Proof.* Let us assume that  $X^*$  is a fixed point of Algorithm 2.1. Then  $X^*$  is a solution of problem (2.19) when  $X^i = X^*$  for any  $q \in \mathcal{K}$ . The Lagrangian of this problem can be expressed as

$$\begin{aligned} L_q(X, \Lambda^q, \Gamma_k) = & \text{Tr}(X^T R_D X) \\ & - \text{Tr}(\Lambda^q (X^T R_{\mathbf{y}\mathbf{y}} X - I^Q)) \\ & - \sum_{k \neq q} \text{Tr}(\Gamma_k X_k^T S_k) \end{aligned} \quad (\text{A.1})$$

where  $\Lambda^q$  and  $\Gamma_k \forall k \neq q$  are matrices of proper dimensions containing the Lagrange multipliers and  $S_k$  is an  $M_k \times (M_k - Q)$  matrix whose columns span the left null space of  $X_k^*$ . As  $X^*$  is a solution of (2.19), it must satisfy

$$\frac{\partial}{\partial X} L_q = 0 = 2R_D X^* - 2R_{\mathbf{y}\mathbf{y}} X^* \Lambda^q - S^q \Gamma^q \quad (\text{A.2})$$

where  $\Gamma^q$  is the matrix obtained by vertically stacking all the rows of  $\Gamma_k$  and  $S^q$  is the block diagonal matrix whose blocks are  $S_k$ , and where the entries of the blocks corresponding to  $q$  are set to zero for both matrices.

Left-multiplying by  $X^{*T}$  and using constraint (2.19b), we obtain

$$X^{*T} R_D X^* = \Lambda^q - \frac{1}{2} \underbrace{X^{*T} S^q \Gamma^q}_0 \quad \forall q. \quad (\text{A.3})$$

Since the left-hand side is independent of  $q$ , we can conclude that  $\Lambda^q = \Lambda$  is the same for every choice of  $q$ . From this and (A.2), we have

$$\frac{\partial}{\partial X_q} L_q = 0 = 2R_{\mathbf{y}_q} X_q^* - 2R_{\mathbf{y}_q \mathbf{y}} X^* \Lambda - \underbrace{S_q \Gamma_q}_0 \quad \forall q \quad (\text{A.4})$$

Combining those equations for  $q \in \mathcal{K}$  yields  $R_D X^* = R_{\mathbf{y}\mathbf{y}} X^* \Lambda$ . As  $X^*$  is computed via the GEVD of the local pencil  $(R_{D_q}^i, R_{\mathbf{y}_q}^i)$  corresponding to  $X^i = X^*$ , it must be such that the local  $\bar{X}$  in (2.16) diagonalizes  $R_{D_q}^i$  at every updating node  $q$ . As a consequence, and considering that the update matrices  $\tilde{X}_k$  are identity matrices due to the fact that  $X^*$  is a fixed point,

$$\bar{X}^T R_{D_q}^i \bar{X} = X^{*T} R_D X^* = \Lambda \quad (\text{A.5})$$

is a diagonal matrix and the columns of  $X^*$  are therefore GEVCs of the pencil  $(R_D, R_{\mathbf{y}\mathbf{y}})$ . □

## A.2 Proof of Theorem 2.2

In order to prove Theorem 2.2, we first prove two intermediate results. In the following, let  $f : \mathbb{R}^{M \times Q} \rightarrow \mathbb{R}$  denote the objective function (2.14a) and let the constraint set (2.14b) be

$$\mathcal{D} = \{X \in \mathbb{R}^{M \times Q} \mid X^T R_{\mathbf{y}\mathbf{y}} X = I_Q\}. \quad (\text{A.6})$$

Furthermore, let the constraint set of the local problems (2.19) be

$$\mathcal{D}_q(V) = \{X \in \mathcal{D} \mid \text{range } X_k \subseteq \text{range } V_k \quad \forall k \neq q\} \quad (\text{A.7})$$

where  $V = X^i$  in (2.19). For convenience, we define the following equivalent procedure to an update of Algorithm 2.1:

$$X^{i+1} \in \underset{X \in \mathcal{M}^i}{\text{argmin}} \|X - X^i\|_F \quad (\text{A.8a})$$

$$\mathcal{M}^i = \underset{X \in \mathcal{D}_{q_i}(X^i)}{\text{argmin}} f(X) \quad (\text{A.8b})$$

where  $q_i = i \bmod K$ . We denote  $F_q$  the mapping producing a new iterate  $X^{i+1}$  from  $X^i$  at the updating node  $q$  at iteration  $i$ .

**Lemma A.1.** *Let  $m_q : \mathbb{R}^{M \times Q} \rightarrow \mathbb{R}$  be such that  $\{m_q(X)\} = f(F_q(X))$ , i.e. mapping  $X^i$  to  $f(X^{i+1})$  in procedure (A.8). Then  $m_q$  is a continuous function for any  $q$  at points where the blocks  $X_{k \neq q}$  have full column rank.*

*Proof.* The constraint set of the local problem (2.16) can be expressed as

$$\mathcal{D}_q(X) = \{P_X^q V \mid V \in \mathbb{R}^{M \times Q}\} \cap \mathcal{D} \tag{A.9}$$

where  $P_X^q$  denotes the orthogonal projection matrix on the linear subspace (A.7).  $m_q(X)$  therefore corresponds to the sum of the  $Q$  smallest GEVLs of  $(P_X^q R_D P_X^{qT}, P_X^q R_{yy} P_X^{qT})$  (which is found by substituting the optimization variable in (2.16) by  $P_X^q V$ ). As the GEVLs of a pencil vary continuously with the entries of its constituent matrices [124], and as  $P_X^q$  varies continuously with  $X$  at points where its blocks  $X_{k \neq q}$  have linearly independent columns [114] (which is true under the assumption that the local problems are non-singular),  $m_q$  is a continuous function at points where the blocks  $X_{k \neq q}$  have full column rank.  $\square$

Let  $(X^i)_{i \in \mathbb{N}}$  be any sequence of iterates satisfying the mapping defined by Algorithm 2.1 and therefore procedure (A.8). We now show that if  $(X^i)_{i \in \mathbb{N}}$  has an accumulation point, then it is a fixed point of Algorithm 2.1 and therefore a stationary point of problem (2.14).

**Lemma A.2.** *The accumulation points of  $(X^i)_{i \in \mathbb{N}}$  are fixed points of Algorithm 2.1.*

*Proof.* Let  $X^*$  be an accumulation point of  $(X^i)_{i \in \mathbb{N}}$ . From the continuity of  $m_q$  (see Lemma A.1), and under the assumption of non-singular local pencils, we have:

$$\lim_{X \rightarrow X^*} m_k(X) = m_k(X^*) \quad \forall k \in \mathcal{K}. \tag{A.10}$$

As  $(f(X^i))_{i \in \mathbb{N}}$  is bounded and decreases monotonically, it converges to some  $f^*$  and therefore, from the continuity of  $f$ ,  $f(X^*) = f^*$ . By

definition of  $m_q$  we have

$$m_{q_i}(X^i) = f(X^{i+1}) \quad (\text{A.11})$$

with  $q_i = i \bmod K$ . Therefore,

$$\lim_{i \rightarrow \infty} m_{q_i}(X^i) = f^*. \quad (\text{A.12})$$

As  $X^*$  is an accumulation point of  $(X^i)_{i \in \mathbb{N}}$ , there is some index set  $\mathcal{N}_k \subseteq \mathbb{N}$  such that  $(X^i)_{i \in \mathcal{N}_k}$  converges to  $X^*$  and  $(q_i)_{i \in \mathcal{N}_k} = (k)_i$  for some node  $k$ . From (A.12) and the continuity of  $m_q$  we have

$$m_k(X^*) = f^* = f(X^*) \quad (\text{A.13})$$

As, by definition (A.7),  $X^*$  is in  $\mathcal{D}_k(X^*)$ , and by the definition of  $m_q$ , the minimum value of  $f$  in  $\mathcal{D}_k(X^*)$  is  $m_k(X^*) = f^*$ ,  $X^*$  must be in  $\mathcal{M}$  (as defined in (A.8b)). In virtue of (A.8a), it must be that  $F_k(X^*) = X^*$  and  $X^*$  is therefore a fixed point of  $F_k$ .

We will now establish that  $X^*$  is also a fixed point of node  $k+1$ . As, by hypothesis, the  $Q$ -th and  $(Q+1)$ -th smallest GEVLs of the local pencils  $(R_{D_k}^i, R_{\tilde{y}_k}^i)$  are distinct at the accumulations points of  $(X^i)_{i \in \mathbb{N}}$ , a small perturbation of the pencil around an accumulation point results in a small perturbation of the generalized eigenspace [78], [124]<sup>a</sup>. As a consequence, the convergence of the subsequence  $(X^i)_{i \in \mathcal{N}_k}$  to  $X^*$  implies the convergence of  $(\mathcal{M}^i)_{i \in \mathcal{N}_k}$  to some  $\mathcal{M}^*$ , which from (A.8) corresponds to the sequence of sets of generalized eigenvectors of the local pencils  $(R_{\tilde{y}_q}^i, R_{D_q}^i)$ , where the convergence of  $(\mathcal{M}^i)_{i \in \mathcal{N}_k}$  must be understood in terms of the Hausdorff distance between sets <sup>b</sup>.

As  $m_k(X^*) = f^*$  and  $f(X^*) = f^*$ , it must be that  $X^* \in \mathcal{M}^*$ . Therefore, there exists some convergent sequence  $\{V^{i+1}\}_{i \in \mathcal{N}_k}$  with  $V^{i+1} \in \mathcal{M}^i$  converging to  $X^*$  (As the convergence of  $(\mathcal{M}^i)_{i \in \mathcal{N}_k}$  to  $\mathcal{M}^*$  implies that for any point  $X$  in  $\mathcal{M}^*$  we can find a set  $\mathcal{M}^i \in (\mathcal{M}^i)_{i \in \mathcal{N}_k}$  containing a point arbitrarily close to  $X$ ). As both sequences converge to the same point,

$$\lim_{i \rightarrow \infty, i \in \mathcal{N}_k} \|X^i - V^{i+1}\|_F = 0, \quad (\text{A.14})$$

and as  $V^{i+1} \in \mathcal{M}^i$ , we have from (A.8a) that

$$\|V^{i+1} - X^i\|_F \geq \min_{X \in \mathcal{M}^i} \|X - X^i\|_F = \|X^{i+1} - X^i\|_F. \quad (\text{A.15})$$

(A.14) in combination with the squeeze theorem therefore implies that

$$\lim_{i \rightarrow \infty, i \in \mathcal{N}_k} \|X^{i+1} - X^i\|_F = 0. \tag{A.16}$$

The convergence of  $(X^i)_{i \in \mathcal{N}_k}$  to  $X^*$  therefore implies the convergence of  $(X^{i+1})_{i \in \mathcal{N}_k}$  to the same point. As  $(X^{i+1})_{i \in \mathcal{N}_k} = (X^i)_{i \in \mathcal{N}_{k+1}}$  with  $(q_i)_{i \in \mathcal{N}_{k+1}} = (k+1)_i$ , the argument showing that  $X^*$  is a fixed point of node  $k$  can be applied to node  $k+1$ , and inductively to node  $k+l$  for any  $l$ .  $X^*$  is therefore a fixed point of Algorithm 2.1.  $\square$

<sup>a</sup>The result is established for non-generalized eigenspaces, but it can be straightforwardly extended to generalized eigenspaces by performing a change of variables similar to (2.42) in order to turn the GEVD into an equivalent eigenvalue decomposition.

<sup>b</sup>The Hausdorff distance  $d_H(\mathcal{V}, \mathcal{W})$  between two sets  $\mathcal{V}$  and  $\mathcal{W}$  is defined as  $\max\{\sup_{X \in \mathcal{V}} \inf_{Y \in \mathcal{W}} \|X - Y\|, \sup_{Y \in \mathcal{V}} \inf_{X \in \mathcal{W}} \|X - Y\|\}$ .  $d_H(\mathcal{V}, \mathcal{W}) = \varepsilon$  implies that for any point in  $\mathcal{V}$  we can find a point in  $\mathcal{W}$  at a distance at most  $\varepsilon$  and vice-versa.

We can now finally prove Theorem 2.2. As  $\mathcal{D}$  is compact,  $(X^i)_{i \in \mathbb{N}}$  must, by definition of an accumulation point, converge to the (possibly infinite) set of its accumulation points. As a consequence, if  $(X^i)_{i \in \mathbb{N}}$  is not a convergent sequence, it will eventually oscillate between points arbitrarily close to accumulation points. As from Lemma A.2, the accumulation points of  $(X^i)_{i \in \mathbb{N}}$  are fixed points of the algorithm, and hence from Theorem 2.1, stationary points of the problem, the sequence has a finite number of accumulation points, separated by a finite and fixed distance. As a consequence, it cannot be that  $\|X^{i+1} - X^i\|_F$  converges to 0 and there must be some index set  $\mathcal{N}$  such that

$$\lim_{i \rightarrow \infty, i \in \mathcal{N}} \|X^{i+1} - X^i\|_F > 0. \tag{A.17}$$

Therefore there must also be some other index set  $\mathcal{N}_k \subseteq \mathcal{N}$  such that  $(q_i)_{i \in \mathcal{N}_k} = (k)_i$ . (A.17) thus contradicts (A.16) and  $(X^i)_{i \in \mathbb{N}}$  must therefore be a convergent sequence.  $\square$

### A.3 Proof of Theorem 2.3

Since  $X^*$  is not a global minimizer and since (2.14) has no local minima<sup>1</sup>, every neighborhood  $\mathcal{V} \subseteq \mathcal{D}$  around  $X^*$  contains a continuum of points  $U \in \mathcal{V}$  for which  $f(U) < f(X^*)$ . Now take any point  $U$  in  $\mathcal{U}$ . Since  $(f(X^i))_{i \in \mathbb{N}}$  is monotonically decreasing, setting  $X^0 = U$  will result in a sequence  $(X^i)_{i \in \mathbb{N}}$  that remains at a finite distance from  $X^*$ . Therefore,  $X^*$  cannot be a stable accumulation point.  $\square$

### A.4 Proof of Equation (2.40)

Let  $U_{kl} \triangleq R_{\mathbf{y}_k}^{\frac{1}{2}} X_{kl}$ . (2.3) becomes

$$\begin{bmatrix} O & P_{\mathbf{y}_k \mathbf{y}_l} \\ P_{\mathbf{y}_l \mathbf{y}_k} & O \end{bmatrix} \begin{bmatrix} U_{kl} \\ U_{lk} \end{bmatrix} = \begin{bmatrix} U_{kl} \\ U_{lk} \end{bmatrix} \Lambda_{kl} \quad (\text{A.18})$$

As described in [88], the full eigenvalue decomposition of this matrix consists in the eigenvectors and diagonal matrix

$$\begin{bmatrix} U_{kl} & U_{kl} \\ U_{lk} & -U_{lk} \end{bmatrix} \text{ and } \begin{bmatrix} \Lambda_{kl} & O \\ O & -\Lambda_{kl} \end{bmatrix} \quad (\text{A.19})$$

As the eigenvectors matrix is unitary, we have

$$2 \| \Lambda_{kl} \|_F^2 = 2 \text{Tr} (\Lambda_{kl}^2) = \| P_{\mathbf{y}_k \mathbf{y}_l} \|_F^2 + \| P_{\mathbf{y}_l \mathbf{y}_k} \|_F^2 = 2 \| P_{\mathbf{y}_k \mathbf{y}_l} \|_F^2 \quad (\text{A.20})$$

which in conjunction with (2.4) gives  $\| P_{\mathbf{y}_k \mathbf{y}_l} \|_F^2 = \Theta_{kl}$ .  $\square$

---

<sup>1</sup>Matrices spanning the generalized eigenspaces of  $(R_D, R_{\mathbf{y}\mathbf{y}})$  corresponding to the  $Q$  largest or smallest GEVLs, are global maximizers and minimizers of (2.14). Matrices spanning the generalized eigenspaces corresponding to any other combination of GEVLs are saddle points.

## A.5 Proof of Theorem 2.4

From the Cauchy-Schwartz inequality, we have

$$\left\| \hat{P}_{\mathbf{y}_k \mathbf{y}_l} - P_{\mathbf{y}_k \mathbf{y}_l} \right\|_F^2 \geq \quad (\text{A.21})$$

$$\left\| \hat{P}_{\mathbf{y}_k \mathbf{y}_l} \right\|_F^2 + \left\| P_{\mathbf{y}_k \mathbf{y}_l} \right\|_F^2 - 2 \left\| \hat{P}_{\mathbf{y}_k \mathbf{y}_l} \right\|_F \left\| P_{\mathbf{y}_k \mathbf{y}_l} \right\|_F \quad (\text{A.22})$$

$$\geq \Theta_{kl} + \hat{\Theta}_{kl}^Q - 2\sqrt{\hat{\Theta}_{kl}^Q} \sqrt{\Theta_{kl}} \quad (\text{A.23})$$

$$\geq \left( \sqrt{\hat{\Theta}_{kl}^Q} - \sqrt{\Theta_{kl}} \right)^2 \quad (\text{A.24})$$

As  $\gamma^2$  corresponds to the  $Q$ -largest squared eigenvalue of  $P_{\mathbf{x}\mathbf{x}} - I$ , the low-rank approximation error  $\left\| \hat{P}_{\mathbf{y}\mathbf{y}} - P_{\mathbf{y}\mathbf{y}} \right\|_F^2$  is bounded by  $(M - Q)\gamma^2$  and therefore so is

$$\sum_{k,l \in \mathcal{K}, k \neq l} \left( \sqrt{\hat{\Theta}_{kl}^Q} - \sqrt{\Theta_{kl}} \right)^2.$$

□

## Appendix B

---

# Convergence of Gauss-Seidel

---

This chapter contains a full proof of the convergence of the non-linear Gauss-Seidel algorithm, described by Algorithm 3.1.

For notational convenience, we define

$$L(X) \triangleq \varphi(X^T \mathbf{y}, X^T B), \quad (\text{B.1})$$

and denote the update in equation (3.6) by the map

$$\mathcal{F}_q^{GS}(X) \triangleq \underset{X_q \in \mathbb{R}^{M_q \times Q}}{\operatorname{argmin}} \varphi \left( X_q^T \mathbf{y}_q + \sum_{k \neq q} X_k^T \mathbf{y}_k, X_q^T B_q + \sum_{k \neq q} X_k B_k \right), \quad (\text{B.2})$$

which allows us to summarize the full procedure as

$$X^{i+1} = \mathcal{F}_q^{GS}(X^i). \quad (\text{GS})$$

We show convergence under the following assumptions:

**Assumption B.1.** *L has compact sub-level sets.*

**Assumption B.2.** *The local update step (3.6) has a unique solution.*

**Assumption B.3.** *There is some  $S < \infty$  such that it is guaranteed that a node becomes updating node at least every  $S$  iterations.*

The proof outline is as follows. We first show that the updates (GS) of the algorithm result in sequences whose accumulation points are fixed points of the algorithm, i.e., they are solutions of the local problems at every node. Finally, we show that such fixed points are stationary points of the global problems.

Let us first state without proof the monotonic decrease of the objective.

**Proposition B.1** (Monotonic decrease). *The sequence of objective values  $(f(X^i))_{i \in \mathbb{N}}$  is monotonically decreasing.*

This fact should be obvious from the definition of the procedure.

We now show that in virtue of the compact sub-level sets of  $L$ , any accumulation point of a sequence generated by the algorithm must be a fixed point of the local problem of at least one node.

**Lemma B.1** (Accumulation points are local fixed points). *Let  $(\bar{X}, q)$  be the accumulation points of two sequences  $(X^i)_{i \in \mathbb{N}}, (q^i)_{i \in \mathbb{N}}$  related by (GS). Then*

$$\bar{X} = \mathcal{F}_q^{GS}(\bar{X}), \quad (\text{B.3})$$

*i.e.  $\bar{X}$  is a fixed point of node  $q$ .*

*Proof.* From the monotonic decrease stated in Proposition B.1 and the fact that  $L$  is bounded below, it must be that  $(L(X^i))_{i \in \mathbb{N}}$  converges to some  $\bar{L}$ . From the continuity of  $L$ , we therefore have  $L(\bar{X}) = \bar{L}$ , and hence also

$$\lim_{i \in \mathcal{I} \rightarrow \infty} L(\mathcal{F}_{q^i}^{GS}(X^i)) = \lim_{i \in \mathcal{I} \rightarrow \infty} L(X^{i+1}) = \bar{L}. \quad (\text{B.4})$$

As  $\mathcal{K}$  is a finite set and  $q$  is an accumulation point of  $(q^i)_{i \in \mathbb{N}}$ , we can find an index set  $\mathcal{I}$  such that  $q^i = q$  for any  $i \in \mathcal{I}$ .

From Lemma F.4, we have

$$\lim_{i \in \mathcal{I} \rightarrow \infty} L(\mathcal{F}_q^{GS}(X^i)) = \lim_{i \in \mathcal{I} \rightarrow \infty} \min_{X_q} L([X_1^{iT}, \dots, X_q^T, \dots, X_K^{iT}]^T) \quad (\text{B.5})$$

$$\leq \min_{X_q} L([\bar{X}_1^T, \dots, X_q^T, \dots, \bar{X}_K^T]^T), \quad (\text{B.6})$$

which combined with (B.4) yields

$$L(\bar{X}) \leq \min_{X_q} L([\bar{X}_1^T, \dots, X_q^T, \dots, \bar{X}_K^T]^T). \quad (\text{B.7})$$

But by definition of the minimum we also have

$$\min_{X_q} L([\bar{X}_1^T, \dots, X_q^T, \dots, \bar{X}_K^T]^T) \leq L(\bar{X}), \quad (\text{B.8})$$

and hence

$$\min_{X_q} L([\bar{X}_1^T, \dots, X_q^T, \dots, \bar{X}_K^T]^T) = L(\bar{X}), \quad (\text{B.9})$$

which implies that  $\bar{X}$  is a local minimizer at node  $q$  and therefore  $\bar{X} = \mathcal{F}_q^{GS}(\bar{X})$ .  $\square$

Using the previous lemma, we can show that the iterates  $X^i$  become arbitrarily close from one another. This is the closest we get from convergence without stronger assumptions.

**Proposition B.2** (Convergence of the residuals). *Assume that Assumptions B.1 and B.2 are satisfied. Let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by (GS). then*

$$\lim_{i \rightarrow \infty} \|X^{i+1} - X^i\| = 0. \quad (\text{B.10})$$

*Proof.* Assume that (B.10) is not true, then there must be some index set  $\mathcal{I}$  such that

$$\lim_{i \in \mathcal{I} \rightarrow \infty} \|X^{i+1} - X^i\| > 0 \quad (\text{B.11})$$

(As the contrary would imply that every convergent subsequence converges to 0, and thus that the full sequence converges to 0). Note that such a sequence always exists, because we assumed that  $X^i$  lived in a compact

space, and the  $\|\cdot\|$  is a continuous function, and as takes compact sets to compact sets [70].

Consider the sequence  $(X^i, X^{i+1}, q^i)_{i \in \mathcal{I}}$ . As the cartesian product of compact spaces is also compact, we can find an index set  $\mathcal{I}' \subseteq \mathcal{I}$  such that  $(X^i, X^{i+1}, q^i)_{i \in \mathcal{I}'}$  converges to some  $(\bar{X}, \bar{X}^{+1}, q)$ . From Lemma B.1,

$$\min_{X_q} L([\bar{X}_1^T, \dots, X_q^T, \dots, \bar{X}_K^T]^T) = L(\bar{X}), \tag{B.12}$$

and from the continuity of  $L$ ,  $L(\bar{X}) = L(\bar{X}^{+1})$ , hence

$$\min_{X_q} L([\bar{X}_1^T, \dots, X_q^T, \dots, \bar{X}_K^T]^T) = L(\bar{X}^{+1}). \tag{B.13}$$

As  $\bar{X}_k = \bar{X}_k^{+1}$  for each  $k \neq q$ ,  $\bar{X}^{+1}$  is a possible solution of (B.13). But because from Assumption B.2 the local problems have a unique solution, we conclude that

$$\bar{X} = \bar{X}^{+1}. \tag{B.14}$$

Therefore

$$\lim_{i \in \mathcal{I}' \rightarrow \infty} \|X^{i+1} - X^i\| = \|\bar{X} - \bar{X}^{+1}\| = 0, \tag{B.15}$$

which contradicts (B.11) and (B.10) must therefore be true.  $\square$

Using Lemma B.1 and Proposition B.2, we can show that an accumulation point must be a fixed point of the full procedure, rather than only of a single local problem.

**Theorem B.1** (Accumulation points are fixed points). *Under Assumptions B.1–B.3, the accumulation points of a sequence generated by (GS) are fixed points of the full procedure.*

*Proof.* Let  $\bar{X}$  be any accumulation of the the sequence generated by (GS). In particular we can find an index set  $\mathcal{I}$  such that  $q^i = q$  for any  $i \in \mathcal{I}$ . We construct a second index set  $\mathcal{J}$  as follows: for every index  $i$  in  $\mathcal{I}$  add an index  $j$  to  $\mathcal{J}$  such that  $|j - i| < S$  and  $q^j = q'$  for some arbitrary  $q'$  (such a sequence can always be constructed thanks to Assumption B.3). Let  $((i^l, j^l))_{i \in \mathbb{N}}$  denote the ascending sequence of indices in  $\mathcal{I}$  and  $\mathcal{J}$ . Finally, let  $\mathcal{M}$  be an an index set such that the sequence  $((i^l, j^l))_{i \in \mathcal{M}}$

is such that

$$\lim_{l \in \mathcal{M} \rightarrow \infty} X^{j^l} = \bar{X}', \quad (\text{B.16})$$

for some  $\bar{X}'$ . Again, such an index set always exists thanks to Assumption B.1. From Lemma B.1, we have

$$\bar{X} = \mathcal{F}_q^{GS}(\bar{X}), \quad (\text{B.17})$$

and

$$\bar{X}' = \mathcal{F}_q^{GS}(\bar{X}'). \quad (\text{B.18})$$

From Proposition B.2 and the triangle inequality

$$\lim_{i \rightarrow \infty} \|X^{i+s} - X^i\| = 0, \quad (\text{B.19})$$

for any  $s > 0$ . Therefore,

$$\lim_{l \in \mathcal{M} \rightarrow \infty} \|X^{i^l} - X^{j^l}\| = 0, \quad (\text{B.20})$$

as  $j^l - i^l$  is bounded, and it must be that  $\bar{X} = \bar{X}'$ . From (B.17) and (B.18),  $\bar{X}$  is therefore a fixed point of both node  $q$  and node  $q'$ . As node  $q'$  was chosen arbitrarily,  $\bar{X}$  is a fixed point for any node in  $\mathcal{K}$ , and hence a fixed point of the full procedure defined by (GS). □

Now that we have shown that the algorithm converges to its set of fixed points, it remains to show that those fixed points are optimal in some sense. We show that fixed points are stationary points with the following proposition.

**Proposition B.3** (Optimality of fixed points). *Let  $X^* = \mathcal{F}_q^{GS}(X^*)$  for every  $q \in \mathcal{K}$ . Then  $X^*$  is a stationary point of problem (3.1), i.e.*

$$\nabla L(X^*) = 0. \quad (\text{B.21})$$

*Proof.*  $X^* = \mathcal{F}_q^{GS}(X^*)$  implies that

$$X_q^* \in \underset{X_q}{\operatorname{argmin}} L([X_1^{*T}, \dots, X_q^T, \dots, X_K^{*T}]^T), \quad (\text{B.22})$$

and hence  $\nabla_q L(X^*) = 0$ . As this is true for any  $q$ , we have  $\nabla L(X^*) = 0$ .  $\square$

We can now state the main result.

**Theorem B.2.** *Assume that Assumptions B.1–B.3 are satisfied and let  $(X^i)_{i \in \mathbb{N}}$  be a sequence generated by (GS). Then  $(X^i)_{i \in \mathbb{N}}$  converges to the set of stationary points of  $L$ , i.e.*

$$\text{Acc}(X^i)_{i \in \mathbb{N}} \subseteq \text{Stat } L. \quad (\text{B.23})$$

*Furthermore, if  $\text{Stat } L$  is a finite set,  $(X^i)_{i \in \mathbb{N}}$  converges to a single point.*

*Proof.* From Theorem B.1, all accumulation points of  $(X^i)_{i \in \mathbb{N}}$  are fixed points of (GS). From Proposition B.3, such fixed points are also stationary points of the global problem (3.1). Finally, the last point regarding convergence to a single point is a direct consequence of Lemma F.3 and Proposition B.2.  $\square$

# Appendix C

---

## Proofs of Chapter 4

---

### C.1 Continuity of quadratic orthogonality constraints

$\mathbb{E}\{\mathbf{y}\mathbf{y}^T\}$  being positive definite, it can be uniquely factored as  $\mathbb{E}\{\mathbf{y}\mathbf{y}^T\} \triangleq RR^T$  [125], then (4.34) can be expressed as,

$$\begin{aligned}\mathcal{X}(\mathbf{y}) &= \{X \mid Y^T Y = I, Y = R^T X\} \\ &= \{X = R^{-T} Y \mid Y^T Y = I\},\end{aligned}\tag{C.1}$$

As the Cholesky decomposition and matrix inverse are continuous on the set of positive definite matrices [113], [126],  $\mathcal{X}(\cdot)$  is a continuous map (by composition).

### C.2 Proof of Lemma 4.1

The map  $\mathcal{R}_q$  is merely the composition of several continuous maps. We denote  $C_q^D(X, \cdot)$  the linear map which given the set

$$\mathcal{D} \triangleq \{D_{j,k} \mid j \in \mathcal{J}_E^k \cup \mathcal{J}_I^k, k \in \mathcal{K}\}\tag{C.2}$$

produces

$$\begin{aligned} \tilde{\mathcal{D}} \triangleq \{X_k^T D_{j,k} \mid j \in \mathcal{J}_E^k \cup \mathcal{J}_I^k, k \in \mathcal{K} \setminus \{q\}\} \\ \cup \{D_{j,q} \mid j \in \mathcal{J}_E^q \cup \mathcal{J}_I^q\}. \end{aligned} \quad (\text{C.3})$$

Indeed,  $U \in \mathcal{R}_q(X)$  implies that  $\exists \tilde{X} : U = C_q(X)\tilde{X}$  and  $U \in \mathcal{X}(\mathbf{y}, D)$ . Hence

$$C_q(X)\tilde{X} \in \mathcal{X}(\mathbf{y}, D)$$

By associativity and the definition (4.32) of  $\mathcal{X}(\cdot, \cdot)$ <sup>1</sup>,

$$\begin{aligned} C_q(X)\tilde{X} \in \mathcal{X}(\mathbf{y}, \mathcal{D}) &\Leftrightarrow \tilde{X} \in \mathcal{X}(C_q(X)^T \mathbf{y}, C_q^D(X, \mathcal{D})) \\ &\Leftrightarrow U \in C_q(X)\mathcal{X}(C_q(X)^T \mathbf{y}, C_q^D(X, \mathcal{D})) \\ &\Leftrightarrow U \in \mathcal{R}_q(X) \end{aligned} \quad (\text{C.4})$$

where the left-multiplication of a set by a matrix must be understood as an element-wise operation. Therefore

$$\mathcal{R}_q(X) = C_q(X)\mathcal{X}(C_q(X)^T \mathbf{y}, C_q^D(X, \mathcal{D})), \quad (\text{C.5})$$

which is continuous by the composition of  $\mathcal{X}$ ,  $C_q$  ( $C_q$  being trivially continuous as its blocks are simply sub-blocks of its argument, see (3.37)) and  $X \mapsto C_q^D(X, \mathcal{D})$  [70, Theorem 17.23].

### C.3 Proof of Lemma 4.2

Upper semicontinuity can be stated as follows: let  $(X^i, m_q(X^i))_i$  be any converging sequence (note that this sequence is not necessarily an NS-DASF sequence). Let its limit be some  $(\bar{X}, \bar{m})$ , then

$$\bar{m} \leq m_q(\bar{X}). \quad (\text{C.6})$$

We now attempt to prove this implication. From Lemma 4.1, for any  $U \in \mathcal{R}_q(\bar{X})$ , there exists<sup>2</sup> a sequence  $(U^i)_{i \in \mathbb{N}}$  converging to  $U$  and such that  $U^i \in \mathcal{R}_q(X^i)$ . From (4.37) it follows that<sup>3</sup>  $m_q(\bar{X}) \in L(\mathcal{R}_q(\bar{X}))$ , and we can therefore select some  $U \in \mathcal{R}_q(\bar{X})$  such that  $L(U) = m_q(\bar{X})$ . By definition of  $m_q$ , the sequence  $U^i \in \mathcal{R}_q(X^i)$  converging to  $U$  is such that  $m_q(X^i) \leq L(U^i)$ . Taking the limit of both sides yields  $\bar{m} \leq L(U) = m_q(\bar{X})$ , completing the proof.

<sup>1</sup>This is with a slight abuse of notation. The maps  $\mathcal{X}(\mathbf{y}, \mathcal{D})$  and  $\mathcal{X}(C_q(X)^T \mathbf{y}, C_q(X)^T D)$  are technically different maps as their input spaces have different dimensions.

<sup>2</sup>This is the definition of set inner semicontinuity [68], [70], [71].

<sup>3</sup>The application of a function to a set denotes the image of that set under the function.

## C.4 Proof of Lemma 4.4

Under the qualification,  $X$  satisfies for any  $k$

$$N_{\mathcal{X}}(X) \cap \text{null } C_k(X)^T \subseteq \{0\}. \quad (\text{C.7})$$

As  $\text{null } C_k(X)^T$  is by definition the orthogonal complement of  $\text{range } C_k(X) = \mathcal{S}_k(X)$ , we have  $\text{null } C_k(X)^T = N_{\mathcal{S}_k(X)}(X)$  (the normal cone to a linear subspace is its orthogonal complement [72]), and hence (C.7) implies that

$$\forall U \in N_{\mathcal{X}}(X), U' \in N_{\mathcal{S}_k(X)}(X), \quad U' \neq U \quad (\text{C.8})$$

and  $U \neq -U'$  (because  $N_{\mathcal{S}_k(X)}(X)$  is a linear subspace) unless  $U = 0$ , which implies [68, Theorem 6.14] that

$$N_{\mathcal{R}_k(X)}(X) \subseteq N_{\mathcal{X}}(X) + N_{\mathcal{S}_k(X)}(X). \quad (\text{C.9})$$

## C.5 Proof of Proposition 4.5

We give a proof by contradiction. Assume that there is some  $U \in N_{\mathcal{X}}(X)$  such that  $T_k C_k(X)^T U = 0$  and  $U \neq 0$ . By construction,  $U$  can be expressed as a non-null linear combination

$$\sum_{m \in \mathcal{K}} \sum_{j \in \mathcal{J}_E^m \cup \mathcal{A}^m(X)} \lambda_j \nabla \theta_j^m(X) \quad (\text{C.10})$$

where  $\theta_j^m : X \mapsto \vartheta_j^m(X_m)$ . Because  $T_k C_k(X)^T U = 0$ , it must be that for any  $n \in \mathcal{N}_k$

$$\sum_{m \in \mathcal{K}} \sum_{j \in \mathcal{J}_E^m \cup \mathcal{A}^m(X)} \lambda_j \sum_{l \in \mathcal{B}_{nk}} X_l^T \nabla_l \theta_j^m(X) = 0. \quad (\text{C.11})$$

But we have for  $l \neq m$  that  $\nabla_l \theta_j^m(X) = 0$ . Therefore, from (C.11), it must be that for any  $n$ ,

$$\sum_{m \in \mathcal{K}} \sum_{j \in \mathcal{J}_E^m \cup \mathcal{A}^m(X)} \lambda_j \delta_{\mathcal{B}_{nk}}(m) X_m^T \nabla \vartheta_j^m(X_m) = 0, \quad (\text{C.12})$$

where  $\delta_{\mathcal{B}_{nk}}(m)$  is 1 if  $m \in \mathcal{B}_{nk}$ , 0 otherwise. We can equivalently write

$$\sum_{m \in \mathcal{B}_{nk}} \sum_{j \in \mathcal{J}_E^m \cup \mathcal{A}^m(X)} \lambda_j X_m^T \nabla \vartheta_j^m(X_m) = 0, \quad (\text{C.13})$$

which contradicts the linear independence assumption (4.5), unless  $\lambda_j = 0$  for every  $j \in \mathcal{J}_E^m \cup \mathcal{A}^m(X)$  and  $m \in \mathcal{B}_{nk}$ .

## Appendix D

---

# Chambolle-Pock algorithm for the group-lasso problem

---

The Chambolle-Pock algorithm (also known as primal-dual hybrid gradient [116], [118]) can be used to minimize functions of the form

$$f(x) + g(x^T A) \tag{D.1}$$

and consists of a primal and dual update at each iteration:

$$x^{i+1} = \text{prox}_{\alpha f}(x^i - \alpha A u^i) \tag{D.2}$$

$$u^{i+1} = \text{prox}_{\beta g^*}(u^i - \beta A^T(2x^{i+1} - x^i))$$

where  $g^*$  denotes the conjugate function of  $g$ , and  $\alpha$  and  $\beta$  are two freely chosen parameters. Convergence is ensured if  $\alpha\beta \|A\|_2^2 < 1$ .

The generic group-lasso problem associated with (4.62) is defined as

$$\min_x \mathbb{E} \left\{ \|x^T \mathbf{y}(t) - \mathbf{d}(t)\|^2 \right\} + \lambda \sum_k \|x_k^T A_k\|_2, \tag{D.3}$$

with  $\lambda > 0$  and other parameters as previously defined for (4.62). Defining

$$\alpha f(x) \triangleq \frac{\alpha}{\lambda} \mathbb{E} \left\{ \|x^T \mathbf{y}(t) - \mathbf{d}(t)\|^2 \right\}, \tag{D.4}$$

$$g(x) \triangleq \sum_k \|x_k\|_2 \tag{D.5}$$

(we include  $\lambda$  in  $f$  for notational convenience), we have [108]

$$\beta g^*(x) = \sum_k \delta_{\{u \mid \|u\|_2 \leq 1\}}(x_k) \quad (\text{D.6})$$

and hence [108]

$$\text{prox}_{\alpha f}(x) = \left( \frac{2\alpha}{\lambda} R_{\mathbf{y}} + I \right)^{-1} \left( x - \frac{2\alpha}{\lambda} \mathbb{E} \{ \mathbf{y} \mathbf{d}^T \} \right) \quad (\text{D.7})$$

$$[\text{prox}_{\beta g^*}(x)]_k = \frac{x_k}{\max(\|x_k\|_2, 1)} \quad (\text{D.8})$$

where  $[\cdot]_k$  denotes the block associated with  $x_k$ . Substituting the two above equations in (D.2) gives the full optimization procedure to obtain a minimizer of (D.3).

## Appendix E

---

# Common inexact solvers

---

We describe in this section a couple of methods satisfying the requirements set by Definition 5.1. In what follows, we consider methods to minimize a real-valued function  $h : \mathbb{R}^{M \times Q} \mapsto \mathbb{R}$ , whose precise structure (e.g. smoothness, convexity, etc.) will be redefined for every method. We first state a useful definition:

**Definition E.1.** A smooth function  $h : \mathbb{R}^{M \times Q} \mapsto \mathbb{R}$  is said to have  $R$ -Lipschitz gradients if there is some  $R > 0$  such that for any  $X, Y \in \mathbb{R}^{M \times Q}$

$$\|\nabla h(X) - \nabla h(Y)\|_F \leq R \|X - Y\|_F. \quad (\text{E.1})$$

In addition, [108, Descent Lemma], (E.1) also implies that

$$h(Y) \leq h(X) + \langle \nabla h(X), Y - X \rangle_F + \frac{R}{2} \|X - Y\|_F^2, \quad (\text{E.2})$$

where  $\langle A, B \rangle_F \triangleq \text{Tr}(A^T B)$ .

## E.1 Line-search methods: gradient descent and Newton's method

Line search methods are defined by the update rule

$$X^{i+1} = X^i + \mu^i P^i, \quad (\text{E.3})$$

where  $P^i$  is the *search direction* and  $\mu^i$  is an appropriately chosen step-size.

**Gradient descent** In the case of gradient descent, we have  $P^i = -\nabla h(X^i)$  and hence

$$X^{i+1} - X^i = -\mu^i \nabla h(X^i). \quad (\text{E.4})$$

As a consequence, we have

$$\frac{1}{\inf_i \mu^i} \|X^{i+1} - X^i\| \geq \|\nabla h(X^i)\|. \quad (\text{E.5})$$

As  $h$  is smooth,  $\partial h(X^i) = \{\nabla h(X^i)\}$  [68], and condition (C3) of Definition 5.1 is therefore satisfied.

Applying the Descent Lemma (E.2) to  $X^i, X^{i+1}$  we have

$$\begin{aligned} h(X^i) - h(X^{i+1}) & \\ & \geq -\langle \nabla h(X^i), X^{i+1} - X^i \rangle_F - \frac{R}{2} \|X^{i+1} - X^i\|_F^2. \end{aligned} \quad (\text{E.6})$$

From (E.4), (E.6) becomes

$$h(X^i) - h(X^{i+1}) \geq \left( \frac{1}{\mu^i} - \frac{R}{2} \right) \|X^{i+1} - X^i\|_F^2. \quad (\text{E.7})$$

and condition (C2) is thus also satisfied if  $\sup_i \mu^i < \frac{2}{R}$ .

**Newton's method** Newton's step consists in setting  $P^i = -(\nabla^2 h(X^i))^{-1} \nabla h(X^i)$ . Hence, from the update rule (E.3), we have

$$-\frac{1}{\mu^i} \nabla^2 h(X^i)(X^{i+1} - X^i) = \nabla h(X^i). \quad (\text{E.8})$$

Taking the norm on both sides and denoting as  $\lambda_{\max}$  an upper bound on the largest eigenvalue of the Hessian (across all iterations),

$$\frac{\lambda_{\max}}{\inf_i \mu^i} \|X^{i+1} - X^i\| \geq \|\nabla h(X^i)\|, \quad (\text{E.9})$$

and condition (C3) is fulfilled by the same reasoning as for gradient descent.

Taking the inner product of both sides of (E.8) with  $(X^{i+1} - X^i)$  and denoting the Cholesky factorization of the Hessian as  $\nabla^2 f(X^i) \triangleq UU^T$ , we have

$$-\langle \nabla h(X^i), X^{i+1} - X^i \rangle_F = \frac{1}{\mu^i} \|U^T(X^{i+1} - X^i)\|^2. \quad (\text{E.10})$$

Hence

$$-\langle \nabla h(X^i), X^{i+1} - X^i \rangle_F \geq \frac{\lambda_{\min}}{\sup_i \mu^i} \|X^{i+1} - X^i\|^2 \quad (\text{E.11})$$

where  $\lambda_{\min}$  denotes a lower bound on the smallest eigenvalue of the Hessian (across all iterations). Combining the above inequality with (E.6) yields

$$h(X^i) - h(X^{i+1}) \geq \left( \frac{\lambda_{\min}}{\sup_i \mu^i} - \frac{R}{2} \right) \|X^{i+1} - X^i\|_F^2, \quad (\text{E.12})$$

and condition (C2) is satisfied as long as the step-size satisfies  $\sup_i \mu^i < \frac{2\lambda_{\min}}{R}$  (note that this also implies that the Hessian must be positive definite).

In practice, the bounds on the step-size cannot be known apriori, but the same results can be obtained if a backtracking line-search [108], [121], [127] is used to obtain the step-size.

## E.2 Regularized exact solver

Given some previous estimate of the solution  $X^-$ , a regularized exact solver would update the solution as (we drop any iteration index, as the solver only performs a single iteration)

$$X^+ = \underset{X}{\operatorname{argmin}} h(X) + \mu \|X - X^-\|_F^2 \quad (\text{E.13})$$

instead of  $\min_X h(X)$ , where  $\mu > 0$  is a freely chosen parameter. From the optimality of  $X^+$  in (E.13), we have

$$h(X^-) - h(X^+) \geq \mu \|X^+ - X^-\|_F^2, \quad (\text{E.14})$$

and condition (C2) is satisfied. Furthermore, the optimality of  $X^+$  also implies that (using the sum-rule for subgradients [72])

$$0 \in \partial h(X^+) + 2\mu(X^+ - X^-), \quad (\text{E.15})$$

And thus there is  $W \triangleq 2\mu(X^- - X^+) \in \partial h(X^+)$ , and condition (C3) is trivially satisfied.

### E.3 Power method

We will rely on the proof available in [120] that the projected gradient algorithm satisfies conditions (C2) and (C3) of Definition 5.1, even when the constraint set is non-convex. The projected gradient algorithm is defined by

$$X^{i+1} = P_{\mathcal{C}}(X^i - \mu \nabla h(X^i)) \quad (\text{E.16})$$

where  $P_{\mathcal{C}}$  is the projection on some constraint set  $\mathcal{C}$  and  $\mu$  some step-size. We will show that the power method is a particular case of (E.16). The power method finds the eigenvector associated with the largest eigenvalue of some matrix  $A$ . It therefore finds a solution of

$$\min_x -x^T A x \quad \text{s.t.} \quad x^T x = 1. \quad (\text{E.17})$$

The power method's update rule is

$$x^{i+1} = \frac{Ax^i}{\|Ax^i\|}. \quad (\text{E.18})$$

Let us now assume that we solve the following (equivalent) problem using a projected gradient algorithm:

$$\min_x -x^T \frac{(A - I)}{2\mu} x \quad \text{s.t.} \quad x^T x = 1. \quad (\text{E.19})$$

Denoting the objective  $h$ , its gradient is

$$\nabla h(x) = \frac{(I - A)}{\mu} x, \quad (\text{E.20})$$

The projected gradient method with step-size  $\mu$  for this problem is thus

$$x^{i+1} = P_{\mathcal{C}}\left(x^i - \mu \frac{x^i - Ax^i}{\mu}\right) = P_{\mathcal{C}}(Ax^i) \quad (\text{E.21})$$

with  $\mathcal{C}$  denoting in this case the unit ball, which is equivalent to the update rule of the power method (E.18). Note that we skipped the discussion on the condition that the step-size should be smaller than the inverse of the Lipschitz constant of  $\nabla h(x)$  [120], and simply mention that this condition can always be enforced by applying the proper scaling on  $A$ , as this changes neither the solution of (E.17) nor the update rule (E.18).

# Appendix F

---

## Useful lemmas

---

### Convergence to the set of accumulation points

**Lemma F.1.** *Let  $(u^i)_{i \in \mathbb{N}}$  be some sequence in a compact metric space. Then  $\text{Acc}(u^i)_{i \in \mathbb{N}}$  is non-empty and*

$$\lim_{i \rightarrow \infty} \min_{\bar{u} \in \text{Acc}(u^i)_{i \in \mathbb{N}}} \|u^i - \bar{u}\|_F = 0, \quad (\text{F.1})$$

*i.e.  $(u^i)_{i \in \mathbb{N}}$  converges to the non-empty set of its accumulation points.*

*Proof.* This statement is in fact the definition of an accumulation point in hiding. A point

$$\bar{u} \in \text{Acc}(u^i)_{i \in \mathbb{N}}$$

is an accumulation point if and only if [69] there exists some index set  $\mathcal{I}$  such that

$$\lim_{i \in \mathcal{I} \rightarrow \infty} \|u^i - \bar{u}\|_F = 0. \quad (\text{F.2})$$

Such an index set is guaranteed to exist in a compact space. Assume that (F.1) is not true. then there is some  $\varepsilon > 0$  such that

$$\forall N, \exists i > N : \min_{\bar{u}} \|u^i - \bar{u}\| \geq \varepsilon. \quad (\text{F.3})$$

Let

$$\mathcal{I} \triangleq \{i \in \mathbb{N} \mid \min_{\bar{u}} \|u^i - \bar{u}\| \geq \varepsilon\} \quad (\text{F.4})$$

As  $(u^i)_{i \in \mathbb{N}}$  lives in a compact space, we can find an index set  $\mathcal{I}' \subseteq \mathcal{I}$  such that

$$\lim_{i \in \mathcal{I}' \rightarrow \infty} u^i = \bar{u} \quad (\text{F.5})$$

for some  $\bar{u} \in \text{Acc}(u^i)_{i \in \mathbb{N}}$ , but this contradicts the fact that  $\mathcal{I}' \subseteq \mathcal{I}$  and (F.1).  $\square$

## Necessary and sufficient condition for convergence

**Lemma F.2.** *A sequence in a compact metric space converges if and only if it has a single accumulation point.*

*Proof.* Let  $(u^i)_{i \in \mathbb{N}}$  be such a convergent sequence. Then there is some  $\bar{u}$  such that

$$\lim_{i \rightarrow \infty} u^i = \bar{u}, \quad (\text{F.6})$$

and every subsequence trivially converges to  $\bar{u}$  [69], proving the necessity. Now assume that  $(u^i)_{i \in \mathbb{N}}$  has a single accumulation point  $\bar{u}$  and that  $(u^i)_{i \in \mathbb{N}}$  does not converge, then there is some  $\varepsilon > 0$  such that

$$\forall N, \exists i > N : \|u^i - \bar{u}\| \geq \varepsilon. \quad (\text{F.7})$$

Let

$$\mathcal{I} \triangleq \{i \in \mathbb{N} \mid \|u^i - \bar{u}\| \geq \varepsilon\} \quad (\text{F.8})$$

As the sequence lives in a compact space, we can find an index set  $\mathcal{I}' \subseteq \mathcal{I}$  such that

$$\lim_{i \in \mathcal{I}' \rightarrow \infty} u^i = \bar{u}'. \quad (\text{F.9})$$

But as the sequence has a single accumulation point, it must be that  $\bar{u}' = \bar{u}$ , which contradicts the fact that  $\mathcal{I}' \subseteq \mathcal{I}$ , proving the sufficiency.  $\square$

## Set of accumulation points

**Lemma F.3.** *Let  $(u^i)_{i \in \mathbb{N}}$  be a sequence in a compact metric space such that*

$$\lim_{i \rightarrow \infty} \|u^i - u^{i+1}\| = 0. \quad (\text{F.10})$$

*Then  $(X^i)_{i \in \mathbb{N}}$  has either an infinite number of accumulation points, or a single one.*

*Proof.* Assume that  $(u^i)_{i \in \mathbb{N}}$  has a finite number of accumulation points, and that accumulation points are separated by a distance of at least  $\delta$  (such a radius exists as the set is finite).

From Lemma F.1, there exists some index  $N$  such that for any  $i > N$ ,  $\exists \bar{u}^i \in \text{Acc}(u^i)_{i \in \mathbb{N}}$  such that

$$\|u^i - \bar{u}^i\| < \frac{\delta}{3}, \quad (\text{F.11})$$

and from (F.10)

$$\|u^i - u^{i+1}\| < \frac{\delta}{3}. \quad (\text{F.12})$$

From the triangle inequality it must be that

$$\|\bar{u}^i - u^{i+1}\| < \frac{2\delta}{3}. \quad (\text{F.13})$$

But we also have

$$\|u^{i+1} - \bar{u}^{i+1}\| < \frac{\delta}{3}, \quad (\text{F.14})$$

and hence, again by the triangle inequality,

$$\|\bar{u}^i - \bar{u}^{i+1}\| < \delta, \quad (\text{F.15})$$

a contradiction unless  $\bar{u}^i = \bar{u}^{i+1}$  for any  $i$ .  $\text{Acc}(u^i)_{i \in \mathbb{N}}$  is therefore a singleton.

□

## Upper semicontinuity of the partial minimum

**Lemma F.4.** *Let  $\mathcal{U}$  and  $\mathcal{V}$  be two metric vector spaces, and  $f : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$  be a continuous map. Then  $u \mapsto \min_v f(u, v)$  is upper-semicontinuous.*

*Proof.* By definition,

$$\forall v', \quad \min_v f(u_i, v) - f(u_i, v') \leq 0. \quad (\text{F.16})$$

As  $\leq$  defines a closed set and  $f$  is continuous, it must be that for any sequence  $u_i \rightarrow \bar{u}$

$$\forall v', \quad \lim_{u_i \rightarrow \bar{u}} \min_v f(u_i, v) - f(\bar{u}, v') \leq 0. \quad (\text{F.17})$$

In particular, if we select  $v' \in \operatorname{argmin}_v f(\bar{u}, v)$ , we have  $f(\bar{u}, v') = \min_v f(\bar{u}, v)$  and therefore

$$\lim_{u_i \rightarrow \bar{u}} \min_v f(u_i, v) \leq \min_v f(\bar{u}, v), \quad (\text{F.18})$$

which is the definition of upper-semicontinuity for  $u \mapsto \min_v f(u, v)$ .  $\square$

## Compactness of the sub-level sets of a composition of functions

**Lemma F.5.** *Let  $\mathcal{U}$  and  $\mathcal{V}$  be two metric spaces. If  $g : \mathcal{V} \mapsto \mathbb{R}$  has compact sub-level sets, and  $f : \mathcal{U} \mapsto \mathcal{V}$  is a continuous bijective map. Then  $g \circ f$  has compact sub-level sets.*

*Proof.* For every sequence  $(v^i)_{i \in \mathbb{N}}$  in  $\mathcal{S}_g^\alpha \triangleq \{v \in \mathcal{V} \mid g(v) \leq \alpha\}$  we can find an index set  $\mathcal{I}$  such that  $(v^i)_{i \in \mathcal{I}}$  is a convergent sequence with limit in  $\mathcal{S}_g^\alpha$ .

Given some sequence  $(u^i)_{i \in \mathbb{N}}$  in  $\mathcal{S}_{g \circ f}^\alpha \triangleq \{u \in \mathcal{U} \mid g(f(u)) \leq \alpha\}$ , we can construct the corresponding sequence  $(f(u^i))_{i \in \mathbb{N}}$  in  $\mathcal{S}_g^\alpha$ . From the above, we can find an index set  $\mathcal{I}$  such that  $(f(u^i))_{i \in \mathcal{I}}$  is a convergent sequence with limit  $\bar{v}$  in  $\mathcal{S}_g^\alpha$ . From [69, Theorem 4.17],  $f^{-1}$  is continuous. Therefore  $(u^i)_{i \in \mathcal{I}}$  is a convergent sequence with limit  $f^{-1}(\bar{v})$  in  $\mathcal{S}_{g \circ f}^\alpha$ .  $\square$

# Bibliography

---

- [1] K. Cao, Y. Liu, G. Meng, and Q. Sun, “An overview on edge computing research”, *IEEE access*, vol. 8, pp. 85 714–85 728, 2020.
- [2] S. Abbasian Dehkordi, K. Farajzadeh, J. Rezazadeh, R. Farahbakhsh, K. Sandrasegaran, and M. Abbasian Dehkordi, “A survey on data aggregation techniques in IoT sensor networks”, *Wireless Networks*, vol. 26, no. 2, pp. 1243–1263, 2020.
- [3] S. Boubiche, D. E. Boubiche, A. Bilami, and H. Toral-Cruz, “Big data challenges and data aggregation strategies in wireless sensor networks”, *IEEE access*, vol. 6, pp. 20 558–20 571, 2018.
- [4] K. Gulati, R. S. K. Boddu, D. Kapila, S. L. Bangare, N. Chandnani, and G. Saravanan, “A review paper on wireless sensor network techniques in internet of things (IoT)”, *Materials Today: Proceedings*, vol. 51, pp. 161–165, 2022.
- [5] V. Raghunathan, S. Ganeriwal, and M. Srivastava, “Emerging techniques for long lived wireless sensor networks”, *IEEE Communications Magazine*, vol. 44, no. 4, pp. 108–114, 2006.
- [6] A. Bertrand, “Distributed signal processing for wireless EEG sensor networks”, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 6, pp. 923–935, 2015.
- [7] A. Hilmani, A. Maizate, and L. Hassouni, “Automated real-time intelligent traffic control system for smart cities using wireless sensor networks”, *Wireless Communications and mobile computing*, vol. 2020, no. 1, p. 8 841 893, 2020.

- [8] M. Abdulkarem, K. Samsudin, F. Z. Rokhani, and M. F. A. Rasid, “Wireless sensor network for structural health monitoring: A contemporary review of technologies, challenges, and future direction”, *Structural health monitoring*, vol. 19, no. 3, pp. 693–735, 2020.
- [9] L. Muduli, D. P. Mishra, and P. K. Jana, “Application of wireless sensor network for environmental monitoring in underground coal mines: A systematic review”, *Journal of Network and Computer Applications*, vol. 106, pp. 48–67, 2018.
- [10] M. S. Jamil, M. A. Jamil, A. Mazhar, A. Ikram, A. Ahmed, and U. Munawar, “Smart environment monitoring system by employing wireless sensor networks on vehicles for pollution free smart cities”, *Procedia Engineering*, vol. 107, pp. 480–484, 2015.
- [11] A. Gazis and E. Katsiri, “A wireless sensor network for underground passages: Remote sensing and wildlife monitoring”, *Engineering reports*, vol. 2, no. 6, e12170, 2020.
- [12] W. Li and S. Kara, “Methodology for monitoring manufacturing environment by using wireless sensor networks (WSN) and the internet of things (IoT)”, *Procedia CIRP*, vol. 61, pp. 323–328, 2017.
- [13] H. M. Jawad, R. Nordin, S. K. Gharghan, A. M. Jawad, and M. Ismail, “Energy-efficient wireless sensor networks for precision agriculture: A review”, *Sensors*, vol. 17, no. 8, p. 1781, 2017.
- [14] R. Khan, I. Ali, M. Zakarya, M. Ahmad, M. Imran, and M. Shoaib, “Technology-assisted decision support system for efficient water utilization: A real-time testbed for irrigation using wireless sensor networks”, *IEEE Access*, vol. 6, pp. 25 686–25 697, 2018.
- [15] R. A. C. Tamayo, M. L. Ibarra, and J. A. G. Macías, “Better crop management with decision support systems based on wireless sensor networks”, in *2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control*, IEEE, 2010, pp. 412–417.
- [16] F. E. Horita, J. P. de Albuquerque, L. C. Degrossi, E. M. Mendiondo, and J. Ueyama, “Development of a spatial decision support system for flood risk management in brazil that combines volunteered geographic information with wireless sensor networks”, *Computers & Geosciences*, vol. 80, pp. 84–94, 2015.
- [17] H.-N. Dai, R. C.-W. Wong, H. Wang, Z. Zheng, and A. V. Vasilakos, “Big data analytics for large-scale wireless networks: Challenges and opportunities”, *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–36, 2019.

- [18] M. Yemini, S. Gil, and A. Goldsmith, “Exploiting local and cloud sensor fusion in intermittently connected sensor networks”, in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, IEEE, 2020, pp. 1–7.
- [19] N. Cao, S. B. Nasir, S. Sen, and A. Raychowdhury, “In-sensor analytics and energy-aware self-optimization in a wireless sensor node”, in *2017 IEEE MTT-s international microwave symposium (IMS)*, IEEE, 2017, pp. 200–203.
- [20] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, “Environmental wireless sensor networks”, *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1903–1917, 2010.
- [21] A. Bertrand, J. Callebaut, and M. Moonen, “Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks”, in *Proc. of the International Workshop on Acoustic Echo and Noise Control (IWAENC)*, 2010.
- [22] O. Elharrouss, N. Almaadeed, and S. Al-Maadeed, “A review of video surveillance systems”, *Journal of Visual Communication and Image Representation*, vol. 77, p. 103116, 2021.
- [23] A. M. Narayanan, P. Patrinos, and A. Bertrand, “Optimal versus approximate channel selection methods for EEG decoding with application to topology-constrained neuro-sensor networks”, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 29, pp. 92–102, 2020.
- [24] A. M. Narayanan, R. Zink, and A. Bertrand, “EEG miniaturization limits for stimulus decoding with EEG sensor networks”, *Journal of Neural Engineering*, vol. 18, no. 5, p. 056042, 2021.
- [25] M. Inggis, H. Griffiths, F. Fioranelli, M. Ritchie, and K. Woodbridge, “Multistatic radar: System requirements and experimental validation”, in *2014 International Radar Conference*, IEEE, 2014, pp. 1–6.
- [26] M. Inggis and A. van der Byl, “NeXtRAD and RHINO radar: Harnessing the herd for networked radar”, in *2014 International Radar Conference*, IEEE, 2014, pp. 1–5.
- [27] A. Djupdal, M. Sjalander, M. Jahre, and S. Aunet, *Minimizing the energy usage of tiny risc-v cores*, [https://carrv.github.io/2023/papers/CARRV2023\\_paper\\_2\\_Djupdal.pdf](https://carrv.github.io/2023/papers/CARRV2023_paper_2_Djupdal.pdf), 2023.
- [28] M. Siekkinen, M. Hienkari, J. K. Nurminen, and J. Nieminen, “How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4”, in *2012 IEEE wireless communications and networking conference workshops (WCNCW)*, IEEE, 2012, pp. 232–237.
- [29] S. Haykin and K. R. Liu, *Handbook on array processing and sensor networks*. John Wiley & Sons, 2010, vol. 63.

- [30] H. Hotelling, “Analysis of a complex of statistical variables into principal components.”, *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.
- [31] K. Fukunaga and W. L. Koontz, “Representation of random processes using the finite karhunen-loeve expansion”, *Information and Control*, vol. 16, no. 1, pp. 85–101, 1970.
- [32] N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. The MIT press, 1949.
- [33] M. S. Bartlett, “The statistical significance of canonical correlations”, *Biometrika*, vol. 32, no. 1, pp. 29–37, 1941.
- [34] P. Comon, “Independent component analysis, a new concept?”, *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.
- [35] S. Markovich-Golan, A. Bertrand, M. Moonen, and S. Gannot, “Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks”, *Signal Processing*, vol. 107, pp. 4–20, 2015.
- [36] B. D. Van Veen and K. M. Buckley, “Beamforming: A versatile approach to spatial filtering”, *IEEE assp magazine*, vol. 5, no. 2, pp. 4–24, 1988.
- [37] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2011.
- [38] K. P. Murphy, *Probabilistic machine learning: an introduction*. MIT press, 2022.
- [39] G. James, D. Witten, T. Hastie, R. Tibshirani, *et al.*, *An introduction to statistical learning*. Springer, 2013, vol. 112.
- [40] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers”, *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.
- [41] C. G. Lopes and A. H. Sayed, “Diffusion least-mean squares over adaptive networks: Formulation and performance analysis”, *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.
- [42] A. H. Sayed *et al.*, “Adaptation, learning, and optimization over networks”, *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.
- [43] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions”, *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

- [44] C. I. Kanatsoulis, X. Fu, N. D. Sidiropoulos, and M. Hong, “Structured SUMCOR multiview canonical correlation analysis for large-scale data”, *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 306–319, 2018. DOI: [10.1109/tsp.2018.2878544](https://doi.org/10.1109/tsp.2018.2878544).
- [45] A. Scaglione, R. Pagliari, and H. Krim, “The decentralized estimation of the sample covariance”, in *2008 42nd Asilomar Conference on Signals, Systems and Computers*, IEEE, 2008, pp. 1722–1726.
- [46] L. Li, A. Scaglione, and J. H. Manton, “Distributed principal subspace estimation in wireless sensor networks”, *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 725–738, 2011.
- [47] Y. Zeng and R. C. Hendriks, “Distributed delay and sum beamformer for speech enhancement via randomized gossip”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 260–273, 2013.
- [48] X. Fu, K. Huang, E. E. Papalexakis, *et al.*, “Efficient and distributed generalized canonical correlation analysis for big multiview data”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2304–2318, 2018.
- [49] A. Bertrand and M. Moonen, “Distributed canonical correlation analysis in wireless sensor networks with application to distributed blind source separation”, *IEEE Transactions on Signal Processing*, vol. 63, no. 18, pp. 4800–4813, 2015.
- [50] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, “Distributed basis pursuit”, *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1942–1956, 2011.
- [51] J. Zhang, A. I. Koutrouvelis, R. Heusdens, and R. C. Hendriks, “Distributed rate-constrained LCMV beamforming”, *IEEE Signal Processing Letters*, vol. 26, no. 5, pp. 675–679, 2019.
- [52] **C. Hovine** and A. Bertrand, “MAXVAR-based distributed correlation estimation in a wireless sensor network”, *IEEE Transactions on Signal Processing*, vol. 70, pp. 5533–5548, 2022.
- [53] A. Bertrand and M. Moonen, “Distributed adaptive node-specific signal estimation in fully connected sensor networks – part I: Sequential node updating”, *IEEE Trans. Signal Processing*, vol. 58, no. 10, pp. 5277–5291, Oct. 2010, ISSN: 1053-587X. DOI: [10.1109/TSP.2010.2052612](https://doi.org/10.1109/TSP.2010.2052612). [Online]. Available: [http://homes.esat.kuleuven.be/%7Eabertran/papers\\_website/09-65.html](http://homes.esat.kuleuven.be/%7Eabertran/papers_website/09-65.html).

- [54] A. Bertrand and M. Moonen, “Distributed adaptive node-specific signal estimation in fully connected sensor networks—part ii: Simultaneous and asynchronous node updating”, *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5292–5306, 2010.
- [55] A. Bertrand and M. Moonen, “Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA”, *Signal Processing*, vol. 104, pp. 120–135, 2014.
- [56] A. Bertrand and M. Moonen, “Distributed adaptive generalized eigenvector estimation of a sensor signal covariance matrix pair in a fully connected sensor network”, *Signal Processing*, vol. 106, pp. 209–214, 2015.
- [57] A. Hassani, A. Bertrand, and M. Moonen, “GEVD-based low-rank approximation for distributed adaptive node-specific signal estimation in wireless sensor networks”, *IEEE Transactions on Signal Processing*, vol. 64, no. 10, pp. 2557–2572, 2015.
- [58] C. A. Musluoglu and A. Bertrand, “A unified algorithmic framework for distributed adaptive signal and feature fusion problems—part I: Algorithm derivation”, *IEEE Transactions on Signal Processing*, 2023.
- [59] **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems in wireless sensor networks”, *IEEE Transactions on Signal Processing*, vol. 72, pp. 4682–4697, 2024.
- [60] **C. Hovine** and A. Bertrand, “Distributed adaptive spatial filtering with inexact local solvers”, *arXiv preprint arXiv:2405.03277*, 2024.
- [61] **C. Hovine** and A. Bertrand, “Distributed MAXVAR: Identifying common signal components across the nodes of a sensor network”, in *2021 29th European Signal Processing Conference (EUSIPCO)*, IEEE, 2021, pp. 2159–2163.
- [62] C. A. Musluoglu, **C. Hovine**, and A. Bertrand, “A unified algorithmic framework for distributed adaptive signal and feature fusion problems — part II: Convergence properties”, *IEEE Transactions on Signal Processing*, vol. 71, pp. 1879–1894, 2023.
- [63] **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems”, in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.
- [64] M. Ali, S. Roy, U. Saxena, T. Sharma, A. Raghunathan, and K. Roy, “Compute-in-memory technologies and architectures for deep learning workloads”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1615–1630, 2022.

- [65] Y. Zheng, H. Yang, Y. Shu, Y. Jia, and Z. Huang, "Optimizing off-chip memory access for deep neural network accelerator", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 4, pp. 2316–2320, 2022.
- [66] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)", in *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, IEEE, 2014, pp. 10–14.
- [67] F. H. Clarke, *Optimization and nonsmooth analysis*. SIAM, 1990.
- [68] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*. Springer Science & Business Media, 2009, vol. 317.
- [69] W. Rudin *et al.*, *Principles of mathematical analysis*. McGraw-hill New York, 1976, vol. 3.
- [70] D. Charalambos and B. Aliprantis, *Infinite Dimensional Analysis: A Hitchhiker's Guide*. Springer-Verlag Berlin and Heidelberg GmbH & Company KG, 2013.
- [71] J.-P. Aubin and H. Frankowska, *Set-valued analysis*. Springer Science & Business Media, 2009.
- [72] J. O. Royset and R. J. Wets, *An Optimization Primer*. Springer, 2021.
- [73] W. Karush, "Minima of functions of several variables with inequalities as side constraints", *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.
- [74] H. Kuhn and A. Tucker, "Nonlinear programming", in *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics*, 1951, pp. 481–492.
- [75] N. Vlatjic and D. Xia, "Wireless sensor networks: To cluster or not to cluster?", in *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, IEEE, 2006, 9–pp.
- [76] D. Xia and N. Vlatjic, "Near-optimal node clustering in wireless sensor networks for environment monitoring", in *21st International Conference on Advanced Information Networking and Applications (AINA'07)*, IEEE, 2007, pp. 632–641.
- [77] X. Sun and Q. Cheng, "On subspace distance", in *International Conference Image Analysis and Recognition*, Springer, 2006, pp. 81–89.
- [78] G. H. Golub and C. F. Van Loan, *Matrix computations*, 4th ed. John Hopkins University press, 2013.
- [79] P. Horst, *Generalized canonical correlations and their application to experimental data*. Journal of clinical psychology, 1961. DOI: [10.1002/1097-4679\(196110\)17:4<331::aid-jclp2270170402>3.0.co;2-d](https://doi.org/10.1002/1097-4679(196110)17:4<331::aid-jclp2270170402>3.0.co;2-d).

- [80] J. D. Carroll, “Generalization of canonical correlation analysis to three or more sets of variables”, in *Proceedings of the 76th annual convention of the American Psychological Association*, Washington, DC, vol. 3, 1968, pp. 227–228. DOI: [10.1037/e473742008-115](https://doi.org/10.1037/e473742008-115).
- [81] J. R. Kettenring, “Canonical analysis of several sets of variables”, *Biometrika*, vol. 58, no. 3, pp. 433–451, 1971. DOI: [10.1093/biomet/58.3.433](https://doi.org/10.1093/biomet/58.3.433).
- [82] M. Sørensen, C. I. Kanatsoulis, and N. D. Sidiropoulos, “Generalized canonical correlation analysis: A subspace intersection approach”, *IEEE Transactions on Signal Processing*, vol. 69, pp. 2452–2467, 2021.
- [83] J. Chen and I. D. Schizas, “Distributed information-based clustering of heterogeneous sensor data”, *Signal Processing*, vol. 126, pp. 35–51, 2016.
- [84] P. Stoica, K. M. Wong, and Q. Wu, “On a nonparametric detection method for array signal processing in correlated noise fields”, *IEEE Transactions on Signal Processing*, vol. 44, no. 4, pp. 1030–1032, 1996.
- [85] Y. Song, P. J. Schreier, D. Ramírez, and T. Hasiija, “Canonical correlation analysis of high-dimensional data with very small sample support”, *Signal Processing*, vol. 128, pp. 449–458, 2016.
- [86] N. Asendorf and R. R. Nadakuditi, “Improved detection of correlated signals in low-rank-plus-noise type data sets using informative canonical correlation analysis (icca)”, *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3451–3467, 2017.
- [87] H. Hotelling, “Relations between two sets of variates”, in *Breakthroughs in statistics*, Springer, 1992, pp. 162–190.
- [88] T. De Bie, N. Cristianini, and R. Rosipal, “Eigenproblems in pattern recognition”, in *Handbook of Geometric Computing*, Springer, 2005, pp. 129–167.
- [89] M. O’Searcoid, *Metric spaces*. Springer Science & Business Media, 2006.
- [90] G. Zuccon, L. A. Azzopardi, and C. Van Rijsbergen, “Semantic spaces: Measuring the distance between different subspaces”, in *International Symposium on Quantum Interaction*, Springer, 2009, pp. 225–236.
- [91] K. Ye and L.-H. Lim, “Schubert varieties and distances between subspaces of different dimensions”, *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 3, pp. 1176–1197, 2016.
- [92] S. E. Schaeffer, “Graph clustering”, *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007.
- [93] J. Via, I. Santamaría, and J. Pérez, “Canonical correlation analysis (cca) algorithms for multiple data sets: Application to blind simo equalization”, in *2005 13th European Signal Processing Conference*, IEEE, 2005, pp. 1–4.

- [94] J. Vía, I. Santamaría, and J. Pérez, “A learning algorithm for adaptive canonical correlation analysis of several data sets”, *Neural Networks*, vol. 20, no. 1, pp. 139–152, 2007.
- [95] B. Ghogh, F. Karray, and M. Crowley, “Eigenvalue and generalized eigenvalue problems: Tutorial”, *arXiv preprint arXiv:1903.11240*, 2019.
- [96] A. H. Sameh and J. A. Wisniewski, “A trace minimization algorithm for the generalized eigenvalue problem”, *SIAM Journal on Numerical Analysis*, vol. 19, no. 6, pp. 1243–1259, 1982.
- [97] G. Pandurangan, P. Robinson, M. Scquizzato, *et al.*, “The distributed minimum spanning tree problem”, *Bulletin of EATCS*, vol. 2, no. 125, 2018.
- [98] Z. Bao, J. Hu, G. Pan, and W. Zhou, “Canonical correlation coefficients of high-dimensional normal vectors: Finite rank case”, *arXiv preprint arXiv:1407.7194*, 2014.
- [99] D. P. Bertsekas, “Nonlinear programming”, *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.
- [100] S. V. Vaseghi, “Wiener filters”, *Advanced Signal Processing and Digital Noise Reduction*, pp. 140–163, 1996.
- [101] M. P. Becker, I. Yang, and K. Lange, “Em algorithms without missing data”, *Statistical Methods in Medical Research*, vol. 6, no. 1, pp. 38–54, 1997.
- [102] W. J. Heiser, “Convergent computation by iterative majorization”, *Recent advances in descriptive multivariate analysis*, pp. 157–189, 1995.
- [103] K. Lange, D. R. Hunter, and I. Yang, “Optimization transfer using surrogate objective functions”, *Journal of computational and graphical statistics*, vol. 9, no. 1, pp. 1–20, 2000.
- [104] Y. Sun, P. Babu, and D. P. Palomar, “Majorization-minimization algorithms in signal processing, communications, and machine learning”, *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2016.
- [105] G. Marjanovic, M. O. Ulfarsson, and A. O. Hero, “Mist: L<sub>0</sub> sparse linear regression with momentum”, in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 3551–3555.
- [106] X.-T. Yuan and T. Zhang, “Truncated power method for sparse eigenvalue problems.”, *Journal of Machine Learning Research*, vol. 14, no. 4, 2013.
- [107] C. Berge, *Topological Spaces: including a treatment of multi-valued functions, vector spaces, and convexity*. Courier Corporation, 1997.

- [108] A. Beck, *First-order methods in optimization*. SIAM, 2017.
- [109] Y. Yang, X. Guan, Q.-S. Jia, L. Yu, B. Xu, and C. J. Spanos, “A survey of admm variants for distributed optimization: Problems, algorithms and features”, *arXiv preprint arXiv:2208.03700*, 2022.
- [110] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, “Trace ratio vs. ratio trace for dimensionality reduction”, in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–8.
- [111] M. Sørensen, C. I. Kanatsoulis, and N. D. Sidiropoulos, “Generalized canonical correlation analysis: A subspace intersection approach”, *IEEE Transactions on Signal Processing*, vol. 69, pp. 2452–2467, 2021. DOI: [10.1109/TSP.2021.3061218](https://doi.org/10.1109/TSP.2021.3061218).
- [112] T. Strypsteen and A. Bertrand, “Bandwidth-efficient distributed neural network architectures with application to neuro-sensor networks”, *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 2, pp. 933–943, 2022.
- [113] V. Rakocevic, “On continuity of the Moore-Penrose and Drazin inverses”, *Matematički Vesnik*, vol. 49, pp. 163–172, 1997.
- [114] G. Forchini, “A note on the continuity of projection matrices with application to the asymptotic distribution of quadratic forms”, *Applicationes Mathematicae*, vol. 1, no. 32, pp. 51–55, 2005.
- [115] 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.
- [116] 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.
- [117] C. A. Musluoglu, M. Moonen, and A. Bertrand, “Improved tracking for distributed signal fusion optimization in a fully-connected wireless sensor network”, in *2022 30th European Signal Processing Conference (EUSIPCO)*, IEEE, 2022, pp. 1836–1840.
- [118] E. K. Ryu and W. Yin, *Large-scale convex optimization: algorithms & analyses via monotone operators*. Cambridge University Press, 2022.
- [119] A. Themelis, “Proximal algorithms for structured nonconvex optimization”, 2018.
- [120] H. Attouch, J. Bolte, and B. F. Svaiter, “Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods”, *Mathematical Programming*, vol. 137, no. 1, pp. 91–129, 2013.

- [121] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.
- [122] E. Anderson, Z. Bai, C. Bischof, *et al.*, *LAPACK users' guide*. SIAM, 1999.
- [123] W. Kozłowski, "The power method for the generalized eigenvalue problem", *Mathematica Applicanda*, vol. 21, no. 35, 1992.
- [124] T. Kato, *Perturbation theory for linear operators*. Springer Science & Business Media, 2013, vol. 132.
- [125] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.
- [126] M. Schatzman, *Numerical analysis: a mathematical introduction*. Oxford University Press on Demand, 2002.
- [127] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives", *Pacific Journal of mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

# List of publications

---

## Articles in internationally reviewed journals

- \* **C. Hovine** and A. Bertrand, “MAXVAR-based distributed correlation estimation in a wireless sensor network”, *IEEE Transactions on Signal Processing*, vol. 70, pp. 5533–5548, 2022
- \* C. A. Musluoglu, **C. Hovine**, and A. Bertrand, “A unified algorithmic framework for distributed adaptive signal and feature fusion problems — part II: Convergence properties”, *IEEE Transactions on Signal Processing*, vol. 71, pp. 1879–1894, 2023
- \* **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems in wireless sensor networks”, *IEEE Transactions on Signal Processing*, vol. 72, pp. 4682–4697, 2024

## Articles in review

- \* **C. Hovine** and A. Bertrand, “Distributed adaptive spatial filtering with inexact local solvers”, *arXiv preprint arXiv:2405.03277*, 2024

## Articles in proceedings of international conferences

- \* **C. Hovine** and A. Bertrand, “Distributed MAXVAR: Identifying common signal components across the nodes of a sensor network”, in *2021*

*29th European Signal Processing Conference (EUSIPCO)*, IEEE, 2021, pp. 2159–2163

- \* **C. Hovine** and A. Bertrand, “A distributed adaptive algorithm for non-smooth spatial filtering problems”, in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5

# Acknowledgments

---

Charles Hovine is with the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics and with the Leuven.AI institute for Artificial Intelligence at KU Leuven, Leuven 3001, Belgium.

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 and No. 101138304) and from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme. Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or ERC.

## Use of generative AI statement

“I did not use generative AI assistance tools during the research/writing process of my thesis. The text/code/images in this thesis are my own (unless otherwise specified) and generative AI has only been used in accordance with the KU Leuven guidelines and appropriate references have been added. I have reviewed and edited the content as needed and I take full responsibility for the content of the thesis.”