KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Distributed Spatial Filtering in Wireless Sensor Networks

Cem Ates Musluoglu

Supervisor:
Prof. dr. ir. A. Bertrand

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

November 2023

Distributed Spatial Filtering in Wireless Sensor Networks

Cem Ates MUSLUOGLU

Examination committee:
Prof. dr. A. Bultheel, chair
Prof. dr. ir. A. Bertrand, supervisor
Prof. dr. ir. M. Moonen
Prof. dr. ir. S. Pollin
Prof. dr. ir. T. van Waterschoot
Prof. dr. ir. G. Leus
(TU Delft)

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

November 2023

Preface

The first words of this thesis were typed a few months ago in Rhodes, with the view of the magnificent sea reflecting the bright sunlight before my eyes... This is the poetic way I had imagined starting my thesis with before going to a conference in Rhodes, but the truth is that the first words were written a few days after I came back, at home, at a very late hour. Yet it is not whether I had a sea view or not that made this thesis possible, but the people who were there for me throughout my journey that started four years ago.

I want to thank first my promotor Prof. Alexander Bertrand, without whom this work would not have been possible. Your advice, feedback, support and encouragement have been so valuable to me, I have learned a lot from you. Your support also extended outside of research, I remember discussing with you about the splitting of Daft Punk (I was quite sad that day), or when you notified me about Dave Clarke playing in Leuven. Whenever I had a question, you always found the time, whether it would be for intricate parts of a mathematical proof or for the times I asked if you could proofread my "letter to the editor" before submitting a paper. Your mentorship and guidance always motivated me to overcome any obstacle I encountered in my research. I am so happy and grateful to have been part of your group, thank you Alex!

I would also like to thank Prof. Marc Moonen and Prof. Sofie Pollin for following my research throughout these years, as well as the members of my examination committee Prof. Toon van Waterschoot, Prof. Geert Leus and Prof. Adhemar Bultheel.

As this thesis is about "space" (spatial filters), I want to take you on a small trip to various locations (in space). Let's first enter the ESAT building, take the stairs, and meet the research group I was lucky to be a member of. The content of this text might not immediately reflect this, but I was part of the *Biomed* group working on biomedical data processing during my PhD studies (still, I would like to refer readers to pages 1, 2, 7, 10 and 139 for mentions of

biomedical applications). What an amazing group that brought joy and fun to my PhD life! Angeliki, I remember one of our earliest discussions after having met, about a certain rapper. Who would have thought that two years later, I would be able to hold a conversation in Greek with you, right? You showed me that it is not only perfectly possible to build great friendships while pursuing a PhD, but that it is actually a must! Thank you for all the discussions, the laughs, the trips, your support, your friendship. When are we having another cool trip *opnieuw*? Christos, you were the first person to invite me to your place after I moved to Leuven and did not know many people. You have helped me a lot in this journey! Thank you for supporting me to get my driver's license, do you think I will ever get it? And please don't be upset by my picture in front of the Olympiakos stadium, I still like PAOK too. Luis, you opened the way for me to become a real DJ, I will have my first performance soon! I still have some learning to do though. I really want to try once the restaurants you suggested in Düsseldorf, but first let's go to the Christmas market? Miguel, at some point I will join the gym with you (terms and conditions apply). Let me know if you find anything interesting in Cinema Zed! Thank you for being the chef in all the BBQs! Konstantinos, what a cool night it was when we went to watch a Greek movie in Brussels! I learned during that movie that we also share the same word for "glass". We should go again for an after work at the Luxembourg Square, this time I will eat beforehand though. Elisabeth, I had so much fun at each one of your concerts! The funny videos you have sent while I was writing my thesis always cheered me up! I am really wondering how that inside joke "Honey!" appeared, do you remember? Thomas, our padel teacher/dungeon master, thank you for carrying with you your bags filled with board games for us to play! Have you played Mascarade? Iustina, I remember our discussions about electronic music, when are you going to start listening to techno? Guido, I hope we can go skiing soon! It's always fun having discussions with you! Simon, I am still surprised by the way you eat kiwis, I might try it once. Our trip to the island next to Helsinki was so fun! Thank you for showing us the ways of science communication and for all your feedback! When are you going to New Zealand? Raphaël, I had so much fun repeating over and over again that inside joke where I barge into your office! Thank you for the various sweets from Bretagne! Joran, the jam session you invited us to that turned out to be a karaoke event was an unforgettable evening in parts of Leuven I am not used to going a lot. Next time we also sing though! Christof, I will always remember the fondue nights that we did whenever you came to Leuven, you also were always on time for the ESAT parties! Nick, I promise I will bring you back your book, and we can have a discussion about it after I read a summary online. Charles, I find the contrast in our office very funny, one of the tidiest and organized desks in the building (yours) next to one of the most chaotic ones (mine). Yuan, I really enjoyed all our discussions ranging from academic books

to video games! Nicolas, I am glad to have switched to your car during our trip in Rhodes, where I could finally choose the songs to put on during the drive. Jingwei, I heard that you are one of the best in badminton and look forward to playing a game with you. Thank you Pooya, Helene, Bavo, Stefan, Stijn, Nico, and all the new members of the group who will continue making Biomed a great place to work. And of course, I would like to thank Prof. Maarten De Vos and Prof. Sabine Van Huffel, who together with Alex, have built this amazing team. Aldona, John, Ida, Elsy, Maarten, Wim thank you for all your help in our daily lives at the office!

The Biomed group has changed a lot since I started here, so let's move now in time (and not space) to meet the previous members who welcomed me in such a warm way. Abhi, thank you for always answering my random questions. It was funny that you thought I was a big fan of Liverpool when we met (but actually I keep switching teams I like). Carolina, I really want a bowling rematch! It's always so fun talking to you, I wish our time in Biomed had overlapped more. Amalia, I have finally ordered a bike! And I am moving to a new place (please don't worry it's not too small)! Mario, I still have a perfect memory of your voice calling us for soup at lunch. Omer, even if you didn't stay long in Leuven, I have so many memories with you, do you remember the train museum? John, from what you described Colombia and Mexico look amazing, I hope to go very soon! Jasper, you were the one who first brought me to Downtown Jack, a place I would revisit from time to time. Tim, your jokes made the office a happier place! When can I come to Maastricht? Jonathan, I am looking forward to seeing you next time I pass by Lausanne! Thank you Matthias, Andrea, Max!

I am very lucky to have also met many PhD researchers from various other groups, whether at the coffee corner or the corridors, in classes or Paleis 12. Kostas, I can have the most absurd discussions so easily with you, but also very serious ones, the range of topics we discuss is impressively large! I am also amazed at your love of *giraffes*. Paul, I had so much fun at every one of our techno nights out, going to various cities together with Fabio, Elias and Matthias. Thank you guys! Where do we go next? Manu, I have so many nice memories of the times we went out in Leuven and Brussels! Teo, thank you for all the rides to Louvain-la-Neuve! Maybe one time I can join one of your trips to Romania? Gaëlle, I remember how much fun I had at the Halloween party you invited me to! Next time I will also dress up! Rémy, let me know when is the next concert you are going to! Pourya, you have amazing dance moves! As I discovered during the ESAT parties and conferences we have been together. Amir, I look forward to playing more racket sports with you, from tennis to badminton! Jesper, thank you for showing me that just standing there and waiting for the ball to arrive is not how padel is supposed to be played. And thank you Mohit, Santiago, Lennart, Manos, Katy, Bram, Renzi, Elisa, Robbe,

Julian, Puya, Randy, Thomas!

Let's now cross the street after passing by the Arenberg Castle to reach the imec building! A big thank you for all the nights out and the amazing dinners! Aris, I am really proud to have shown you an ice cream place in Paris that ended up being one of your favorites (right?)! The next time we go there, you do all the talking in French though, in fact I should have written this *en français*. But first, when do we play basketball? Fırat, seninle spontane buluşmalarımız her zaman çok keyifliydi, özellikle beş dakika içerisinde birbirimizi ikna edip Rave Rebels'a bilet aldığımıza hâlâ inanamıyorum! Niki, your energy always brought me so much joy! I remember dancing in the streets of Leuven after returning from the Balkan Trafik festival since we could not have enough. Athina, I am amazed at how good you are at playing the various games of the Leuven fair, I want a basketball rematch! Hikmet, seninle hamburgercide oturup dertleştiğimiz akşamı hiç unutmayacağım. Seninle sohbet etmek hep çok güzel! Maider, thank you for hosting us so many times! Every time you prepared delicious food, and especially that paella! Siddik, Seyed, Sarah, Mahmoud thank you for all the dinners and BBQs!

We are now moving a bit further away! Lausanne, Brussels, Paris, New York, Ankara and many more! Jan, c'était vraiment génial de te revoir plusieurs fois ces dernières années ! Tu te souviens de la fois où on a raté le dernier train pour Leuven, et on avait dû prendre le bus avec une petite escale à Kortenberg au milieu de la nuit ? Jean, peu après m'être installé à Leuven, tu étais venu me rendre visite et je voulais t'emmener voir l'Atomium et les alentours (c'était la journée sans voitures). Je me souviens de notre déception quand on a vu qu'il n'y avait rien à manger, et pas grand chose à faire non plus, retour au centre ville ! Fouco, j'ai encore les captures d'écrans de nos partages de localisation qui nous permettait de voir qui arriverait en premier au KFC à Valence. On se voit bientôt en Bretagne ! J'éspère que j'aurai mon permis de conduire d'ici là (probablement pas). Tarik, les vacances à Carqueiranne étaient incroyables, quand est-ce qu'on refait ça ? Je me souviens de notre discussion à Paris, quelques jours avant le nouvel an en 2021, du fait que le mot japonais "asahi" pourrait très bien passer pour un mot turc. Alex, je trouve vraiment drôle que la dernière fois que je suis venu à Lausanne, on ne s'est pas croisé les deux derniers jours (sur quatre au total), alors que je restais chez toi ! Quentin, quand est-ce que tu repasses par Leuven ? On pourrait se retrouver à Paris aussi pour changer. Micka, tu te souviens comment on avait eu du mal à se retrouver près d'IKEA avant de partir à Carqueiranne ? "T'es où ?" Momo, on était vraiment à ça de se voir en Belgique ! Il faut qu'on s'organise ! Ekin, beni Paris'te ağırladığın için çok teşekkür ederim! 2022 yılbaşı kutlaması benim için o kadar zevkli ve ilginçti ki, uzun zamandır mühendislerin çoğunluk olmadığı bir partiye gitmemiştim. Yeni çalışmalarını görmeyi sabırsızlıkla bekliyorum! Arda, doktoram boyunca

seninle arada sırada yaptığımız telefon konuşmaları hep çok keyifliydi, Leuven, New York, Ankara veya başka bir yerde en yakın zamanda görüşürüz umarım! Güney, Emircan, sizinle Brüksel'deki buluşmalarımız her zaman için doktoramın çok güzel anıları arasında olacak. Yiğit, Jules, Noyan, bu son birkaç yıl boyunca sizlerle çok sık bir araya gelememiş olsak da, her görüşmemizde hiç kopmamışız gibi sohbetimizin devam etmesinden çok mutluyum. Emily, Aigli, Rafaella, Yusuf abi, Yekbun abla, Zetta, Tassos, Canan abla, thank you very much!

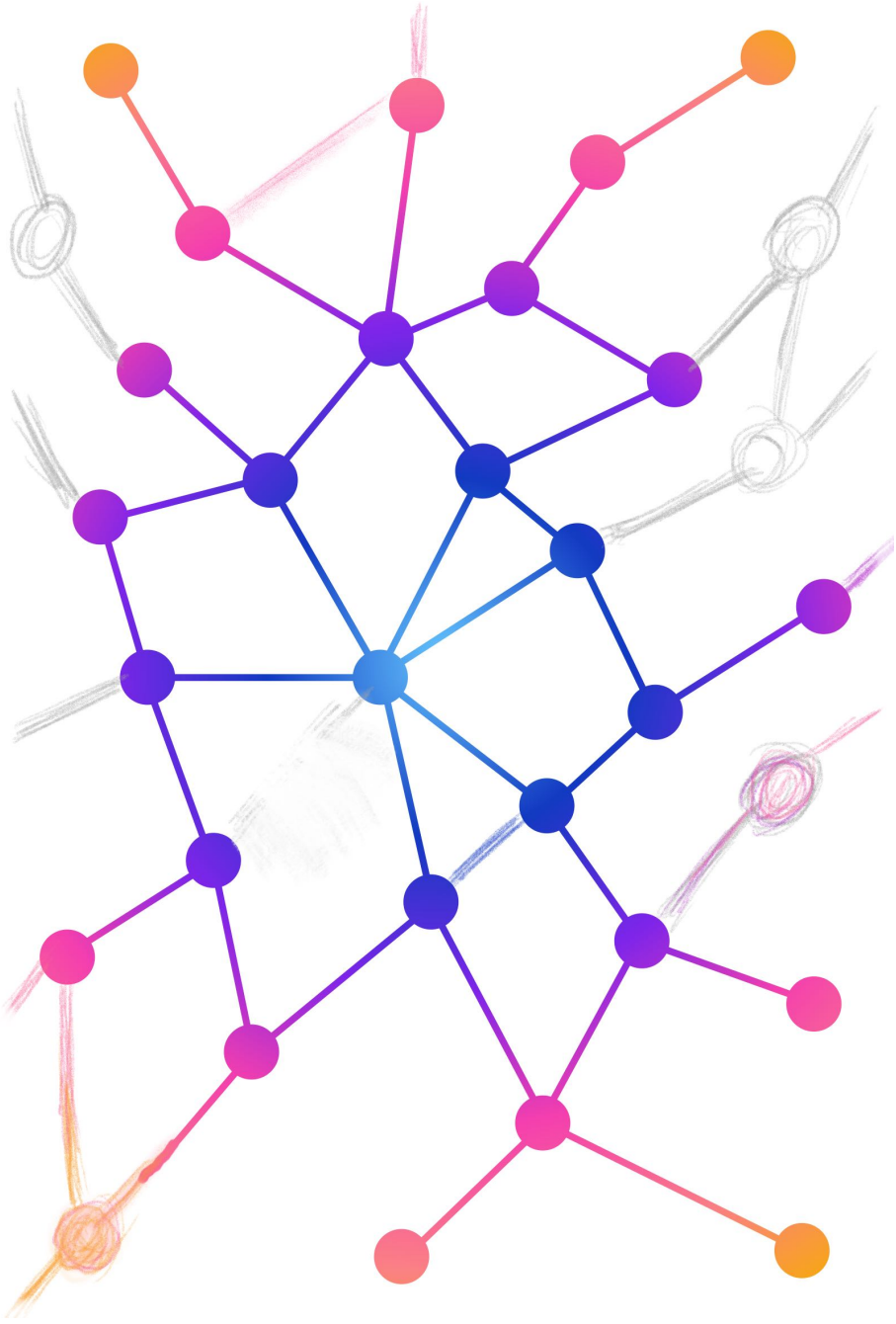
Şimdi ise Türkiye'ye bir yolculuk yapalım, ve bu noktaya gelmemi sağlayan ailemi ziyaret edelim. Anne, bana inancını hiçbir zaman eksik etmedin, hayatımdaki büyük kararlarda hep destek oldun. Liseden yeni çıkmış benim Lozan'a yerleşmeme yardımcı olduğun zamanı daha dün gibi hatırlıyorum. Aradan dokuz yıl geçmiş. Her şey için çok teşekkür ederim, senin sayende buralara kadar gelebildim, sana ne kadar teşekkür etsem azdır! Geçtiğimiz yaz bana Türkiye'ye gelmem için aldığın uçak bileti için de ayrıca teşekkürler (üç buçuk yıllık gelmeme rekorumun kırılma zamanı gelmişti). Anneanne, bana o kadar çok şey öğrettin ki, öğütlerin bana her zaman yol gösterici oldu. Hayat derslerinden resime kadar, bütün emeklerine çok teşekkür ederim! Bir de yemek yapmayı senin kadar sevebilsem keşke. Baba, her Türkiye'ye gelişimde sizinle yaptığımız geziler hep çok güzel anı olarak aklımda. Bir sefer belki dalışlarınıza ben de katılırım. Kariyer tavsiyeleriniz ve desteğiniz için çok teşekkür ederim! Teyze, enişte, sizlerle olan beraberliklerimiz benim için çok değerli. Bahçenizde meyve topladığımız gün ne kadar güzeldi. Buraya geldiğinizde ben de size (marketten) çok güzel meyve getireceğim. Tavsiyeleriniz için çok teşekkür ederim! Can, seninle rastgele mesajlaşmalarımıza bayılıyorum, hep mutlu ediyor beni! Senin beni ziyaret ettiğinde Dinant'daki Noel marketine gidişimiz ne kadar zevkliydi, şanslıydık çünkü Noel sonrası hâlâ açık olan nadir marketlerden bir tanesiydi. Desteğin için çok teşekkür ederim, yakında Montreal'de görüşürüz artık. Babaanne, hala, Engin amca, sizinle deniz kenarında balık yediğimiz akşam ne güzeldi, çok sağolun!

Dede, mühendis olmamı görmeyi çok istiyordunuz, keşke görebilseydiniz, hep aklımdasınız.

Thank you all for joining me on this trip!

Cem,

Leuven, November 2023



Abstract

Wireless sensor networks (WSNs) paved the way for accessing data previously unavailable by deploying sensors in various locations in space, each collecting local measurements of a target source signal. By exploiting the information resulting from the multitude of signals measured at the different sensors of the network, various tasks can be achieved, such as denoising or dimensionality reduction which can in turn be used, e.g., for source localization or detecting seizures from electroencephalography measurements.

Spatial filtering consists of linearly combining the signals measured at each sensor of the network such that the resulting filtered signal is optimal in some sense. This technique is widely used in biomedical signal processing, wireless communication, and acoustics, among other fields. In spatial filtering tasks, the aim is to exploit the correlation between the signals of all sensors in the network, therefore requiring access to all signals collected at the nodes. Due to the high energy cost of transmitting these signals to a central processing unit, many applications require a fully distributed approach to solve spatial filtering problems in order to reduce the energy and bandwidth requirements. Although various distributed signal processing methods already exist, many of them are not designed to solve spatial filtering problems, which makes them either impractical or not usable in this setting. On the other hand, existing methods for distributed spatial filtering are tailored for specific problems. The aim of this thesis is therefore to provide a generic framework for designing distributed algorithms for such spatial filtering problems.

In the first part, we derive the steps of the proposed Distributed Adaptive Signal Fusion (DASF) framework, which allows us to design adaptive and distributed algorithms to solve spatial filtering problems in a WSN context. The framework can be used to solve a large family of optimization problems including many commonly used spatial filtering criteria such as minimum mean square error, principal component analysis, trace ratio optimization, minimum variance beamforming, and canonical correlation analysis, among others. We provide a

technical analysis of the convergence properties of the proposed method and show that convergence to an optimal solution is achieved in most practical scenarios while also discussing solutions for the very contrived cases where convergence is not guaranteed.

In the second part, the focus is on improving the DASF framework and extending it to larger problem families. More specifically, we describe an approach for improving the adaptivity properties of the framework while keeping the amount of additional data transmission required for this low. We also extend the DASF framework to node-specific problems where each node has a different task to solve. Finally, a computationally efficient variant of the DASF framework is provided for fractional programs, significantly reducing the computational burden at each node of the network.

The DASF framework and its extensions presented in this thesis therefore fill the gap in the existing literature on distributed spatial filtering problems for which traditional distributed signal processing methods are usually not suited. The framework also unifies existing distributed algorithms for specific spatial filtering problems without having to design a tailored approach for each instance.

To summarize, this thesis provides a generic unified framework to solve spatial filtering problems in an adaptive and distributed fashion, in order to cope with the energy and bandwidth limitations of wireless sensor networks. The proposed distributed algorithms converge to the optimal solution of the problems of interest under mild conditions generally satisfied in practice as demonstrated in various simulations.

Beknopte samenvatting

Draadloze sensornetwerken (WSNs) maakten de weg vrij voor toegang tot gegevens die voorheen onbeschikbaar waren door sensoren op verschillende locaties in de ruimte in te zetten, die elk lokale metingen van een doelbronsignaal verzamelen. Door gebruik te maken van de informatie die voortkomt uit de veelheid aan signalen die bij de verschillende sensoren van het netwerk worden opgemeten, kunnen verschillende taken worden uitgevoerd, zoals ruisonderdrukking of dimensionaliteitsreductie, die op hun beurt gebruikt kunnen worden voor bronlokalisatie of de detectie van epileptische aanvallen m.b.v. elektro-encefalografie.

Spatiale filtering komt overeen met het lineair combineren van de signalen die bij elke sensor van het netwerk worden opgemeten, zodat het resulterende gefilterde signaal op een of andere manier optimaal is. Deze techniek wordt veel gebruikt in onder andere biomedische signaalverwerking, draadloze communicatie en akoestiek. Bij spatiale filtertaken is het doel om de correlatie tussen de signalen van alle sensoren in het netwerk te benutten. Hiervoor is dus toegang nodig tot alle signalen die op de knooppunten worden verzameld. Vanwege de hoge energiekosten van het verzenden van deze signalen naar een centrale verwerkingseenheid, vereisen veel toepassingen een volledig gedistribueerde aanpak om problemen met spatiale filtering op te lossen met het oog op het verminderen van de energie- en bandbreedtevereisten. Hoewel er reeds verschillende gedistribueerde signaalverwerkingsmethoden bestaan, zijn veel ervan niet ontworpen om problemen met spatiale filtering op te lossen, waardoor ze onpraktisch ofwel onbruikbaar zijn in deze context. Aan de andere kant zijn bestaande methoden voor gedistribueerde spatiale filtering op maat gemaakt voor specifieke problemen. Het doel van dit proefschrift is daarom om een generiek raamwerk te bieden voor het ontwerpen van gedistribueerde algoritmen voor dergelijke spatiale filterproblemen.

In het eerste deel leiden we de stappen af van het voorgestelde gedistribueerde adaptieve signaalfusie (DASF)-raamwerk, dat het mogelijk maakt om adaptieve

en gedistribueerde algoritmen te ontwerpen om spatiale filterproblemen in een WSN-context op te lossen. Het raamwerk kan worden gebruikt om een grote familie van optimalisatieproblemen op te lossen, waaronder veel veelgebruikte spatiale filtercriteria zoals de minimale gemiddelde kwadratische fout, hoofdcomponentenanalyse, de optimalisatie van de verhouding van het spoor van twee matrices, straalvorming met minimale variantie en canonieke correlatieanalyse. We ontwikkelen een technische analyse van de convergentie-eigenschappen van de voorgestelde methode en laten zien dat convergentie naar een optimale oplossing wordt bereikt in de meeste praktische scenario's, terwijl we ook oplossingen bespreken voor de zeer vergezochte gevallen waarin convergentie niet gegarandeerd is.

In het tweede deel ligt de focus op het verbeteren van het DASF-raamwerk en het uitbreiden ervan naar grotere families van problemen. Meer specifiek beschrijven we een aanpak om de adaptieve eigenschappen van het raamwerk te verbeteren en tegelijkertijd de hoeveelheid extra datatransmissie die hiervoor nodig is laag te houden. We breiden het DASF-framework ook uit naar knooppuntspecifieke problemen waarbij elk knooppunt een andere taak heeft om op te lossen. Ten slotte wordt een computationeel efficiënte variant van het DASF-raamwerk ontwikkeld voor fractionele problemen, waardoor de rekenlast op elk knooppunt van het netwerk aanzienlijk wordt verminderd.

Het DASF-framework en de uitbreidingen ervan die in dit proefschrift worden gepresenteerd, vullen daarom de leemte op in de bestaande literatuur over gedistribueerde spatiale filterproblemen waarvoor traditionele gedistribueerde signaalverwerkingsmethoden doorgaans niet geschikt zijn. Het raamwerk verenigt ook bestaande gedistribueerde algoritmen voor specifieke spatiale filterproblemen zonder dat voor elk geval een aanpak op maat hoeft te worden ontworpen.

Samenvattend biedt dit proefschrift een generiek, uniform raamwerk om spatiale filterproblemen op een adaptieve en gedistribueerde manier op te lossen, om zo om te gaan met de energie- en bandbreedtebeperkingen van draadloze sensornetwerken. De voorgestelde gedistribueerde algoritmen convergeren naar de optimale oplossing van relevante problemen onder milde omstandigheden waar in de praktijk doorgaans aan wordt voldaan, zoals aangetoond in verschillende simulaties.

Contents

Abstract vii

Beknopte samenvatting ix

Contents xi

List of Acronyms xvii

List of Symbols xix

List of Figures xxi

List of Tables xxiii

1 Introduction 1
- 1.1 Spatial filtering and signal fusion . . . 2
- 1.2 Wireless sensor networks and distributed processing . . . 7
- 1.3 Distributed spatial filtering and signal fusion . . . 10
- 1.4 Research objectives and overview . . . 15
- 1.5 Mathematical notation . . . 18

I The distributed adaptive signal fusion (DASF) framework 21

2 The distributed adaptive signal fusion framework: Algorithm derivation 23
- 2.1 Introduction . . . 25
- 2.2 Problem description . . . 26
  - 2.2.1 Scope of signal fusion optimization problems . . . 29
  - 2.2.2 Adaptivity and approximation in practical settings . . . 30

2.2.3 General assumptions 32
2.3 Distributed adaptive signal fusion in fully-connected networks (FC-DASF) 34
2.3.1 Algorithm derivation 35
2.3.2 Link between the central and local problems 40
2.3.3 Multiple signals, deterministic terms and variables . . . 44
2.4 Topology-independent DASF (TI-DASF) 45
2.4.1 Star topologies 45
2.4.2 Tree topologies 47
2.4.3 General connected graphs 50
2.5 Examples and simulations 53
2.5.1 Quadratically constrained quadratic problem 56
2.5.2 The trace ratio optimization problem 58
2.5.3 Quadratic problem with a spherical constraint 59
2.5.4 DASF in a tracking problem 60
2.6 Conclusion 62

3 The distributed adaptive signal fusion framework: Convergence properties 63
3.1 Introduction 65
3.2 Review of the DASF framework 65
3.2.1 Problem description and assumptions 66
3.2.2 The DASF algorithm 68
3.3 Convergence and optimality 72
3.3.1 Convergence of the objective 74
3.3.2 Technical conditions for stationarity of fixed points . . . 75
3.3.3 Technical conditions for convergence 79
3.3.4 Convergence to stationary points and global minima . . 81
3.4 Selected examples 83
3.4.1 Least squares / minimum mean square error and ridge regression 84
3.4.2 Linearly constrained minimum variance 86
3.4.3 Generalized eigenvalue decomposition and principal component analysis 88
3.4.4 Trace ratio optimization 90
3.4.5 Canonical correlation analysis 91
3.5 Fixes in case of violations of conditions 92
3.5.1 Avoiding violations of Condition 1b 92
3.5.2 Avoiding violations of Condition 2 92
3.6 Conclusion 93
Appendices 94
3.A Fixed points are KKT points under Condition 1a 94
3.B Fixed points are KKT points under Condition 1b 96

3.C $\text{rank}(\mathbf{H}) = KJ - J$ 98
3.D Accumulation points are fixed points 104
3.E Proof of convergence to a single point 106

II Extensions of the DASF framework 109

4 Improving the tracking performance and adaptivity of the DASF algorithm 111
4.1 Introduction 113
4.2 Efficient communication for improved tracking 113
4.3 Simulations 116
4.3.1 LCMV beamforming 117
4.3.2 EVD problem 118
4.3.3 Results 118
4.4 Conclusion 120

5 The DASF framework for node-specific problems 121
5.1 Introduction 123
5.2 Problem setting 123
5.3 DASF for node-specific problems 126
5.4 Simulations 132
5.4.1 Stationary setting 133
5.4.2 Adaptive setting 135
5.5 Conclusion 135

6 The trace ratio optimization problem and a new distributed algorithm 137
6.1 Introduction 139
6.2 The trace ratio optimization problem 140
6.2.1 Problem definition and relationship to other problems . 140
6.2.2 The trace ratio optimization algorithm 142
6.3 Distributed methods for the TRO problem 144
6.3.1 Adaptive TRO in multi-channel signal processing 145
6.3.2 The DASF algorithm for the TRO problem 145
6.3.3 The distributed TRO algorithm 147
6.3.4 Convergence of the DTRO algorithm 149
6.4 Communication and computational aspects 151
6.4.1 Communication costs 151
6.4.2 Computational complexity 152
6.5 Simulations and discussions 153
6.5.1 Stationary setting 153
6.5.2 Adaptive Setting 155
6.6 Conclusion 156

Appendices . . . 158
6.A Convergence of the objective . . . 158
6.B Stationarity of fixed points . . . 159
6.C Modifications for rank-deficiency . . . 160

7 Distributed adaptive signal fusion for fractional programs 163
7.1 Introduction . . . 165
7.2 Fractional programming review . . . 166
7.3 Problem setting and preliminaries . . . 168
7.3.1 Fractional problems for signal fusion in WSNs . . . 169
7.3.2 Adaptivity and approximations of statistics . . . 171
7.3.3 General assumptions . . . 172
7.4 Cost-efficient DASF for fractional programs . . . 173
7.4.1 Data flow of fractional DASF (F-DASF) . . . 173
7.4.2 Updating scheme of F-DASF . . . 175
7.5 Technical analysis and convergence . . . 177
7.5.1 Preliminaries . . . 178
7.5.2 Convergence of the objective . . . 179
7.5.3 Stationarity of fixed points . . . 180
7.5.4 Convergence of arguments . . . 183
7.5.5 Convergence to stationary points and global minima . . . 184
7.6 Simulations . . . 186
7.6.1 Regularized total least squares . . . 187
7.6.2 A problem with a non-compact constraint set . . . 190
7.7 Conclusion . . . 192
Appendices . . . 193
7.A Fixed points are KKT points under Condition 4a . . . 193

III Conclusion 197

8 Conclusion 198
8.1 Main contributions . . . 198
8.1.1 Part I . . . 198
8.1.2 Part II . . . 199
8.2 Future research directions . . . 200
8.2.1 Further potential extensions . . . 200
8.2.2 Realistic scenario studies and improvements . . . 201

Appendices 204

A Optimization problems with matrix variables 205

A.1 The gradient of a function with matrix variables 206
A.2 Optimization over matrix variables 210

B Examples of optimization problems with matrix variables 213
B.1 Least squares 214
B.2 Linearly constrained minimum variance 215
B.3 Generalized eigenvalue decomposition 216
B.4 Trace ratio optimization 217
B.5 Canonical correlation analysis 218
B.6 Linearly constrained minimum norm 220
B.7 Tikhonov regularization on the minimum norm 221
B.8 Linear objective with a quadratic constraint 223
B.9 Quadratic over linear 224
B.10 Regularized total least squares 226

Bibliography 229

Acknowledgments 245

Curriculum vitae 247

List of Publications 249

List of Acronyms

C	**CCA**	Canonical Correlation Analysis
D	**DASF**	Distributed Adaptive Signal Fusion
	DANSF	Distributed Adaptive Node Specific Signal Fusion
	(D)SFO	(Distributed) Signal Fusion Optimization
E	**EEG**	Electroencephalography
F	**FC**	Fully-Connected
	F-DASF	Fractional Distributed Adaptive Signal Fusion
G	**(G)EVC**	(Generalized) Eigenvector
	(G)EVD	(Generalized) Eigenvalue Decomposition
	(G)EVL	(Generalized) Eigenvalue
K	**KKT**	Karush-Kuhn-Tucker
L	**LCMV**	Linearly Constrained Minimum Variance
	LS	Least Squares
M	**MedSE**	Median Squared Error
	MMSE	Minimum Mean Square Error
P	**PCA**	Principal Component Analysis
R	**RTLS**	Regularized Total Least Squares
T	**TRO**	Trace Ratio Optimization
W	**WSN**	Wireless Sensor Network

List of Symbols

General

$\mathbb{R}$, $\mathbb{R}^m$, $\mathbb{R}^{m\times n}$	Set of real scalars, vectors of size m and matrices of size $m \times n$, respectively
$\mathbb{N}$	Set of non-negative integers
$.^T$	Transpose
$.^{-1}$	Inverse
I_Q	Identity matrix of size $Q \times Q$
$\mathrm{tr}(\cdot)$	Trace operator
$\mathrm{rank}(\cdot)$	Rank operator
$\|\cdot\|$	ℓ_2-norm
$\|\cdot\|_F$	Frobenius norm
$\|\cdot\|$	Absolute value of a scalar/Cardinality of a set
$\triangleq$	Equal by definition
$\Leftrightarrow$	If and only if
$\Rightarrow$	Implies
$\forall$	For all
$\min_{X\in\mathcal{S}} f(X)$	Minimal value of function f over the constraint $X \in \mathcal{S}$
$\operatorname{argmin}_{X\in\mathcal{S}} f(X)$	Argument of function f reaching its minimal value over the constraint $X \in \mathcal{S}$
$\nabla_X f$	Gradient of function f with respect to X
$BlkDiag(\cdot)$	Operator creating a block-diagonal matrix from its arguments
$\mathrm{EVC}_Q(A)$	Matrix containing the Q eigenvectors of A corresponding to its Q largest eigenvalues
$\mathrm{GEVC}_Q(A,B)$	Matrix containing the Q generalized eigenvectors of the pair (A,B) corresponding to its Q largest generalized eigenvalues
$\mathbb{E}[\cdot]$	Expectation operator
$\mathcal{N}(\mu,\sigma^2)$	Normal distribution with mean μ and standard deviation σ
$\mathbb{1}\{\cdot\}$	Indicator function

Frequently used notations

K	Number of nodes in a network
$\mathcal{K}$	Set of nodes in a network
k	Single node in a network
$\mathcal{N}_k$	Set of nodes neighboring node k
q	Updating node
i	Iteration index
X	Variable/Matrix of linear spatial filters
Q	Number of columns of X/Number of spatial filters
$\mathbf{y}_k$	Multi-channel signal measured at node k
X_k	Block of X at node k/Matrix of linear spatial filters of node k
M_k	Number of channels of $\mathbf{y}_k$/Number of rows of X_k
$\mathbf{y}$	Network-wide signal obtained by stacking every $\mathbf{y}_k$
M	Number of channels of $\mathbf{y}$/Number of rows of X
t	Time index
N	Number of time samples measured at each iteration
R	Number of repetition of time sample windows in Chapter 4
$R_{\mathbf{yy}}$	Covariance matrix of $\mathbf{y}$
f, φ	Objective function
h_j, η_j	Constraint function j
j	Index for constraint functions
J	Total number of constraints
$\mathcal{S}$	Constraint set
$\mathbb{P}$	Optimization problem
$\mathcal{L}$	Lagrangian
λ, $\boldsymbol{\lambda}$, Λ	Lagrange multiplier
X^*	Optimal solution
$\mathcal{X}^*$	Set of optimal solutions

List of Figures

1.1 Representation of a spatial filtering process with 5 sensors. . . 4
1.2 Comparison of the data partitioning. 9
1.3 Data flow of the distributed LCMV algorithm. 13
1.4 Overview of the links between the chapters. 16

2.1 Block diagram of DASF. 38
2.2 Examples of various network topologies with seven nodes. . . . 46
2.3 Example of a tree network. 48
2.4 Data flow of DASF. 51
2.5 Convergence results of DASF for various network sizes. 57
2.6 Convergence results of DASF for various network topologies. . 59
2.7 Convergence results of DASF for various number of columns of X. 60
2.8 Convergence results of DASF in an adaptive setting. 61

3.1 Convergence results of DASF for various number of constraints. 87
3.A Example network used in the proof rank($\mathbf{H}$) $= KJ - J$. 100

4.1 Convergence results of DASF with improved tracking. 119

5.1 Convergence results of DANSF. 134

6.1 Convergence comparison between DTRO and DASF. 154
6.2 Convergence results of DTRO in an adaptive setting. 156

7.1 Convergence comparison between F-DASF and DASF for the RTLS problem. 189
7.2 Convergence comparison between F-DASF and DASF for the quadratic over linear problem. 191

List of Tables

2.1 (D)SFO problem examples. 28
2.2 Practical approximations of common functions. 32
2.3 Network topologies used in the simulations. 54
2.4 Summary of parameters used in the simulations of DASF. . . . 56

4.1 Per batch communication cost for the original, straightforward, and proposed methods. 116
4.2 Summary of parameters used in the tracking simulations. . . . 118

5.1 Examples of problems with node-specific objectives. 125

7.1 Examples of problems with fractional objectives. 169
7.2 Summary of parameters used in the simulations of F-DASF. . . 187

1 Introduction

Sensor networks paved the way for many new technological and scientific advances by accessing information previously unavailable. For example, continuously being able to monitor the brain is crucial for many treatments in the biomedical and healthcare fields, especially for patients suffering from neurological disorders. Using sensor networks, electrical activity of the brain can be measured by sensors placed on the scalp of the patients and processed to extract valuable information, for example, to detect epileptic seizures, or control limbs. Alternatively, consider microphone networks that continuously monitor sounds in their environment, which can be used to distinguish between speakers in hearing aids, or to enhance speech in telephony. Traditional techniques that can be applied to achieve these goals include what we refer to as spatial filtering, or in other words, combining the data collected from all sensors, e.g., resulting from the electrical activity of the brain. Chronic monitoring is however only possible if the wearable devices are small enough, either to be comfortably worn in daily life activities or to be deployed in various locations in space, which means that they cannot have unlimited energy resources due to their small battery size. In its current state, this procedure requires the devices to continuously transmit the data these sensors measure to a central device, such as a smartphone or computer, which is unfortunately not sustainable for a long period of time for the sensors, as it requires large amounts of energy resources. A promising solution is to completely remove this central machine from consideration, such that the sensors achieve their goals by only collaborating with each other. Such a *distributed* approach requires designing new techniques from scratch, but achieving this will open the doors for many applications such as continuous monitoring, and unlock the potential of sensor networks in a broad range of fields, including biomedical, telecommunications, and acoustics.

1.1 Spatial filtering and signal fusion

Filtering can be summarized as manipulating data to extract certain information from it. When discussing filtering in a signal processing context, it is common to associate this term with temporal filtering, where samples of a signal measured at different time instants are summed together, each time sample having a different weight, or "importance", assigned to it. These weights constitute what is referred to as the filter. The objective of a filter is to obtain a filtered signal that satisfies certain desired properties. A famous example is the low-pass filter, such that the resulting filtered signal is free of higher frequencies such as noise.

Time is a core ingredient of signal processing as a signal is very commonly a function of time, however, it is not the only "dimension" where useful information can reside. Channels form what is often referred to as the spatial constituent of a signal and result from measuring signals at different locations in space, or under different conditions. Spatial signal processing aims to extract desired information shared by every one of these channels, which in many cases can only happen if these channels are treated as a whole, instead of analyzing each one of them independently. *Spatial filtering* or *signal fusion* corresponds to applying weights to these various channels and combining them so as to obtain a signal containing this desired information. Therefore, spatial filtering is similar to its temporal counterpart, the main difference being that the processing is done across the channels of a signal instead of time.

The reason why measuring signals at different locations is interesting is that it provides valuable (spatial) information about the signals from different channels that can be exploited, for example, to reconstruct a source signal, or extract important features. Spatial filtering is used in various fields such as:

- Biomedical signal processing [1–4]: Multiple electroencephalography (EEG) sensors can be placed on the scalp to measure the electrical activity of the brain. A spatial filter can then be used to process the raw signals measured on these EEG sensors to enhance/denoise the signals, compress the data, or extract relevant features. Applications of spatial filtering in EEG signal processing include seizure detection, sensory stimulus encoding and decoding, motor imagery analysis, and artifact removal, among others.
- Wireless communication [5–9]: Multiple antennas can be used to measure different channels of a signal transmitted from a source, which can then be used, for example, in source localization, or direction of arrival estimation.

- Acoustics [10–14]: A microphone array can be used to measure acoustic signals at different locations, and is used in applications such as hearing aids or video conferencing.

Without the added spatial information provided by the multiple channels of data collected by sensors at different locations, the applications mentioned above would either be very difficult to realize or even infeasible. This is because the redundancies present in the signals across the various channels constitute the core ingredients required to achieve the goals of these applications, where the correlation between the channels of the measured signals is exploited in such a way that the resulting filtered signal has desired, application-dependent properties. Note that we have used the term "correlation" which implies a statistical aspect in the spatial filtering process. Indeed, in most applications of spatial filtering, the signals measured at the sensors are *stochastic*, i.e., they contain random components. This can be due to measurement noise, interference from other sources of signals, change in experimental conditions, etc. Spatial filtering can then be interpreted as a way to correlate the signals measured across the different sensors.

Formally, we consider M sensors, where each sensor m measures its local signal y_m. A spatial filter for these signals is a collection of M weights x_m such that the resulting spatially filtered signal at time index t is

$$\widehat{y}(t) = x_1 y_1(t) + x_2 y_2(t) + \cdots + x_M y_M(t), \tag{1.1}$$

i.e., the filtered signal is obtained by linearly combining the signals measured at the different sensors. In this text, we consider discrete time signals sampled in a regular manner at an unspecified sampling frequency. It is common to stack every signal y_m into an $M-$dimensional signal $\mathbf{y}$ such that

$$\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_M(t)]^T \in \mathbb{R}^M \tag{1.2}$$

for each time sample t, where T is the transpose operator. The signals y_m are then defined as the channels of $\mathbf{y}$. We refer to $\mathbf{y}$ as the *global signal*, i.e., the multi-channel signal that results from the stacking of all signal channels measured at the M sensors. We can similarly define the *spatial filter* as

$$\mathbf{x} = [x_1, x_2, \ldots, x_M]^T \in \mathbb{R}^M \tag{1.3}$$

such that the spatial filtering operation in (1.1) can be compactly written in the following way:

$$\widehat{y}(t) = \mathbf{x}^T \mathbf{y}(t). \tag{1.4}$$

An example is illustrated in Figure 1.1. Various applications can require multiple spatial filters, which can be represented in a matrix form X, where each column

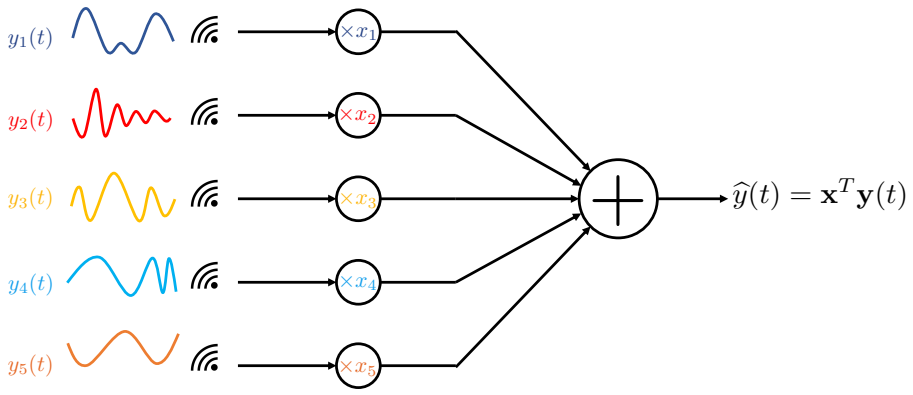


Figure 1.1: Representation of a spatial filtering process with 5 sensors.

of X corresponds to a vector-valued spatial filter [1, 8, 13]. In that case, the filtered signal $X^T\mathbf{y}$ also has multiple channels. In this chapter, we restrict ourselves to examples where the spatial filter is a vector for the sake of an easier presentation, while the more general matrix case will be considered in the remaining chapters.

An important question arising next is how to select a "good" spatial filter, which will be of central interest throughout this text. Here, we focus on optimal spatial filtering [9, 15, 16], where the filter is chosen such that it satisfies a certain optimality criterion. Classical examples are beamformers [17, 18], multi-channel Wiener filtering [1, 13], principal component analysis, filters based on (generalized) eigenvectors of spatial covariance matrices [19–21], canonical correlation analysis [22–24], etc. Mathematically, this means that the filter is taken to be the solution of an optimization problem. The exact optimization problem to choose is application-dependent, where different goals are translated into different problems, however, a common attribute across various applications is that the optimization problem and, by extension, the optimal filter depends on the data, i.e., $\mathbf{y}$. More specifically, the optimization problem typically depends on statistical quantities related to $\mathbf{y}$ (and other signals if they are present).

As an example, consider the linearly constrained minimum variance (LCMV) problem, frequently used in beamforming applications [9, 17, 18]. The LCMV filter is the solution of the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^M}{\text{minimize}} \quad & \mathbb{E}[|\mathbf{x}^T\mathbf{y}(t)|^2] \\ \text{subject to} \quad & \mathbf{x}^T B = \mathbf{a}^T, \end{aligned} \tag{1.5}$$

where $\mathbb{E}[\cdot]$ denotes the expectation, taken over the distribution of $\mathbf{y}$ at time t. Problem (1.5) can be interpreted as finding a spatial filter $\mathbf{x}$ such that the expected power of the filtered signal $\widehat{y}(t) = \mathbf{x}^T\mathbf{y}(t)$ is minimal while satisfying linear constraints[1]. In practical scenarios, the constraints enforce desired properties on the filter, and the filtered signal, for example to preserve signals from desired angles and frequencies in the beamforming context.

The objective of the LCMV problem can be rewritten as $\mathbf{x}^T R_{\mathbf{yy}}(t)\mathbf{x}$, where

$$R_{\mathbf{yy}}(t) = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]. \tag{1.6}$$

$R_{\mathbf{yy}}(t)$ is an $M \times M$ matrix corresponding to the (auto-)correlation (also called (auto-)covariance if $\mathbf{y}$ is zero-mean) matrix of $\mathbf{y}$ at time t. Assuming $R_{\mathbf{yy}}(t)$ to be invertible and B to be full column rank, the optimal solution $\mathbf{x}^*$ of Problem (1.5) at time t is given by (see Appendix B.2)

$$\mathbf{x}^*(t) = R_{\mathbf{yy}}^{-1}(t)B(B^T R_{\mathbf{yy}}^{-1}(t)B)^{-1}\mathbf{a}. \tag{1.7}$$

This shows that an optimal filter for Problem (1.5) — and in general, for all other spatial filtering problems we are interested in this text — is time-dependent without further assumptions. In practice, it is however not possible to compute a new filter for every time instant t because it would be computationally too expensive on one hand, and would on the other hand require access to the instantaneous auto-correlation matrix at each t. An assumption that can then be made on $\mathbf{y}$ to remove time-dependence is its *stationarity*, which implies that the collection $\{\mathbf{y}(t), \ldots, \mathbf{y}(t+N-1)\}$ of samples of $\mathbf{y}$ and the time-shifted collection $\{\mathbf{y}(t+T), \ldots, \mathbf{y}(t+N+T-1)\}$ follow the same distribution, for any t, N and T. Unfortunately, stationarity is often too strong of an assumption to consider in realistic scenarios, therefore, a common alternative is to assume wide sense stationarity. In the latter case, means and covariances are assumed to be constant in time, while auto-correlations are assumed to be finite, which would then remove the time-dependence of the correlation matrix of $\mathbf{y}$:

$$R_{\mathbf{yy}} \triangleq R_{\mathbf{yy}}(t) = R_{\mathbf{yy}}(0), \tag{1.8}$$

such that the optimal filter also becomes time-independent:

$$\mathbf{x}^* = R_{\mathbf{yy}}^{-1}B(B^T R_{\mathbf{yy}}^{-1}B)^{-1}\mathbf{a}. \tag{1.9}$$

As mentioned previously, the aim of many spatial filtering problems is to optimally combine the channels of the measured signals by exploiting the

[1] In practice, the signals, filters, and constraints are complex-valued in many LCMV applications, and spatial filtering in general. In this text, we concentrate on optimization over real variables, however, the results can be extended to the case of minimizing real functions with complex-valued arguments by decomposing them into their real and imaginary parts and/or by using Hermitian transpose operators instead of the standard matrix transpose [25–28].

correlation across these channels. Therefore, the correlation matrix of $\mathbf{y}$, and second-order moments in general, will play a central role in the optimization problems we will consider.

The next important question is how to compute statistical quantities related to the stochastic signals measured across the sensors. Computing these quantities exactly is not possible in practice, since it requires the perfect knowledge of the underlying distributions followed by the random variables that affect these signals, which is not realistic. However, they can be estimated, in general using the samples of the signals measured at the sensors at different time instants. Starting at time t, suppose that each sensor m collects N samples of y_m, then the correlation matrix of $\mathbf{y}$ is commonly estimated as a sample average over the collected N samples (see e.g. [3, 9, 29–31])

$$R_{\mathbf{yy}} \approx \frac{1}{N} \sum_{\tau=t}^{t+N-1} \mathbf{y}(\tau)\mathbf{y}^T(\tau). \tag{1.10}$$

If $\mathbf{y}$ satisfies a property named *ergodicity*, this approximation is guaranteed to be good for large values of N, as ergodicity implies that the true expectation equals the sample average as $N \to +\infty$.

In practical scenarios, the signals can often be considered (wide sense) stationary over a certain period of time, which we refer to as *short-term stationarity*. It is then important to periodically re-estimate the statistical quantities of interest using new signal observations and re-compute optimal filters using new batches of data so as to keep track of the changes in the signal statistics when they occur. This is commonly referred to as adaptive filtering [9, 16, 32, 33].

Throughout this text, we will assume that the signals we consider are short-term stationary, i.e., (wide sense) stationary over the period of time it takes to compute an optimal solution $\mathbf{x}^*$ of the considered problems, as is commonly done in the adaptive signal processing literature [9, 32–35]. The signals are also assumed to be ergodic such that a large enough number of samples N of the signals collected at the sensors is sufficient to approximate the statistical quantities of interest. Additionally, all signals we consider are assumed to be zero-mean, in which case the covariance matrix is equal to the correlation matrix. We will therefore adopt the former term in this text to refer to this quantity.

1.2 Wireless sensor networks and distributed processing

A wireless sensor network (WSN) is a collection of sensor nodes placed at various locations in space, where each node can measure a signal, process it locally, and is able to communicate data with other nodes in the network in a wireless fashion. Such WSNs allow to easily acquire data at multiple locations simultaneously, which is useful in several application domains including health monitoring [2,36–38], acoustics [39–43], spectrum sensing for cognitive (radio) sensor networks [44,45], structural monitoring [46], environmental studies [47,48], and many others [49–55]. Examples of WSNs include wireless body area networks (WBANs) [36,56–59], wireless EEG sensor networks (WESNs) [2,60,61], and wireless acoustic sensor networks (WASNs) [39,62–64].

Spatial filtering and signal fusion applications can be naturally extended to WSNs when the sensor nodes can process the signals they measure and transmit their data to other nodes in the network. Indeed, in most applications where WSNs are deployed, the aim is to find or estimate a common signal, filter, or a set of parameters that satisfy a pre-defined optimality criterion involving the observation data from all the nodes [27,65], i.e., solving a specific signal processing task in a collaborative fashion. A straightforward option to achieve this goal would be to centralize all signals measured at the sensor nodes in a central processing unit that can then make the required processing. For the LCMV example given in the previous section, this would mean that this central unit has access to samples of $\mathbf{y}$ such that it can compute the optimal filter $\mathbf{x}^*$. However, in many WSNs, including those where spatial filtering is applied, each sensor node can itself collect multiple channels, implying that each node is required to transmit a large number of samples of a multi-channel signal to the central unit. Moreover, in an adaptive context where signal statistics are varying in time, this process would need to be continuously repeated, which makes it very costly bandwidth and energy-wise (continuously transmitting high dimensional data). The sensor nodes typically do not have such large bandwidth and energy resources, therefore making the data centralization option impractical, or even infeasible.

To tackle this issue, *distributed* methods have to be considered, where a central processing unit is not needed and the nodes only communicate with each other. In this way, the nodes collaborate with each other to solve their tasks, by processing their data locally and sharing information with their neighboring nodes. Such *distributed signal processing* methods have been described in the literature for various tasks, where the goal is to find an optimal filter that minimizes a function f dependent on the collection $\mathbf{y}$ of the signals measured

across all sensor nodes in the network. We give a brief overview of some of the most well-known methods for such problems below [27].

- **Incremental strategies [66, 67]:** In this approach, the data sharing among nodes is done by following a cyclic path over the network. The starting node applies one "learning step", i.e., solves a local problem, which results in a local estimate of the filter that is eventually sent to the following node on the path. The next node then uses the filter it receives to compute a new one, and the process is repeated over the cyclic path. Note that this method restricts the number of information links, where each node can only receive information from a single other node.
- **Consensus [68, 69]:** This strategy requires the nodes to collaborate to reach a consensus regarding a target estimation variable. In each step, each node combines the estimates of its neighbors into a new local estimate via a specific learning step. This process continues until all nodes have reached a consensus on the value of the estimate. It also allows for asynchronous updates, where there is no need for a cyclic updating scheme as in the incremental case.
- **Diffusion [70–72]:** This strategy further develops the collaborative aspect of consensus where each node combines local estimates from its neighbors, but treats it in an adaptive and stochastic context. It was shown to lead to faster convergence results while also being more stable [73].

Other methods that can be combined with the ones above include gossip [74,75] and the alternating direction method of multipliers [76,77] (where the latter is also often used in a consensus setting).

The strategies presented above typically assume that the global task f can be decomposed into a sum of local tasks $f(\mathbf{x}, \mathbf{y}(t)) = \sum_k f_k(\mathbf{x}, \mathbf{y}_k(t))$, where each function f_k depends on the signal $\mathbf{y}_k$ measured at node k and is independent of the data measured at other nodes. An example is provided in Figure 1.2 (*bottom*) for the case of the least squares problem. Note that in this case, the dimension of $\mathbf{x}$ is not necessarily equal to the total number of channels in the WSN (as was the case for spatial filtering). Instead, it is typically equal to the dimension of $\mathbf{y}_k$, which here represents the dimension of the locally collected data at node k (containing either temporal, spatial, or spatio-temporal data).

In spatial filtering tasks, this per-node separability of the objective (or the constraints) is not satisfied since this type of separability would contradict the aim of spatial filtering techniques, which is to correlate every channel of the *network-wide* signal $\mathbf{y}$. In these cases, we typically have $f(\mathbf{x}^T\mathbf{y}(t)) = f(\sum_k \mathbf{x}_k^T\mathbf{y}_k(t))$, as can be seen in the LCMV beamformer example given in

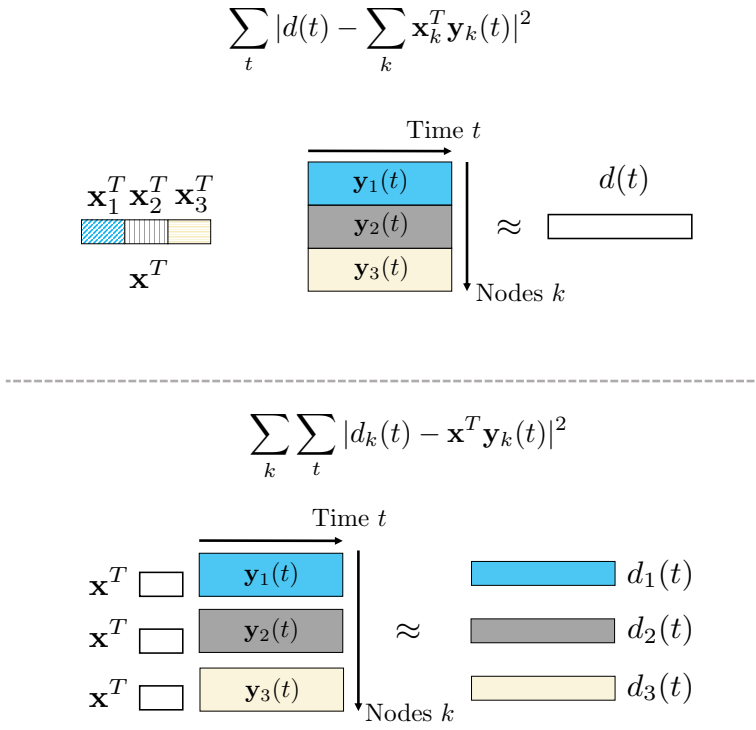


Figure 1.2: Comparison of the data partitioning in a 3−node network for the distributed spatial filtering and signal fusion (top) and the traditional consensus-type (bottom) settings with corresponding example objective functions for the case of least squares estimation. For each node k, the signal it measures locally is denoted as $\mathbf{y}_k$. In the consensus setting, note that the objective is per-node separable and has a shared optimization variable $\mathbf{x}$ which is assumed to be the same across all nodes.

(1.5). The difference in the way the data is partitioned between the traditional consensus-type setting and the distributed spatial filtering and signal fusion one is illustrated in Figure 1.2 for the least squares problem.

A commonly encountered strategy to solve distributed spatial filtering and signal fusion problems is to compute all inner products involving the data vector $\mathbf{y}$ via a standard consensus-type subroutine that performs in-network averaging or summation [78, 79], or by artificially rewriting the problem as a consensus problem (e.g., by treating the filter output d itself as a shared (consensus) optimization variable in the example of Figure 1.2). However, this strategy typically results in a distributed algorithm with nested iterative loops for each

sample time t, each in itself requiring a substantial number of communication rounds. Such an iterative distributed subroutine typically has to be executed from scratch for each new sample observation at the sensors (or for each block of consecutive samples in a block-based computation). This leads to a large communication burden, which also scales poorly with network size, i.e., a larger network increases the *per-node* transmission cost in these subroutines, as it takes longer to reach a consensus. In many cases, the use of such consensus-type subroutines even leads to a setting where each node shares more data than what it actually collects at its sensors. These methods are therefore not well-suited for developing distributed strategies for spatial filtering, as they are either not applicable or require a large amount of data to be communicated. Nevertheless, properties specific to spatial filtering can be exploited to design distributed algorithms, as explained next.

1.3 Distributed spatial filtering and signal fusion

Solving spatial filtering tasks in a distributed fashion is crucial in fields where energy, communication, and computational efficiency are a necessity such as EEG signal processing in WESNs or acoustic signal processing in WASNs, where sensor nodes with limited battery life have to be deployed and are expected to function over a relatively large period of time. For example, in WESNs, small wireless sensor nodes can be placed on the scalp of patients for daily monitoring, where spatial filtering techniques can be used, for example, to detect seizures [60].

The main obstacle encountered when designing distributed methods for spatial filtering is the fact that the spatial correlation across *all* channels of the network has to be exploited to find an optimal filter, since each node in the network contains a part of the global information that is useful to every other node. Mathematically, this means that the objective function depends on the variable $\mathbf{x}$ through the relationship $\mathbf{x}^T\mathbf{y}(t) = \sum_k \mathbf{x}_k^T\mathbf{y}_k(t)$, where $\mathbf{y}_k$'s correspond to the local signal measured at nodes k and $\mathbf{y}$ is the network-wide signal obtained by stacking all $\mathbf{y}_k$'s[2]. Note that $\mathbf{y}_k$ is itself also multi-channel, corresponding to the case where each node has multiple local sensors, each measuring a different signal.

This implies that *some* data has to be shared between the nodes in the network to be able to solve a spatial filtering problem in a distributed fashion. In the aim of reducing the amount of data to be shared within the network, a viable

[2]In applications where $\mathbf{y}$ is interpreted as a feature vector, this type of separation of the observed data is also known as feature partitioning or distributed features [80–85].

option would be to require the sensor nodes to compress the signals they observe before transmitting them to other nodes. The following algorithm proposed in [42] describes such a method to solve the LCMV beamforming problem (1.5) in a distributed fashion[3], which we will briefly explain for illustrative purposes. One of the main results of this thesis will be the development of a unifying framework to design such distributed algorithms for a wide range of spatial filtering problems, in which the example below will turn out to be a special case.

A distributed algorithm for the LCMV problem

Consider a WSN with K nodes, each measuring an M_k-channel signal $\mathbf{y}_k$. The aim of the network is to find the optimal filter $\mathbf{x}^*$ solving the LCMV problem (1.5) in a distributed fashion. The problem is repeated here for convenience:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^M}{\text{minimize}} \quad & \mathbb{E}[|\mathbf{x}^T\mathbf{y}(t)|^2] \\ \text{subject to} \quad & \mathbf{x}^T B = \mathbf{a}^T, \end{aligned} \tag{1.11}$$

where $\mathbf{y}$ is the $M-$channel signal defined as the stacking of all local signals:

$$\mathbf{y}(t) = [\mathbf{y}_1^T(t), \mathbf{y}_2^T(t), \ldots, \mathbf{y}_K^T(t)]^T, \tag{1.12}$$

with $M = \sum_{k=1}^K M_k$. Throughout this text, we will refer to $\mathbf{y}$ as the *global* or *network-wide signal*, as it contains all channels of every node in the network. Similarly, the optimization problem to be solved over the network, such as in (1.11), and the variable/filter $\mathbf{x}$ are referred to as the *global* or *network-wide problem* and *variable/filter*, respectively. The coefficient matrix B is $M \times L$ and assumed to be full column rank, also partitioned in a similar way as to $\mathbf{y}$:

$$B = [B_1^T, B_2^T, \ldots, B_K^T]^T, \tag{1.13}$$

such that each B_k is $M_k \times L$. We assume that B_k is available to node k while $\mathbf{a}$ is known to every node.

Let each node k collect N time samples of their *local signal* $\mathbf{y}_k$. This interval of time will correspond to one iteration i of the algorithm. We define $\mathbf{x}^i$ to be the *estimate at iteration* i of $\mathbf{x}$, partitioned in the same way as $\mathbf{y}$ in (1.12):

$$\mathbf{x}^i = [\mathbf{x}_1^{iT}, \mathbf{x}_2^{iT}, \ldots, \mathbf{x}_K^{iT}]^T, \tag{1.14}$$

[3]In the algorithm description, important terms that we will use throughout this text are emphasized in *italic*.

such that each local signal $\mathbf{y}_k$ has a corresponding *estimate of the local filter* $\mathbf{x}_k^i \in \mathbb{R}^{M_k}$ at node k. Each node k will then be responsible for updating its local filter $\mathbf{x}_k$ throughout the iterations of the algorithm.

At iteration i, let us select one node, denoted as q, to be the *updating node* at this iteration. A different updating node will be selected at each iteration, in a round-robin fashion. In [42], it was proposed to let every node $k \neq q$ in the network compress the N samples of its local signal in a linear fashion using its current estimate of its local filter $\mathbf{x}_k^i$:

$$\widehat{y}_k^i(t) = \mathbf{x}_k^{iT} \mathbf{y}_k(t) \in \mathbb{R}. \tag{1.15}$$

A similar compression is applied to the local coefficient matrices B_k by their corresponding nodes k:

$$\widehat{\mathbf{b}}_k^{iT} = \mathbf{x}_k^{iT} B_k \in \mathbb{R}^{1\times L}. \tag{1.16}$$

We will consider for now that the network is fully-connected, i.e., every node can communicate data to every other node in the network. Every node $k \neq q$ then transmits N samples of their compressed signal $\widehat{y}_k^i$ along with their compressed coefficient vector $\widehat{\mathbf{b}}_k^i$ to the updating node q. Note that for each node k, transmitting $\widehat{y}_k^i$ and $\widehat{\mathbf{b}}_k^i$ corresponds to a compression ratio of M_k compared to sending $\mathbf{y}_k$ and B_k.

Node q can then combine its local data, i.e., N samples of $\mathbf{y}_q$ and the coefficient matrix B_q, with the data it received by every other node in the network to obtain its *locally available data*, defined as

$$\widetilde{\mathbf{y}}_q^i(t) \triangleq [\mathbf{y}_q^T(t), \widehat{y}_1^i(t), \ldots, \widehat{y}_{q-1}^i(t), \widehat{y}_{q+1}^i(t), \ldots, \widehat{y}_K^i(t)]^T, \tag{1.17}$$

$$\widetilde{B}_q^i \triangleq [B_q^T, \widehat{\mathbf{b}}_1^i, \ldots, \widehat{\mathbf{b}}_{q-1}^i, \widehat{\mathbf{b}}_{q+1}^i, \ldots, \widehat{\mathbf{b}}_K^i]^T, \tag{1.18}$$

such that $\widetilde{\mathbf{y}}_q^i$ is an $\widetilde{M}_q-$channel signal and $\widetilde{B}_q^i$ is an $\widetilde{M}_q \times L$ matrix, where $\widetilde{M}_q = M_q + K - 1$. A visual summary of the data flow in the network is provided in Figure 1.3.

Defining a *local variable* $\widetilde{\mathbf{x}}_q$ at node q, we can then construct a compressed, local version of the global LCMV problem (1.11) at node q:

$$\begin{aligned} \underset{\widetilde{\mathbf{x}}_q \in \mathbb{R}^{\widetilde{M}_q}}{\text{minimize}} \quad & \mathbb{E}[|\widetilde{\mathbf{x}}_q^T \widetilde{\mathbf{y}}_q^i(t)|^2] \\ \text{subject to} \quad & \widetilde{\mathbf{x}}_q^T \widetilde{B}_q^i = \mathbf{a}^T. \end{aligned} \tag{1.19}$$

Note the similarity between the global problem (1.11) and the *local/compressed problem* (1.19) at node q: both have the same form, i.e., the local problem is

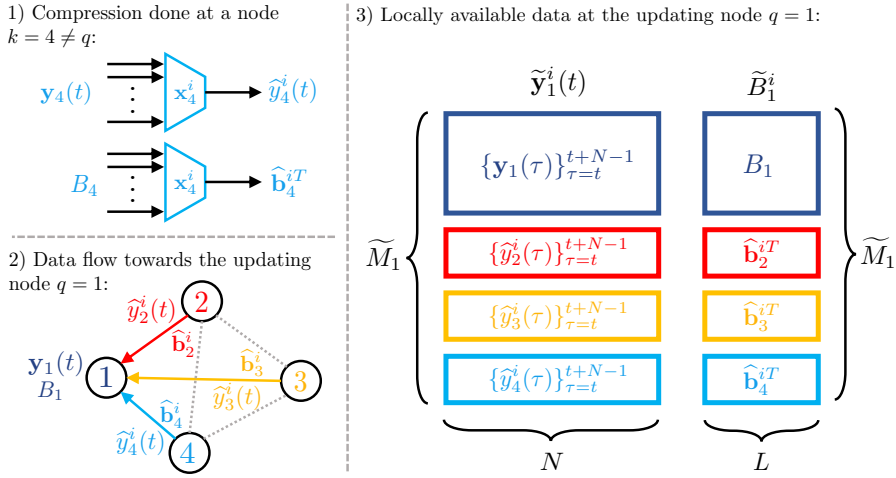


Figure 1.3: Different steps of the data flow of the distributed LCMV algorithm. 1) The nodes first compress their signal $\mathbf{y}_k$ and their coefficient matrix B_k. 2) The compressed data is transmitted to the updating node q. In this example, $q = 1$. 3) The updating node has access to its own uncompressed data, together with the compressed data it received from the other nodes in the network.

also an LCMV problem, yet with different parameters and a smaller dimension. Node q can then compute the solution $\widetilde{\mathbf{x}}_q^*$ of this problem as in (1.9), assuming that $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} = \mathbb{E}[\widetilde{\mathbf{y}}_q^i(t)\widetilde{\mathbf{y}}_q^{iT}(t)]$ is invertible and $\widetilde{B}_q^i$ full column rank:

$$\widetilde{\mathbf{x}}_q^* = (R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})^{-1}\widetilde{B}_q^i\left(\widetilde{B}_q^{iT}(R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})^{-1}\widetilde{B}_q^i\right)^{-1}\mathbf{a}, \tag{1.20}$$

where the covariance matrix $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ can be estimated using a sample average over N samples of $\widetilde{\mathbf{y}}_q^i$ as in (1.10).

The solution $\widetilde{\mathbf{x}}_q^*$ is then partitioned as $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$ in (1.17) and (1.18):

$$\widetilde{\mathbf{x}}_q^* = [\mathbf{x}_q^{(i+1)T}, g_1^{i+1}, \ldots, g_{q-1}^{i+1}, g_{q+1}^{i+1}, \ldots, g_K^{i+1}]^T, \tag{1.21}$$

where $\mathbf{x}_q^{i+1} \in \mathbb{R}^{M_q}$ and all g_k^{i+1}'s are scalars. Finally, the updating node q transmits each g_k^{i+1} to the corresponding node k such that all nodes in the network update their local variable as

$$\mathbf{x}_k^{i+1} = \begin{cases} \mathbf{x}_q^{i+1} & \text{if } k = q, \\ \mathbf{x}_k^i g_k^{i+1} & \text{if } k \neq q, \end{cases} \tag{1.22}$$

therefore resulting in a new estimate for the global variable $\mathbf{x}$. The LCMV beamformer output can also be computed locally at any time as $\widetilde{\mathbf{x}}_q^{(i+1)T}\widetilde{\mathbf{y}}_q^i(t)$.

At the next iteration, every node in the network collects a new batch of N samples of their local signal $\mathbf{y}_k$. Another node is then chosen to be the updating node and this process is repeated. Eventually, the iterations consist of different nodes solving local LCMV problems that can be viewed as compressed versions of the global LCMV problem. Starting from a randomly selected value for $\mathbf{x}^0$, convergence of the sequence $(\mathbf{x}^i)_{i\geq 0}$ to the optimal solution $\mathbf{x}^*$ of the global problem (1.11) can be shown to be achieved under mild constraints [42].

Note that this algorithm satisfies the following properties desired in WSNs applications:

- **Communication efficiency:** Each node compresses the signal samples they measure, going from M_k channels to a single one, before transmitting them to other nodes.
- **Computational efficiency:** The problem solved locally at each node is "smaller" than the global problem. For example, the matrices to invert to compute the optimal solution in (1.20) are smaller in size compared to the ones in (1.9), therefore requiring fewer computational resources.
- **Adaptivity:** By taking a new batch of samples at each iteration, the algorithm can adapt to changes in the statistical properties of the signals measured at the nodes, since the optimal solution will be computed using the new batch of data. Assuming short-term stationarity, the algorithm is time-recursive, in the sense that the filter at iteration $i+1$ depends on the filter obtained at iteration i. This is different from other approaches that apply consensus constraints on each new block of samples of the beamformer output signal z, which requires to re-estimate z from scratch for each new block.

Similar distributed algorithms have been proposed for various spatial filtering problems such as generalized eigenvalue decomposition (GEVD) [86], spatial principal component analysis (PCA) [87], least squares (LS), minimum mean square error (MMSE) or multi-channel Wiener filtering (MWF) [88, 89], beamforming [40, 90], canonical correlation analysis (CCA) [91] and generalized CCA [92]. We can remark that all of these methods use a similar strategy as the one described previously for the LCMV problem. However, each of these algorithms was treated separately, with convergence (to the optimal filter) analyses and conditions that were tailored to these specific cases. For example, it was assumed in the distributed LCMV approach that $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ is invertible and

$\widetilde{B}_q^i$ is full column rank, along with other mild technical conditions that we have omitted. The following important question then arises:

Is there a relationship between the convergence conditions of the algorithms corresponding to the different problems?

Additionally, some important optimization problems that can be used in spatial filtering applications do not yet have a distributed implementation, such as trace ratio optimization (TRO) [93–98], or regularized total least squares (RTLS) [99, 100]. A natural question resulting from finding a relationship between these methods is then:

Can we generalize the steps of these algorithms such that they can be extended to other spatial filtering problems?

1.4 Research objectives and overview

The aim of this thesis is to design and derive a framework for solving spatial filtering and signal fusion problems in a distributed fashion over any network topology, taking inspiration from the algorithm presented in the previous section. In particular, the framework should take into consideration the main requirements of WSN settings, namely

- **Communication efficiency**,
- **Computational efficiency** and
- **Adaptivity**.

Additionally, the framework should cover a large family of problems encountered in spatial filtering applications such that we are able to:

- Have a **unified** description of the procedure and its technical guarantees for all problems fitting the framework,
- **Generalize** to new spatial filtering problems without having to create a tailored approach for each problem.

Such a unified framework, named Distributed Adaptive Signal Fusion (DASF) will be described in Part I including a formal definition of the problems we are interested in solving, a detailed derivation of the steps of the algorithm, and a rigorous convergence analysis. Various modifications of the framework will then

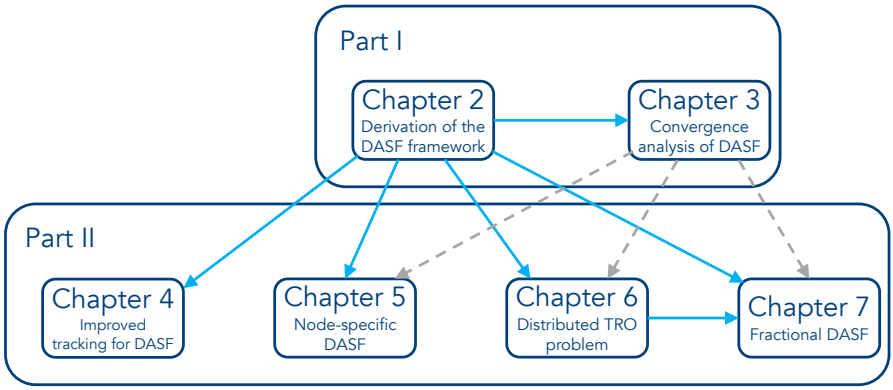


Figure 1.4: Overview of the links between the chapters. The contents of a chapter pointing to another other one with a solid line are important for the latter one, whereas for dashed lines the contents are useful but not necessary.

be discussed in Part II in the aim of extending it to further problem settings or improving its efficiency. A summary of the content of the chapters of this text is described below, while an overview of the links between the chapters is given in Figure 1.4.

Part I

Chapter 2 derives the steps of the DASF framework, which is a generic algorithmic framework aimed at solving spatial filtering problems in a distributed fashion by reducing the bandwidth and energy costs required from each node in the network. This unified framework can be used to solve a large family of spatial filtering problems on any network topology. Simulation results are provided for multiple problems in various experimental settings. This chapter is based on [101]:

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*, vol. 71, pp. 1863-1878, 2023.

Chapter 3 provides a rigorous convergence analysis of the DASF framework, detailing the various conditions required for the DASF algorithm to converge to an optimal solution. Various example problems are analyzed in detail while fixes

are provided for cases where certain convergence conditions are not satisfied. This chapter is based on [102]:

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.

Part II

Chapter 4 proposes a communication-efficient approach for improving the adaptive properties of the DASF algorithm. This is particularly interesting in practical scenarios where the statistical properties of the signals measured in the network change relatively quickly. This chapter focuses on fully-connected networks specifically and is based on [103]:

C. A. Musluoglu, M. Moonen, and A. Bertrand, "Improved Tracking for Distributed Signal Fusion Optimization in a Fully-Connected Wireless Sensor Network," in Proceedings of the *2022 30th European Signal Processing Conference (EUSIPCO)*, pp. 1836-1840, 2022.

Chapter 5 extends the DASF framework to node-specific optimization problems, i.e., when each node in the network has a different (yet related) problem to solve. This chapter is based on [104]:

C. A. Musluoglu and A. Bertrand, "A Distributed Adaptive Algorithm for Node-Specific Signal Fusion Problems in Wireless Sensor Networks," Accepted in Proceedings of the *2023 31st European Signal Processing Conference (EUSIPCO)*, pp. 1-5, 2023.

Chapter 6 is intended both as a case study of a specific spatial filtering problem and as a transition to Chapter 7. We will study the TRO problem and remark that the DASF algorithm can be made computationally more efficient for this problem. This chapter is based on [105, 106]:

C. A. Musluoglu and A. Bertrand, "Distributed Adaptive Trace Ratio Optimization in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3653-3670, 2021.

C. A. Musluoglu and A. Bertrand, "Distributed trace ratio optimization in fully-connected sensor networks," in Proceedings of the *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 1991-1995, 2021.

Chapter 7 proposes a computationally efficient DASF-based algorithm for spatial filtering problems with a fractional objective, by generalizing the results obtained in Chapter 6 for the TRO problem. This chapter is based on [107,108]:

C. A. Musluoglu and A. Bertrand, "Distributed Adaptive Signal Fusion for Fractional Programs," Submitted for review, pp. 1-13, 2023.

C. A. Musluoglu and A. Bertrand, "A computationally efficient algorithm for distributed adaptive signal fusion based on fractional programs," in Proceedings of the *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1-5, 2023.

Part III

Chapter 8 provides a summary of the results obtained in this text. Future research directions are also elaborated.

Appendices

Appendix A presents computational techniques to compute the gradient of a function of matrix variables and set up the Lagrangian of an optimization problem with matrix variables.

Appendix B lists various optimization problems of interest and derives their solutions using the techniques presented in Appendix A.

1.5 Mathematical notation

Throughout this text, apart from a few exceptions, uppercase letters are used to represent matrices and sets, the latter in calligraphic script, while scalars, scalar-valued functions, and vectors are represented by lowercase letters, the latter in bold. Additionally, we denote by "0" the zero scalar as well as the all-zero vectors/matrices, i.e., we use the same notation for the zero element independently of its size.

A certain mathematical object χ (such as a matrix, set, etc.) at node q and iteration i is denoted as χ_q^i. The notation $\left(\chi^i\right)_{i\in\mathcal{I}}$ refers to a sequence of elements χ^i over every index i in the ordered index set $\mathcal{I}$. If it is clear from the context (often in the case where i is over all natural numbers), we omit

the index set $\mathcal{I}$ and simply write $\left(\chi^i\right)_i$. A similar notation $\{\chi^i\}_{i\in\mathcal{I}}$ is used for non-ordered sets.

Part I

The distributed adaptive signal fusion (DASF) framework

2 | The distributed adaptive signal fusion framework: Algorithm derivation

This chapter is largely based on **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*, vol. 71, pp. 1863-1878, 2023.

ABSTRACT | In this chapter, we describe a general algorithmic framework for solving linear signal or feature fusion optimization problems in a distributed setting, for example in a WSN. The framework covers several classical spatial filtering problems, including minimum variance beamformers, multi-channel Wiener filters, principal component analysis, canonical correlation analysis, (generalized) eigenvalue problems, etc. The proposed distributed adaptive signal fusion (DASF) algorithm is an iterative method that solves these types of problems by allowing each node to share a linearly compressed version of the local sensor signal observations with its neighbors to reduce the energy and bandwidth requirements of the network. We first discuss the case of fully-connected networks and then extend the analysis to more general network topologies. The general DASF algorithm is shown to have several existing distributed algorithms from the literature as a special case, while at the same time allowing to solve new distributed problems as well with guaranteed convergence and optimality. This chapter focuses on the algorithm derivation of the DASF framework along with simulations demonstrating its performance. A technical analysis along with convergence conditions and proofs are provided in Chapter 3.

2.1 Introduction

Although various "workhorse" strategies and frameworks have been described in the literature to solve signal processing problems in a distributed fashion over a WSN (see [66–72, 74–77]), most of these methods rely on the separability of the global cost function f as a sum of local functions f_k: $f(X) = \sum_k f_k(X)$, where f_k depends only on the local data of node k, and where X is a *shared* optimization variable across all nodes. In spatial filtering and signal fusion problems, the aim is to find a network-wide linear spatial filter $X \in \mathbb{R}^{M \times Q}$, $Q < M$, to apply to the network-wide $M-$channel time signal $\mathbf{y}$ containing all sensor signals $\mathbf{y}_k$ from all nodes k in the network, where $\mathbf{y}(t) \in \mathbb{R}^M$ corresponds to the sample of $\mathbf{y}$ at time or sample index t. In Chapter 1, we have introduced such spatial filters for the case $Q = 1$, i.e., when X is a vector. Throughout the remaining parts of this text, we generalize them to the matrix case, where each column of X corresponds to a different vector-valued spatial filter. The filter X is typically designed to exploit the spatial correlation across the different nodes to optimize some network-wide objective function in the form $f(X^T\mathbf{y}(t))$. In this case, the function f itself is not per-node separable, but the argument is, i.e., $f(X^T\mathbf{y}(t)) = f(\sum_k X_k^T\mathbf{y}_k(t))$. We refer to such cases as a distributed signal fusion optimization (DSFO) problem to emphasize that the framework also applies to traditional array processing problems such as beamforming or spatial filtering.

Based on the block partitioning of the DSFO problems, a tempting alternative strategy could be to use a nonlinear Gauss-Seidel method [65, 109], or so-called block coordinate descent algorithms. These are iterative algorithms in which a block of variables in X is optimized while keeping all others fixed, and where the fixed blocks change across iterations. By selecting the blocks of X according to the nodes, each iteration of the nonlinear Gauss-Seidel method can then be "outsourced" to the node that is responsible for optimizing the corresponding coordinates, which makes it a better fit for the class of problems we are interested in. However, the convergence results for such nonlinear Gauss-Seidel methods often require conditions such as convexity assumptions or constraint sets that can be written as Cartesian products, where each factor corresponds to a constraint set for the block X_k [65, 109], which would not be satisfied in many spatial filtering optimization problems. Moreover, when optimizing the selected block of coordinates, forcing the other ones to remain constant leads to a new optimization problem that is often different from the original problem, and which can be significantly more difficult to solve.

This chapter introduces a generic distributed algorithm, referred to as the Distributed Adaptive Signal Fusion (DASF) algorithm, which can be used to

solve generic linear DSFO problems. In each iteration of the algorithm, a node within the network is selected to receive compressed data from other nodes and to locally solve a lower-dimensional version of the original network-wide problem. A convenient property is that an instance of the same algorithm that solves the centralized network-wide optimization problem can be used to also solve the local (compressed) problems at each iteration. The compression allows reducing bandwidth and energy usage in the network, while we still achieve convergence to the centralized solution. Moreover, since the locally constructed problem is of lower dimension, the computational cost required to solve it is also smaller compared to solving the network-wide problem. This makes the proposed algorithm also attractive in fully deterministic distributed settings where computational or memory resources are limited.

Various existing distributed algorithms can be shown to be special cases of our proposed DASF algorithm, including distributed algorithms for generalized eigenvalue decomposition (GEVD) [86], spatial principal component analysis (PCA) and eigenvalue decomposition (EVD) [87], least squares (LS), minimum mean square error (MMSE) or multi-channel Wiener filtering (MWF) [88, 89], linearly constrained minimum variance (LCMV) beamforming [40, 42, 90], canonical correlation analysis (CCA) [91] and generalized CCA [92]. However, each of these algorithms was previously treated separately, with convergence proofs that were tailored to these specific cases. Our aim is to thoroughly define a unified algorithmic framework that contains these already existing algorithms but also extends to new DSFO problems. We also provide a toolbox available both in Matlab and Python to generate and validate new distributed algorithms within this framework [110].

In Section 2.2, we formally define the framework setting and the assumptions used throughout this chapter. We then propose the DASF algorithm for fully-connected networks in Section 2.3, and later generalize it to general topologies in Section 2.4. Finally, we demonstrate the performance of the algorithm in a few new DSFO examples in Section 2.5, thereby demonstrating the generalization properties of the DASF framework.

2.2 Problem description

Consider a set of K nodes, where $\mathcal{K} = \{1, \ldots, K\}$ denotes the set of nodes. The nodes are interconnected in a connected graph where an edge between nodes k and q implies that these nodes can share data (e.g., via a wireless link). The set of neighbors of node k, i.e., the nodes that are connected to node k, is denoted by $\mathcal{N}_k$ (which excludes node k itself).

Each node $k \in \mathcal{K}$ measures samples of a local M_k-channel signal $\mathbf{y}_k(t) \in \mathbb{R}^{M_k}$ at every time instant t. We define the network-wide M-channel signal $\mathbf{y}$ as

$$\mathbf{y}(t) = [\mathbf{y}_1^T(t), \ldots, \mathbf{y}_K^T(t)]^T, \tag{2.1}$$

with $M = \sum_k M_k$. All $\mathbf{y}_k$'s, and therefore also $\mathbf{y}$, are assumed to be (short-term) stationary and ergodic stochastic signals, such that their statistical properties can be properly estimated given a sufficiently large number of samples at different time instants.

The different channels of $\mathbf{y}$ can be spatially correlated across all nodes in the network, and we do not assume this correlation structure to be known. In a centralized setting, the channels of $\mathbf{y}$ can be linearly combined (fused) using a network-wide spatial filter $X \in \mathbb{R}^{M \times Q}$ with Q output signals (with $Q \ll M$), where we typically aim to find an optimal X such that the filter outputs $X^T\mathbf{y} \in \mathbb{R}^Q$ satisfy some optimality conditions. Typical examples of such filter design problems are listed in Table 2.1, which can all be viewed as special cases of a general class of problems that will be formalized in the next subsection, which we refer to as (distributed) signal fusion optimization ((D)SFO) problems. This table is not exhaustive and various other signal fusion problems fit this framework (see e.g. [111,112]). Note that all of these examples require knowledge of the *full* correlation matrix $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$, which can only be estimated in a centralized setting where the data from all the nodes are collected in a single fusion center, allowing to estimate the correlation between any two channel pairs of $\mathbf{y}$. One of the key strengths of our proposed DASF framework is that it avoids such a data centralization (in the sense that there is never a node which has access to all the channels of $\mathbf{y}$, i.e., $R_{\mathbf{yy}}$ cannot be constructed), while still achieving the solution of the centralized problem.

Additionally, we consider inner products of the form X^TB, where $B \in \mathbb{R}^{M \times L}$ is a deterministic (i.e., fixed and time-independent) matrix. The LCMV problem introduced in Chapter 1 and also shown in Table 2.1 is an example where such an inner product with a deterministic matrix appears. Similar to $\mathbf{y}$ in (2.1), this term is defined as

$$B = [B_1^T, \ldots, B_K^T]^T, \tag{2.2}$$

where we only require that B_k is known to node k. We note that the argument B allows a deterministic representation of $\mathbf{y}$ in which multiple time samples of $\mathbf{y}$ are stored in the columns of B. Nevertheless, we make a distinction between both expressions to emphasize time-adaptive properties of the algorithm (see Section 2.2.2).

It is noted that some of the problems in Table 2.1 involve a second signal $\mathbf{v} : \mathbf{v}(t) = [\mathbf{v}_1^T(t), \ldots, \mathbf{v}_K^T(t)]^T$ collected by the WSN (e.g., in the case of GEVD and CCA), and possibly another filter W to be optimized (e.g., in the case

Table 2.1: (D)SFO problems that are special cases of (2.3)

$\mathbf{y}$, $\mathbf{v}$ *and* $\mathbf{d}$ *are multi-variate stochastic processes (signals). RR represents the ridge regression method. In the CCA case, the minimization is done with respect to* X *and* W*.*

Problem	Cost function to minimize	Constraints
LCMV [40, 42, 90]	$\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]$	$X^TB = H$
EVD / PCA [87]	$-\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]$	$X^TX = I_Q$
GEVD [86]	$-\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]$	$\mathbb{E}[X^T\mathbf{v}(t)\mathbf{v}^T(t)X] = I_Q$
TRO [105]	$-\frac{\mathbb{E}[\|\|X^T\mathbf{v}(t)\|\|^2]}{\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]}$	$X^TX = I_Q$
LS/MMSE /MWF [88, 89]	$\mathbb{E}[\|\|\mathbf{d}(t) - X^T\mathbf{y}(t)\|\|^2]$	$X \in \mathbb{R}^{M\times Q}$
RR	$\mathbb{E}[\|\|\mathbf{d}(t) - X^T\mathbf{y}(t)\|\|^2]$	$\mathrm{tr}(X^TX) \leq \alpha^2$
CCA [91]	$-\mathbb{E}[\mathrm{tr}(X^T\mathbf{y}(t)\mathbf{v}^T(t)W)]$	$\mathbb{E}[X^T\mathbf{y}(t)\mathbf{y}^T(t)X] = I_Q$ $\mathbb{E}[W^T\mathbf{v}(t)\mathbf{v}^T(t)W] = I_Q$

of CCA). This additional signal could either be derived from the same set of sensors (e.g., a time-lagged version of $\mathbf{y}$ as in [91], or observing $\mathbf{y}$ during two different regimes as in [86, 113]), or they could come from different types of sensors with which the nodes are equipped. In the remaining parts of this chapter, we will typically consider the case of a single filter X, a single observed (multi-channel) sensor signal $\mathbf{y}$ and a single deterministic parameter B, yet all results can be easily generalized to multiple signals, parameters or filters. This generalization will be briefly addressed at the end of the next subsection and in Section 2.3.3.

It is important to note here that every other quantity of the problem that is not represented by an inner product with X is assumed to be available at each node (e.g., H and $\mathbf{d}$ in the LCMV and LS / MMSE / MWF examples in Table 2.1). This means that each node is able to evaluate the objective and constraint functions of the optimization problem solved over the network if it has access

to $X^T\mathbf{y}$ and X^TB.

2.2.1 Scope of signal fusion optimization problems

We first provide a generic description of the signal fusion optimization (SFO) problems that will be covered by the DASF framework. While this description may seem rather exotic at first, we will provide several examples throughout this chapter to illustrate how it contains many familiar problems as a special case. The SFO problems studied in this work can be written in the following way:

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \varphi\left(X^T\mathbf{y}(t), X^TB\right) \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) \leq 0, \quad \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) = 0, \quad \forall j \in \mathcal{J}_E, \end{aligned} \tag{2.3}$$

where φ and the η_j's are differentiable scalar- and real-valued functions, and the sets $\mathcal{J}_I$ and $\mathcal{J}_E$ represent the index sets of inequality and equality constraints respectively. Additionally, we define $\mathcal{J} = \mathcal{J}_I \cup \mathcal{J}_E$, and the number of constraints in total is given by $J = |\mathcal{J}|$. Note that optimization problems of the form (2.3) have matrix-valued variables, which can be solved using similar techniques as to the vector case (see Appendix A).

An important observation is that X always appears in an inner product with $\mathbf{y}$ or B, which corresponds to a signal fusion or spatial filtering operation. Furthermore, note that the functions that contain a stochastic signal $\mathbf{y}$ as an argument must contain an operator to translate this stochastic variable into a deterministic loss or constraint function (i.e., the functions in instances of Problem (2.3) are deterministic, as any stochastic variable is converted into a deterministic value), for example through the use of an expectation operator (see Table 2.1). In most practical cases, including those mentioned in Table 2.1, the evaluation of φ requires the knowledge or estimation of the network-wide spatial covariance matrix $R_{\mathbf{yy}} = E[\mathbf{y}(t)\mathbf{y}^T(t)]$. In our case, we assume that this matrix is unknown, in which case the spatial correlation between the nodes should be learned on the fly by the proposed distributed algorithm.

The formulation (2.3) covers a wide range of popular spatial filtering and signal processing problems, including those shown in Table 2.1. For example, for the LCMV case, we have $\varphi(X^T\mathbf{y}(t)) = \mathbb{E}[\|X^T\mathbf{y}(t)\|^2]$, and $\eta_j(X^TB) = [X^TB - H]_j$ for each element $[X^TB - H]_j$ of the matrix $X^TB - H$. Note that quadratic terms of the form X^TX, which appear in some problems in Table 2.1, should be seen as $(X^TB)\cdot(X^TB)^T$ with $B = I_M$. The reader is also referred to Section 2.5 in which a few extra examples are provided.

Finally, to simplify notation in various parts of the algorithm derivation, we also define the differentiable functions f and h_j's, replacing φ and η_j's respectively, to describe Problem (2.3) as a function of X only, in which case $\mathbf{y}$ and B should be viewed as internal function parameters:

$$\begin{aligned} f(X) &\triangleq \varphi\left(X^T\mathbf{y}(t), X^T B\right), \\ h_j(X) &\triangleq \eta_j\left(X^T\mathbf{y}(t), X^T B\right), \ \forall j \in \mathcal{J}. \end{aligned} \tag{2.4}$$

Furthermore, we denote the constraint set of (2.3) as $\mathcal{S}$, the complete solution set as $\mathcal{X}^*$ and a single solution as X^*, i.e., $X^* \in \mathcal{X}^*$.

Note on further generalizations: The problem description (2.3) considers only one argument of each type $(X^T\mathbf{y}(t), X^T B)$ involving only one filter variable X, one stochastic signal $\mathbf{y}$ and one deterministic matrix B. However, this is merely for conciseness and intelligibility of the description of our framework, i.e., the framework can be straightforwardly generalized to multiple versions of variables and each of the two arguments in (2.3), e.g., to also cover the cases of GEVD and CCA in Table 2.1. Formally, the full scope of SFO problems we consider is

$$\begin{aligned} &\underset{X^{(a)},\forall a}{\text{minimize}} \ \varphi\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, ...\right), \ \forall a, b, c \\ &\text{subject to } \eta_j\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, ...\right) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ &\qquad\qquad \eta_j\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, ...\right) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{2.5}$$

Additionally, even though we restrict ourselves to real-valued arguments, the results can be extended to complex-valued arguments (with real-valued cost functions) based on standard techniques such as those explained in [25–28]. Further extensions are possible where each node minimizes a different function, as in [89, 114], which will be discussed in Chapter 5.

2.2.2 Adaptivity and approximation in practical settings

The problems we are interested in typically involve an expectation operator over random signals with unknown distributions. In practical settings, the expectation operators in the objective and constraint functions of (2.3) are usually approximated using sample averages [3, 9, 29–31]. The expectation operator over the distribution of $\mathbf{y}(t)$ for a generic deterministic function g

taking $\mathbf{y}(t)$ as an argument is then approximated[1] as

$$\mathbb{E}[g(\mathbf{y}(t))] \approx G(Y(t)) \triangleq \frac{1}{N} \sum_{\tau=t}^{t+N-1} g(\mathbf{y}(\tau)), \tag{2.6}$$

where $Y(t) = [\mathbf{y}(t), \ldots, \mathbf{y}(t+N-1)]$ denotes a matrix that contains N observations of $\mathbf{y}$, starting with the observation at time sample t. Here, it is assumed that the signal $\mathbf{y}$ is ergodic such that the expectation operators can be accurately approximated using (2.6). In particular for the second-order statistics, which are commonly encountered in signal processing problems (see Table 2.1), the approximation (2.6) results in

$$\mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)] \approx \frac{1}{N} Y(t) Y^T(t). \tag{2.7}$$

Table 2.2 gives the practical approximations of some commonly encountered functions in problems of interest (including expressions found in Table 2.1) using (2.6). Note that the stationarity assumption removes any time dependence from the problems of interest such that any window of contiguous time samples of $\mathbf{y}$ can be used. In the remaining parts of this thesis, we will generally use the expectation operator $\mathbb{E}[\cdot]$ for ease of notation and mathematical tractability. However, in practice this operator will typically be replaced with a finite-window sample average as in (2.6).

As mentioned in Chapter 1, we assume stationarity throughout this text for mathematical tractability, which is a common assumption in the analysis of adaptive filters [9,32–35]. However, in practice, we do not require the signals to be fully stationary, as long as the dynamics in the underlying signal statistics are sufficiently slow, such that (2.6) gives a reasonable approximation. In this case, the solution X^* of (2.3) becomes time-dependent. When the DASF framework is applied in such an adaptive context, the targeted instance of Problem (2.3) is effectively replaced by its sample average counterpart, where the statistics are regularly re-estimated in a block-based fashion, i.e., each time a new block of N samples becomes available. As we will explain in Sections 2.3 and 2.4, every new block of N samples will initiate one new iteration of the DASF algorithm, i.e., the iterations of DASF are spread over different sample blocks, such that the algorithm becomes time-recursive (implicitly assuming $G(Y(t)) \approx G(Y(t+N))$). We will demonstrate in Section 2.5.4 through an example that the proposed DASF algorithm can indeed track slow changes in the signal statistics, and is able to recover from abrupt changes.

[1] Convergence of the approximation to the true expectation for $N \to +\infty$ is studied in the stochastic optimization literature. We refer the reader to the sample average approximation method in particular [115].

Table 2.2: Practical approximations of common functions.

Approximations computed in practice to evaluate some commonly encountered functions using (2.6) and N observations of the stochastic signals, e.g., $\{\mathbf{y}(\tau)\}_{\tau=t}^{t+N-1}$. $Y(t)$ denotes the sample matrix of $\mathbf{y}$ for N observations, $Y = [\mathbf{y}(t), \ldots, \mathbf{y}(t+N-1)]$, while V and D are similarly defined for $\mathbf{v}$ and $\mathbf{d}$ respectively.

$\mathbb{E}[g(\mathbf{y}(t))]$	$G(Y(t))$
$\mathbb{E}[\|X^T\mathbf{y}(t)\|^2]$	$\|X^TY(t)\|_F^2/N$
$\mathbb{E}[X^T\mathbf{y}(t)\mathbf{y}^T(t)X]$	$X^TY(t)Y^T(t)X/N$
$\mathbb{E}[\|\mathbf{d}(t) - X^T\mathbf{y}(t)\|^2]$	$\|D(t) - X^TY(t)\|_F^2/N$
$\mathbb{E}[\mathrm{tr}(X^T\mathbf{y}(t)\mathbf{v}^T(t)W)]$	$\mathrm{tr}(X^TY(t)V^T(t)W)/N$

2.2.3 General assumptions

In order for our DASF algorithm to be applicable and achieve convergence and optimality, the problem (2.3) must satisfy some sufficient conditions, which are listed in this subsection for completeness, while the technical details will be explained in Chapter 3. It is noted that these conditions are usually satisfied for all examples listed in Table 2.1 (except in some contrived cases) and we refer the reader to Section 3.4 for some examples on how these conditions can be checked in practice. In addition to these sufficient conditions, we restate the implicit assumption from Section 2.2.1 that the functions f and h_j in (2.4) are smooth functions, i.e., they are continuously differentiable over the variable X on their respective domain, or equivalently, the functions φ and η_j are continuously differentiable over X for any $\mathbf{y}$ and B.

Assumption 1. The targeted instance of Problem (2.3) is well-posed, in the sense that the solution set is not empty and varies continuously with a change in the parameters of the problem.

The notion of (generalized Hadamard) well-posedness we require is based on [116,117]. The main difference is that we require the map from the parameter (inputs of the problem) space to the solution space to be continuous instead of upper semicontinuous, which is required for the convergence proof of the DASF algorithm. These technical details are presented in Chapter 3. Note that since

in practice, the DASF algorithm will be used for solving a particular instance of Problem (2.3), Assumption 1 is only required for that particular problem and not all problems within the scope of the framework. As an example, consider the PCA problem of Table 2.1. It can be shown that the PCA problem satisfies Assumption 1 if the covariance matrix of $\mathbf{y}$ is positive definite and its $Q+1$ largest eigenvalues are all distinct.

Assumption 2. The linear independence constraint qualifications (LICQ) hold at the solutions of Problem (2.3), i.e., the solutions satisfy the Karush-Kuhn-Tucker (KKT) conditions.

If X^* is a solution of Problem (2.3), then Assumption 2 implies that the gradients $\nabla_X h_j(X^*)$, $j \in \mathcal{J}^*$, are linearly independent[2], where $\mathcal{J}^* \subseteq \mathcal{J}$ is the set of all indices j for which $h_j(X^*) = 0$. We refer the reader to [118] for further details on constraint qualifications. If there is no constraint function η_j in Problem (2.3), Assumption 2 implies that $\nabla_X f(X^*) = 0$.

Assumption 3. f has compact sublevel sets in $\mathcal{S}$, i.e., for all $m \in \mathbb{R}$, $\{X \in \mathcal{S} \mid f(X) \leq m\}$ is compact.

Note that in $\mathbb{R}^{M \times Q}$, compactness is equivalent to closedness and boundedness of a set, which is a relatively mild condition. In fact, as is shown in Chapter 3, the assumption can be further relaxed by only requiring that at least one sublevel set $\{X \in \mathcal{S} \mid f(X) \leq f(X^0)\}$ is compact, where X^0 is the initialization point of the DASF algorithm.

Moreover, the convergence proof in Chapter 3 requires an additional sufficient condition that is akin to the LICQ. We postpone its definition to Chapter 3 because of its technical nature, and because it is a relatively mild condition, which is generally satisfied in practice. Nevertheless, an important implication of this condition, which is relevant to disclose at this point, will be that it imposes an upper bound on the number of constraints J the problem is allowed to have (see Section 3.3.2):

$$J \leq \min\left(\frac{Q^2}{K-1}\sum_{k\in\mathcal{K}}|\mathcal{N}_k|,\ \left(1+\min_{k\in\mathcal{K}}|\mathcal{N}_k|\right)Q^2\right). \tag{2.8}$$

[2] A set of matrices $\{A_j\}_{j\in\mathcal{J}}$ is linearly independent when $\sum_{j\in\mathcal{J}}\alpha_j A_j = 0$ is satisfied if and only if $\alpha_j = 0$, $\forall j \in \mathcal{J}$, or equivalently, when $\{\text{vec}(A_j)\}_{j\in\mathcal{J}}$ is a set of linearly independent vectors, where $\text{vec}(\cdot)$ is the vectorization operator.

Here, K is the total number of nodes, $|\mathcal{N}_k|$ is the number of neighbors of node k, and Q is the number of columns of X.

We also make the implicit assumption that a (centralized) algorithm is available to solve (with arbitrary accuracy) the targeted problem instance, i.e., the instance of (2.3) for which we aim to design a distributed algorithm. This is a reasonable premise since it makes little sense to design a distributed algorithm for a problem that cannot even be solved in a centralized setting. Nevertheless, it is important as our DASF framework will use the same solver to find solutions of compressed versions of the targeted problem at each node, as will be explained next. We note that there are no restrictions on the solver, which can be chosen freely (e.g., closed-form solutions, steepest descent methods, interior point methods, trust region methods, etc.).

2.3 Distributed adaptive signal fusion in fully-connected networks (FC-DASF)

For the sake of an easier exposition, let us first consider the special case of a fully-connected network where data transmitted by any node is received by every other node (more general topologies are treated in Section 2.4). As the optimization problem (2.3) depends on the full signal $\mathbf{y}$ and its (unknown) spatial correlation across the nodes, the nodes would need to share some information with each other. However, sharing the full signal $\mathbf{y}$ would require significant bandwidth and consume a large amount of power. Instead, each node will linearly compress its local M_k-channel signal into an $\widehat{M}-$channel signal (with $\widehat{M} < M_k$) before broadcasting it to the other nodes in the network. At iteration i, the compression is done by multiplying the signal $\mathbf{y}_k$ at node k with an $M_k \times \widehat{M}$ matrix F_k^i, which we refer to as the local compressor at node k. The $\widehat{M}-$dimensional compressed signal resulting from this compressive spatial filtering can be written as

$$\widehat{\mathbf{y}}_k^i(t) \triangleq F_k^{iT}\mathbf{y}_k(t). \tag{2.9}$$

The deterministic parameters B_k are similarly compressed to obtain $\widehat{B}_k^i$, which are also broadcast between nodes. The index i emphasizes that these compressors are not constant and will be iteratively updated across time. Note that the iteration index i is also added to the stochastic signal $\widehat{\mathbf{y}}_k$ in order to indicate that the content of the signal (and hence its statistics) changes in each iteration due to an update of the underlying compression matrix. However, this does not imply that we iterate over a single batch of samples of this signal, i.e., an update from $\widehat{\mathbf{y}}_k^i$ to $\widehat{\mathbf{y}}_k^{i+1}$ (or F_k^i to F_k^{i+1}) only affects future samples of $\widehat{\mathbf{y}}_k$

that are collected after performing the update. As a result, the compressor F_k operates as a block-adaptive filter, which updates its coefficients after every new block of N samples, such that F_k^i operates on the samples in the data matrix $Y(iN)$, whereas F_k^{i+1} operates on the samples in $Y(iN+N)$, where $Y(t)$ is defined as in (2.6). In other words, an update of the compressor F_k affects how future samples of $\mathbf{y}_k$ are compressed, yet previously collected samples will not be re-compressed or re-broadcast.

Equation (2.9) results in a compression ratio of $M_k/\widehat{M}$. The compression implies that the nodes do not have full access to the network-wide signal $\mathbf{y}$. Nevertheless, we will show that an optimal solution $X^* \in \mathcal{X}^*$ of Problem (2.3) will be achieved with $\widehat{M} = Q$ under mild technical conditions, despite the compression happening at every node. Here, Q denotes the number of columns of X, i.e., the number of output signals of the filter X (in many cases only a single-channel output is desired, such that $\widehat{M} = Q = 1$). In a fully-connected network, this implies that we must assume that $Q < M_k$ in order to achieve a bandwidth reduction at node k. However, in the case of more general topologies, we can also achieve this for $Q \geq M_k$ (see Section 2.4). If the bandwidth constraints allow it, the compression can be relaxed such that $\widehat{M} \geq Q$, for which faster convergence is expected due to the reduced compression ratio. In this work, we focus on the scenario $\widehat{M} = Q$, while the case $\widehat{M} \geq Q$ is briefly discussed in Remark 2.9.

After introducing the DASF algorithm for (2.3) in the next subsection, we will provide insights on the relationship between the network-wide problem and the problems solved at each node in Section 2.3.2. Extensions to the more general form (2.5) will be presented in Section 2.3.3.

2.3.1 Algorithm derivation

Consider the partitioning of the optimization variable X in per-node sub-blocks, i.e.,

$$X = [X_1^T, \ldots, X_K^T]^T, \tag{2.10}$$

where $X_k \in \mathbb{R}^{M_k \times Q}$. This way, every X_k has a corresponding local signal $\mathbf{y}_k$, i.e., the part of X that is applied to $\mathbf{y}_k$ in the expression $X^T\mathbf{y}$, such that we can write

$$X^T\mathbf{y}(t) = \sum_{k \in \mathcal{K}} X_k^T \mathbf{y}_k(t). \tag{2.11}$$

A similar observation for the local deterministic terms B_k implies that we can write the objective φ using the local filters and data:

$$\begin{aligned} f(X) &= \varphi(X^T \mathbf{y}(t), X^T B) \\ &= \varphi\left(\sum_{k\in\mathcal{K}} X_k^T \mathbf{y}_k(t), \sum_{k\in\mathcal{K}} X_k^T B_k\right). \end{aligned} \tag{2.12}$$

The constraint functions η_j can also be written in a similar way. Therefore, we are able to express the full optimization problem (2.3) using X_k's, $\mathbf{y}_k$'s and B_k's. The main idea behind the DASF framework is to partially reconstruct and solve a compressed version of Problem (2.3) at any selected node, which is referred to as the "updating node", and which changes from iteration to iteration.

Let us now set $\widehat{M} = Q$ and

$$F_k^i = X_k^i, \tag{2.13}$$

where X_k^i corresponds to the local estimate of X_k at node k at iteration i. Stacking them together as in (2.10), we obtain the estimate X^i of the global variable X at iteration i. This means that, within the DASF algorithm, the X_k^i's act both as compressors and as part of the optimization variable. Combining (2.9) and (2.13), the compressed signal that is broadcast by node k can be written as

$$\widehat{\mathbf{y}}_k^i(t) \triangleq X_k^{iT} \mathbf{y}_k(t) \in \mathbb{R}^Q, \tag{2.14}$$

which implies that the network-wide filter output at iteration i can be computed as the sum of the compressed signals in (2.14), i.e.,

$$\widehat{\mathbf{y}}^i(t) \triangleq X^{iT} \mathbf{y}(t) = \sum_{k\in\mathcal{K}} X_k^{iT} \mathbf{y}_k(t) = \sum_{k\in\mathcal{K}} \widehat{\mathbf{y}}_k^i(t). \tag{2.15}$$

In a similar way, the compressed deterministic terms at node k are

$$\widehat{B}_k^i \triangleq X_k^{iT} B_k \in \mathbb{R}^{Q\times L}, \tag{2.16}$$

and we have

$$\widehat{B}^i \triangleq X^{iT} B = \sum_{k\in\mathcal{K}} X_k^{iT} B_k = \sum_{k\in\mathcal{K}} \widehat{B}_k^i. \tag{2.17}$$

Since the network is assumed to be fully-connected, each node has access to (observations of) all signals $\widehat{\mathbf{y}}_k^i$, such that each node can compute the filter outputs (2.15). At iteration i, we select one node among the nodes of the network to be the updating node, which we denote as node $q \in \mathcal{K}$. For each different iteration, a different node in $\mathcal{K}$ will be selected to be the updating node q. In this work, we consider the round-robin selection $q = (i \mod K) + 1$.

Let q be the updating node at iteration i, and define the stacked vector containing all available signals at node q as

$$\widetilde{\mathbf{y}}_q^i(t) \triangleq [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_1^{iT}(t), \ldots, \widehat{\mathbf{y}}_{q-1}^{iT}(t), \widehat{\mathbf{y}}_{q+1}^{iT}(t), \ldots, \widehat{\mathbf{y}}_K^{iT}(t)]^T, \tag{2.18}$$

which contains $\widetilde{M}_q \triangleq M_q + Q(K-1)$ channels. Note that node q only has access to uncompressed observations of $\mathbf{y}_q$ and a corresponding batch of compressed observations of all the other nodes. Similarly, the matrix containing all available deterministic terms is obtained by stacking B_q which is available at node q and the compressed $\widehat{B}_k^i$'s received from other nodes:

$$\widetilde{B}_q^i \triangleq [B_q^T, \widehat{B}_1^{iT}, \ldots, \widehat{B}_{q-1}^{iT}, \widehat{B}_{q+1}^{iT}, \ldots, \widehat{B}_K^{iT}]^T, \tag{2.19}$$

which is an $\widetilde{M}_q \times L$ matrix. Based on (2.18) and (2.19), we define a new local variable $\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}$ at node q, such that we are able to formulate a local optimization problem using only data available at node q at iteration i:

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \varphi(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) \\ \text{subject to} \quad & \eta_j(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_j(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{2.20}$$

A key observation here is the similarity between (2.20) and (2.3). This means that node q can locally apply the same solver as the one used for solving the centralized problem, albeit on a problem of smaller size. Note that this implies that the computational cost required to solve Problem (2.20) is smaller compared to solving (2.3).

At iteration i, node q solves the local problem (2.20), and we denote its solution as $\widetilde{X}_q^*$, which can be partitioned as

$$\widetilde{X}_q^* = [X_q^{(i+1)T}, G_1^{(i+1)T}, \ldots, G_{q-1}^{(i+1)T}, G_{q+1}^{(i+1)T}, \ldots, G_K^{(i+1)T}], \tag{2.21}$$

where X_q^{i+1} is $M_q \times Q$ and each G_k^{i+1} is $Q \times Q$. By comparing (2.21) with (2.18), we see that G_k^{i+1} refers to the part of $\widetilde{X}_q$ that is multiplied with the received compressed data $\widehat{\mathbf{y}}_k^i$ from node k in the inner product $\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t)$ in (2.20). Since $\widehat{\mathbf{y}}_k^i(t) = X_k^{iT} \mathbf{y}_k(t)$, we can instead multiply the compressor X_k^i at node k with this matrix G_k^{i+1}. As a result, each node k in the network updates its local X_k as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q, \\ X_k^i G_k^{i+1} & \text{if } k \neq q, \end{cases} \tag{2.22}$$

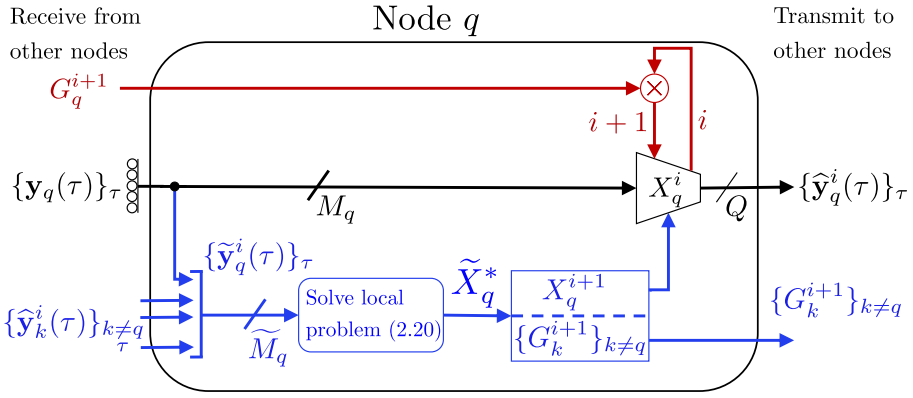


Figure 2.1: Block diagram representation of the steps followed by a given node q in the FC-DASF algorithm. The black part is executed for any sample time t at each node. The red and blue parts are only executed at each iteration increment $i \to i+1$, where the blocks in blue are only executed when node q is the updating node. Otherwise, the part in red is carried out. Node q has always access to its own signal samples $\mathbf{y}_q(t)$ measured at its own sensors (represented by rings, in black), while the compressed signal samples $\widehat{\mathbf{y}}_k^i(t)$ are transmitted to node q by the respective nodes k (represented by arrows in blue). For intelligibility, we omitted the data flow of the expression $X^T B$ from the diagram.

where X_q^{i+1} and G_k^{i+1} are obtained from the partitioning (2.21) of $\widetilde{X}_q^*$. Since the updating node q does not have access to the X_k^i of the other nodes $k \neq q$, it needs to communicate the matrices G_k^{i+1} to the other nodes, so that they can update their local X_k as well[3]. A block diagram of this process is provided in Figure 2.1.

If the minimization (2.20) has multiple solutions, an ambiguity exists on the choice of the local variable, which can be resolved by selecting a specific solution at each iteration. We propose to select the solution $\widetilde{X}_q^*$ for which the distance $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$ is minimal, where $\widetilde{X}_q^i$ is defined as

$$\widetilde{X}_q^i \triangleq [X_q^{iT}, I_Q, \ldots, I_Q]^T. \tag{2.23}$$

The choice of the Frobenius norm $\|\cdot\|_F$ as a distance metric is arbitrary. Other distance functions d can also be used (and might be better suited for the specific instance of Problem (2.3) at hand), as long as they are continuous and satisfy $d(X,Y) = 0 \Leftrightarrow X = Y$. These conditions on d are needed for the convergence of the proposed method, as explained in Section 3.D.

[3]Note that the communication cost to transmit these G_k matrices is negligible compared to the transmission of a batch of observations of $\widehat{\mathbf{y}}_k$'s (see also Remark 2.1).

Algorithm 1: Fully-Connected Distributed Adaptive Signal Fusion (FC-DASF) algorithm
Code available in [110]

X^0 initialized randomly, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) Every node k collects a new batch of N samples of $\mathbf{y}_k$ (see Remark 2.1), compresses these to N samples of $\widehat{\mathbf{y}}_k^i$ using (2.14) and transmits them to node q.
  $\widehat{B}_k^i$ is computed using (2.16) and transmitted to node q.
  at Node q **do**
    2a) Compute $\widetilde{X}_q^*$ as the solution of (2.20). If the solution is not unique, select the solution which minimizes $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$ with $\widetilde{X}_q^i$ defined in (2.23).
    2b) Partition $\widetilde{X}_q^*$ as in (2.21).
    2c) Transmit G_k^{i+1} to node k for every $k \neq q$.
  end
  3) Every node updates X_k^{i+1} according to (2.22).
  $i \leftarrow i + 1$

Note: Each iteration uses a different batch of N samples in step 1, i.e., the iterations can be spread over different time segments in order to avoid retransmitting the same batch of N samples across the network (see also Remark 2.1). This makes the sample time index t coupled to the iteration index i.

The procedure is then iteratively repeated, where new samples of signals measured at the nodes of the network are used, and transmitted to a new updating node after being compressed. All the steps of the FC-DASF algorithm as explained above are summarized in Algorithm 1. Algorithm 1 converges under mild technical conditions to an optimal filter X^* solving the network-wide problem (2.3). The convergence results with detailed analysis and proofs can be found in Chapter 3.

Remark 2.1. It is noted that a transmission of the compressed signal $\widehat{\mathbf{y}}_k^i$ at node k corresponds in practice to transmitting a batch with the N most recent time samples of $\widehat{\mathbf{y}}_k^i$. This allows for the receiving node q to estimate the necessary signal statistics to evaluate or optimize (2.20), where a larger value of N results in a closer approximation of the true value (remember that

the objective function in (2.3) and (2.20) has a built-in operator to transform the stochastic signal $\mathbf{y}$ into a deterministic loss, which in practice is usually replaced with an average over N samples, as in Table 2.2). Leveraging the (short-term) stationarity assumption, different batches of N samples are used in each iteration, such that the communication bandwidth becomes independent of the number of iterations. Therefore, Algorithm 1 behaves similarly to an adaptive filter which learns over time how to optimally filter newly observed samples (based on past samples). In a tracking context where the signal statistics of $\mathbf{y}$ change over time, the algorithm can still be applied if the statistics change slower than the convergence speed of the algorithm.

The DASF framework could in principle also be applied in a batch-mode (non-adaptive) framework, in which all operations are performed entirely on a single batch of samples (instead of spreading out the iterations over different sample batches of length N). In this case, the argument $X^T\mathbf{y}$ can be dropped, and the argument $X^T B$ can be used to represent the batch of samples, in which all available samples of $\mathbf{y}$ are stored in the columns of B.

Remark 2.2. Although each node is able to communicate with every other node in a fully-connected network, Algorithm 1 is still distributed in nature. Indeed, the network-wide data is never centralized, i.e., the updating nodes q have never access to $\mathbf{y}$ or B, but only to $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$, and therefore cannot estimate any network-wide statistics such as $\mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$, whereas a centralized solver has access to this information.

2.3.2 Link between the central and local problems

In this subsection, we further explain the link between the central problem (2.3) and the local problems (2.20) at the updating node q, where the latter can be viewed as a parameterized version of the former. This will provide some additional insights and show some useful properties of the DASF framework. Their relationship can be described by means of the transformation matrix:

$$C_q^i = \left[\begin{array}{c|c|c} 0 & \Theta_{<q}^i & 0 \\ \hline I_{M_q} & 0 & 0 \\ \hline 0 & 0 & \Theta_{>q}^i \end{array}\right] \in \mathbb{R}^{M \times \widetilde{M}_q}, \tag{2.24}$$

where $\Theta^i_{<q} = BlkDiag(X^i_1, \ldots, X^i_{q-1})$ and $\Theta^i_{>q} = BlkDiag(X^i_{q+1}, \ldots, X^i_K)$. Then, from (2.14) and (2.18), one can validate that

$$\widetilde{\mathbf{y}}^i_q(t) = C^{iT}_q \mathbf{y}(t), \tag{2.25}$$

while (2.16) and (2.19) result in

$$\widetilde{B}^i_q = C^{iT}_q B. \tag{2.26}$$

Using these relationships in (2.20), we see that

$$\begin{aligned}\varphi(\widetilde{X}^T_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^T_q \widetilde{B}^i_q) &= \varphi\left(\widetilde{X}^T_q (C^{iT}_q \mathbf{y}(t)), \widetilde{X}^T_q (C^{iT}_q B)\right) \\ &= \varphi\left((C^i_q \widetilde{X}_q)^T \mathbf{y}(t), (C^i_q \widetilde{X}_q)^T B\right) \\ &= f(C^i_q \widetilde{X}_q). \end{aligned} \tag{2.27}$$

Similarly, we find that, $\forall j \in \mathcal{J}$:

$$\begin{aligned}\eta_j(\widetilde{X}^T_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^T_q \widetilde{B}^i_q) &= \eta_j\left((C^i_q \widetilde{X}_q)^T \mathbf{y}(t), (C^i_q \widetilde{X}_q)^T B\right) \\ &= h_j(C^i_q \widetilde{X}_q). \end{aligned} \tag{2.28}$$

This implies that the local optimization variable $\widetilde{X}_q$ defines a parameterization of the global variable X. Indeed, if we define

$$\widetilde{X}_q = [X^T_q, G^T_1, \ldots, G^T_{q-1}, G^T_{q+1}, \ldots, G^T_K]^T, \tag{2.29}$$

where X_q is $M_q \times Q$ and every G_k is $Q \times Q$, we have at iteration i (for the updating node q)

$$X = C^i_q \widetilde{X}_q = \begin{bmatrix} X^i_1 \boxed{G_1} \\ \vdots \\ X^i_{q-1} \boxed{G_{q-1}} \\ \boxed{X_q} \\ X^i_{q+1} \boxed{G_{q+1}} \\ \vdots \\ X^i_K \boxed{G_K} \end{bmatrix}. \tag{2.30}$$

Note that only the framed variables in (2.30) appear as optimization variables in the local problem (2.20), which is clear from comparing (2.21) with (2.29).

This shows that the updating node q only has the full freedom to update X_q, i.e., its local compressor. The remaining parts of X can only change up to a multiplication from the right by a matrix G_k, $k \neq q$ when it is node q's turn to solve the local problem. Therefore, by sequentially changing the updating node across iterations, we allow every node to fully update its own local compressor while only manipulating the other sub-blocks of X within their respective column spaces.

The solution $\widetilde{X}_q^*$ of the local problem (2.20) at node q and iteration i, which can also be written as

$$\begin{aligned}\widetilde{X}_q^* &\triangleq \underset{\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}} \; f\left(C_q^i \widetilde{X}_q\right), \\ &= \underset{\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}} \; \varphi\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right), \end{aligned} \tag{2.31}$$

where $\widetilde{\mathcal{S}}_q^i$ denotes the constraint set of (2.20), defines a new point X^{i+1} for the global problem (2.3) via (2.30).

In the following lemma, we show that the global variable X^i produced by the DASF algorithm[4] always satisfies the global constraint set $\mathcal{S}$ for any iteration $i > 0$.

Lemma 2.1. For any iteration $i > 0$,

$$\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i \Leftrightarrow C_q^i \widetilde{X}_q \in \mathcal{S}. \tag{2.32}$$

In particular, $X^i \in \mathcal{S}$ and $\widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i$ for all $i > 0$.

Proof. From (2.28), it follows automatically that any point $\widetilde{X}_q$ in the constraint set of the local problem (2.20) has a corresponding point $X = C_q^i \widetilde{X}_q$ in the constraint set of the global problem (2.3). This implies that, if $\widetilde{X}_q$ is a feasible point of (2.20), the point X parameterized by $\widetilde{X}_q$, such that $X = C_q^i \widetilde{X}_q$, is a feasible point of (2.3), and vice versa, which proves (2.32).

For the second statement, we start by noting that (by construction) $X^{i+1} = C_q^i \widetilde{X}_q^*$. Since $\widetilde{X}_q^*$ is the solution of (2.20), X^{i+1} must be a feasible point of (2.3) for all $i \geq 0$, which follows from (2.32). Equivalently, X^i is a feasible point of

[4] The proof of Lemma 2.1 also holds for the general topology-independent DASF algorithm in Section 2.4.

(2.3), i.e., $X^i \in \mathcal{S}$, if $i > 0$. From the definition of $\widetilde{X}_q^i$ given in (2.23), we find that $X^i = C_q^i \widetilde{X}_q^i$. Then, the first result (2.32) of this lemma implies that $\widetilde{X}_q^i$ is a feasible point of (2.20), i.e., $\widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i$, for all $i > 0$. □

The results of Lemma 2.1 mean that all points $(X^i)_{i>0}$ generated by the algorithm will be in the constraint set of the global problem (2.3). Additionally, the final result states that $\widetilde{X}_q^i$ itself is a feasible point of (2.20), which is important to achieve convergence and a monotonic decrease in f, as it allows the algorithm to stay in the current point X^i if no reduction in f can be obtained at node q in iteration i, in which case $X^{i+1} = X^i$ (a formal proof is given in Chapter 3).

Remark 2.3. The fact that the local problem (2.20) inherits the structure from the global problem (2.3) is one of the key differences between the DASF framework and the nonlinear Gauss-Seidel method (sometimes referred to as the alternating optimization method), which would consist of only updating X_q^i in (2.3), while freezing the other X_k^i's $\forall k \neq q$. In the latter case, the subproblem that has to be solved in each iteration typically has a different structure than the original one, often leading to problems that are more difficult to solve or for which a straightforward solver might not even exist. Moreover, the extra degrees of freedom to manipulate the X_k's of other nodes through the G_k matrices in the parameterization (2.30) allow optimizing X^i over a larger subset of $\mathbb{R}^{M \times Q}$ in each individual iteration, leading to larger descents in the loss function f in each iteration. In Gauss-Seidel methods, these G_k matrices do not exist, i.e., all X_k's are fixed except one.

Remark 2.4. By combining (2.25) and (2.30), we find that

$$\widetilde{X}_q^{*T} \widetilde{\mathbf{y}}_q^i(t) = X^{(i+1)T} \mathbf{y}(t), \tag{2.33}$$

which means that node q always has access to the filtered signal $X^{(i+1)T}\mathbf{y} = \widehat{\mathbf{y}}^{i+1}$ based on the most recent version of the filter X^{i+1}. If any of the other nodes would act as a data sink, the updating node q has to forward the observations of $\widehat{\mathbf{y}}^{i+1}$ to the data sink.

2.3.3 Multiple signals, deterministic terms and variables

The DASF algorithm can be immediately adapted to the generalized version of (2.3) defined in (2.5), i.e., with multiple filters, signals and deterministic terms. In the case of multiple signals (stochastic variables) or deterministic terms appearing in the problem in the forms $X^T\mathbf{y}$ and X^TB respectively, every single object is treated as previously presented, creating new data to be communicated between the nodes for every additional expression. Examples include the GEVD and TRO given in Table 2.1 having two signals $\mathbf{y}$ and $\mathbf{v}$. An example of a problem with two deterministic terms is given in Section 2.5.

Similarly, we could also consider cases with multiple optimization variables. Taking the example of CCA given in Table 2.1, the two optimization variables (X, W) appear as $X^T\mathbf{y}$ and $W^T\mathbf{v}$ in the problem. Then, nodes $k \neq q$ compress their signals as $\widehat{\mathbf{y}}_k^i = X_k^{iT}\mathbf{y}_k$ and $\widehat{\mathbf{v}}_k^i = W_k^{iT}\mathbf{v}_k$ and transmit them to node q. Then, node q solves its local problem as

$$\left(\widetilde{X}_q^*, \widetilde{W}_q^*\right) = \underset{(\widetilde{X}_q, \widetilde{W}_q) \in \widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}} \varphi\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{W}_q^T \widetilde{\mathbf{v}}_q^i(t)\right). \tag{2.34}$$

Partitioning $\widetilde{W}_q$ as

$$\widetilde{W}_q = [W_q^T, H_1^T, \ldots, H_{q-1}^T, H_{q+1}^T, \ldots, H_K^T]^T, \tag{2.35}$$

similarly to (2.29) for $\widetilde{X}_q$, node q sends the G_k^{i+1} and H_k^{i+1}'s to the corresponding nodes k such that they update their local variables as in (2.22). The same procedure can be applied to more than two variables.

Remark 2.5. The communication burden required to transmit the compressed terms $X_k^{iT}B_k$ is minimal because B_k's are deterministic parameters, as opposed to the signals $\mathbf{y}_k$, which require sending batches of multiple compressed observations in each iteration to estimate the signal statistics at the updating node (see Remark 2.1). The communication cost of the deterministic part can be further reduced when we have an expression of the form $(X^TB^{(1)}) \cdot (X^TB^{(2)})^T = X^T\Gamma X$, where Γ is a block-diagonal deterministic matrix written as

$$\Gamma = BlkDiag(\Gamma_1, \ldots, \Gamma_K), \tag{2.36}$$

where each Γ_k is known by node k. At iteration i, each node k could then transmit

$$\widehat{\Gamma}_k^i = X_k^{iT}\Gamma_k X_k^i \in \mathbb{R}^{Q \times Q} \tag{2.37}$$

instead of $X_k^{iT} B_k^{(1)}$ and $X_k^{iT} B_k^{(2)}$, which is more efficient when $Q < 2L$. The expression (2.37) is analogous to (2.14) and (2.16) for the matrix Γ in (2.36). Similar to $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$ in (2.18) and (2.19), we can define the locally available version of Γ at the updating node q and iteration i as

$$\widetilde{\Gamma}_q^i = BlkDiag(\Gamma_q, \widehat{\Gamma}_1^i, \ldots, \widehat{\Gamma}_{q-1}^i, \widehat{\Gamma}_{q+1}^i, \ldots, \widehat{\Gamma}_K^i) = C_q^{iT} \Gamma C_q^i. \tag{2.38}$$

The last expression can be verified in a similar way as to (2.25) and (2.26).

Although expression (2.36) is quite specific, it is encountered often in spatial filtering, for example for orthogonality constraints such as $X^T X = I_Q$ or ℓ_2-norm regularization terms. For example, consider the PCA constraint $X^T X = I_Q$, where $B^{(1)} = B^{(2)} = I_M$. In this case, we have $\Gamma = I_M$ and it is therefore sufficient for the nodes to transmit $\widehat{\Gamma}_k^i = X_k^{iT} \Gamma_k X_k^i = X_k^{iT} X_k^i$ instead of X_k^i. At the updating node q and iteration i, the locally available version of I_M is then

$$\widetilde{\Gamma}_q^i = C_q^{iT} \Gamma C_q^i = C_q^{iT} C_q^i. \tag{2.39}$$

2.4 Topology-independent DASF (TI-DASF)

Until this point, we have considered fully-connected WSNs only, where every node in the network is a neighbor of every other node. In this section, we extend our discussions and describe the DASF algorithm for other network topologies. For this purpose, we first consider star topologies which are helpful to introduce the main idea, which will then lead to generalizations to tree topologies and finally to any (connected) network topology (see Figure 2.2).

2.4.1 Star topologies

We keep the same definitions introduced in Sections 2.2 and 2.3 and consider now that the WSN has a central node $c \in \mathcal{K}$ to which every other node $k \neq c$ is connected and having only node c as a neighbor (in the example in Figure 2.2, node c corresponds to node 1). In the case where the center node c is the updating node, we have the same setting as the fully-connected case and the steps described in the previous section apply, therefore we present here a strategy when the updating node $q \neq c$. A straightforward approach would be to let node c relay all the data from all other nodes to create a virtually fully-connected network. However, this would put high bandwidth requirements on node c,

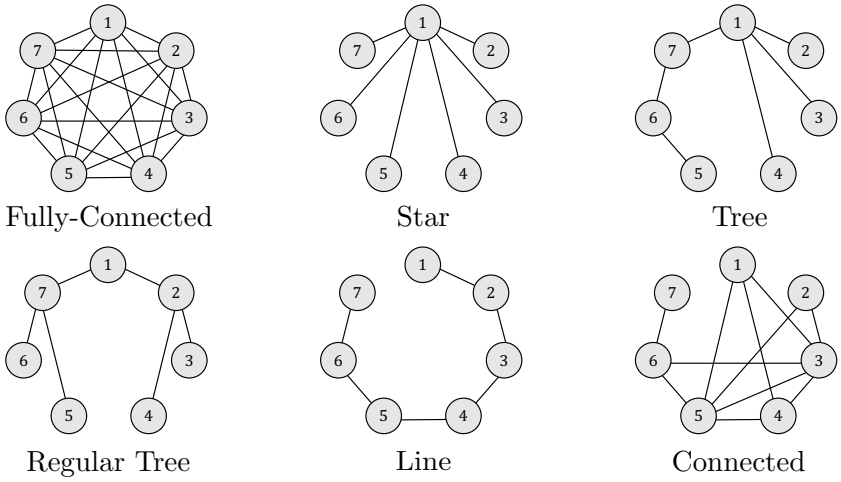


Figure 2.2: Examples of various network topologies with seven nodes.

which would not scale well with respect to network size. Instead, we claim that it is sufficient for node q to have access to the signal defined in (2.15), which is slightly rewritten here as

$$\widehat{\mathbf{y}}^i(t) = X^{iT}\mathbf{y}(t) = X_q^{iT}\mathbf{y}_q(t) + \sum_{k\neq q}\widehat{\mathbf{y}}_k^i(t). \tag{2.40}$$

Note that the second term can be computed at node c (which includes node c's own sensor observations $\mathbf{y}_c$, as well as the compressed signals $\widehat{\mathbf{y}}_k^i$ of the nodes $k \neq q$) such that only the sum has to be forwarded instead of the individual terms. The data received by node q from c is then a $Q-$channel signal given by

$$\widehat{\mathbf{y}}_{c\to q}^i(t) \triangleq \sum_{k\in\mathcal{K}\setminus\{q\}}\widehat{\mathbf{y}}_k^i(t), \tag{2.41}$$

and a similar expression can be written for the deterministic terms:

$$\widehat{B}_{c\to q}^i \triangleq \sum_{k\in\mathcal{K}\setminus\{q\}}\widehat{B}_k^i. \tag{2.42}$$

From the perspective of node q, the network consists of only itself and node c, so by following analogous steps to Algorithm 1, node q creates its vector of locally available data:

$$\begin{aligned}\widetilde{\mathbf{y}}_q^i(t) &= [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_{c\to q}^{iT}(t)]^T,\\ \widetilde{B}_q^i &= [B_q^T, \widehat{B}_{c\to q}^{iT}]^T,\end{aligned} \tag{2.43}$$

such that the corresponding $\widetilde{X}_q^*$ is obtained at iteration i by solving the local problem (2.20) using the data that is available at node q, as in (2.43). Similar to (2.21), we define the partitioning:

$$\widetilde{X}_q^* = [X_q^{(i+1)T}, G_c^{(i+1)T}]^T \in \mathbb{R}^{(M_q+Q)\times Q}, \tag{2.44}$$

where $G_c^{i+1} \in \mathbb{R}^{Q\times Q}$ is analogous to the G_k^{i+1}'s in the previous section. Since node q has only one link, there is only one such matrix resulting from solving the local problem (2.20), which is sent to node c. Finally, the central node c disseminates this matrix G_c^{i+1} to the other nodes to update their local compressor as in (2.22), but with $G_k = G_c$ for all k.

2.4.2 Tree topologies

We now consider a network represented by a tree, i.e., a graph without cyclic paths. A leaf node is defined as a node with a single neighbor, i.e., a node k for which $|\mathcal{N}_k| = 1$. Recalling that our objective is to be able to recreate (2.40) at the updating node q, we perform an in-network fusion across the different tree branches that are rooted in node q. This fusion can be done in a bottom-up fashion without central coordination. Indeed, the strategy for each node $k \neq q$ is to wait until it has received the compressed signals from all its neighbors except one (denoted as node n), sum these, and add its own compressed signal $\widehat{\mathbf{y}}_k^i$, and transmit to its remaining neighbor n. Formally, the compressed signal being sent from node $k \neq q$ to n at iteration i is

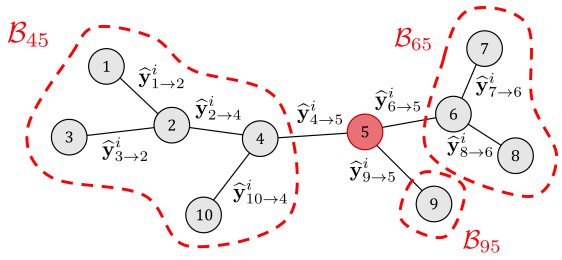
$$\widehat{\mathbf{y}}_{k\to n}^i(t) = X_k^{iT}\mathbf{y}_k(t) + \sum_{l\in\mathcal{N}_k\setminus\{n\}} \widehat{\mathbf{y}}_{l\to k}^i(t). \tag{2.45}$$

Note that this is a recursive definition, which is bootstrapped by the leaf nodes, for which the second (recursive) term vanishes. This data fusion flow is illustrated in Figure 2.3 for an example network. The fused signals will eventually arrive at the updating node q which receives

$$\widehat{\mathbf{y}}_{n\to q}^i(t) = X_n^{iT}\mathbf{y}_n(t) + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{\mathbf{y}}_{k\to n}^i(t) = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i(t), \tag{2.46}$$

from each of its neighbors $n \in \mathcal{N}_q$, where we define $\mathcal{B}_{nq}$ to be the connected subgraph containing node n when the link between nodes n and q is removed (see Figure 2.3). The same process is applied to the deterministic terms such that node q receives

$$\widehat{B}_{n\to q}^i = X_n^{iT}B_n + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{B}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} \widehat{B}_k^i \tag{2.47}$$



$$C_5^i = \begin{bmatrix} 0 & X_1^i & 0 & 0 \\ 0 & X_2^i & 0 & 0 \\ 0 & X_3^i & 0 & 0 \\ 0 & X_4^i & 0 & 0 \\ I_{M_5} & 0 & 0 & 0 \\ 0 & 0 & X_6^i & 0 \\ 0 & 0 & X_7^i & 0 \\ 0 & 0 & X_8^i & 0 \\ 0 & 0 & 0 & X_9^i \\ 0 & X_{10}^i & 0 & 0 \end{bmatrix}$$

Figure 2.3: Example of a tree network where the updating node is node 5. Each neighbor of node 5 creates its own cluster containing the nodes "hidden" from node 5 behind them, shown here as $\mathcal{B}_{45}$, $\mathcal{B}_{65}$, $\mathcal{B}_{95}$. The resulting transition matrix is given by C_5^i.

from all its neighbors $n \in \mathcal{N}_q$.

Writing $\mathcal{N}_q = \{n_1, \ldots, n_{|\mathcal{N}_q|}\}$, we have the vector of available data at the updating node q:

$$\begin{aligned} \widetilde{\mathbf{y}}_q^i(t) &= [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_{n_1\to q}^{iT}(t), \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}(t)]^T, \\ \widetilde{B}_q^i &= [B_q^T, \widehat{B}_{n_1\to q}^{iT}, \ldots, \widehat{B}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T. \end{aligned} \tag{2.48}$$

As an example, consider the network provided in Figure 2.3. When node 5 is the updating node, we have $\mathcal{N}_5 = \{4, 6, 9\}$ and its locally available data is then given by

$$\begin{aligned} \widetilde{\mathbf{y}}_5^i(t) &= [\mathbf{y}_5^T(t), \widehat{\mathbf{y}}_{4\to5}^{iT}(t),\ \widehat{\mathbf{y}}_{6\to5}^{iT}(t),\ \widehat{\mathbf{y}}_{9\to5}^{iT}(t)]^T, \\ \widetilde{B}_5^i &= [B_5^T, \widehat{B}_{4\to5}^{iT},\ \widehat{B}_{6\to5}^{iT},\ \widehat{B}_{9\to5}^{iT}]^T. \end{aligned} \tag{2.49}$$

We note that the relationship between the network-wide data $\mathbf{y}$, B and the locally available $\widetilde{\mathbf{y}}_q^i$, $\widetilde{B}_q^i$ can again be described by means of a compression matrix C_q^i as in (2.25)-(2.26), such that $\widetilde{\mathbf{y}}_q^i = C_q^{iT}\mathbf{y}$ and $\widetilde{B}_q^i = C_q^{iT}B$. An example of such a matrix C_q^i is shown in Figure 2.3. We recommend the reader to use the example of this figure to appreciate the structure of this matrix, yet we also provide a general definition for completeness. In general, this matrix can be defined as

$$C_q^i = \left[\begin{array}{c|c} 0 & \\ I_{M_q} & \Theta_{-q}^i \\ 0 & \end{array}\right], \tag{2.50}$$

where I_{M_q} is placed in the $q-$th block-row, and Θ^i_{-q} is a block matrix with K block-rows and $|\mathcal{N}_q|$ block-columns, where the block at the $k-$th block-row and $m-$th block-column is represented by $\Theta^i_{-q}(k,m) \in \mathbb{R}^{M_k \times Q}$. Each block-column corresponds to one of the neighbors $n \in \mathcal{N}_q$ of q, which we re-index as $m_n \in \{1, \dots, |\mathcal{N}_q|\}$. Then, we have

$$\Theta^i_{-q}(k, m_n) = \begin{cases} X^i_k & \text{if } k \in \mathcal{B}_{nq}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.51}$$

As in the previous cases, the transition from the local variable to the network-wide one is given by $X = C^i_q \widetilde{X}_q$ at node q and iteration i. We can verify that $\widetilde{X}_q$ is a feasible point of the local problem if and only if $C^i_q \widetilde{X}_q$ is a feasible point of the global problem, i.e., (2.32) and more generally Lemma 2.1 also holds here. Node q then solves its local problem (2.20) using the locally available data described in (2.48), to obtain $\widetilde{X}^*_q$, partitioned as

$$\widetilde{X}^*_q = \left[X_q^{(i+1)T}, G_{n_1}^{(i+1)T}, \dots, G_{n_{|\mathcal{N}_q|}}^{(i+1)T} \right]^T \in \mathbb{R}^{\widetilde{M}_q \times Q}, \tag{2.52}$$

with $\widetilde{M}_q \triangleq M_q + |\mathcal{N}_q| \cdot Q$. Each $G^{i+1}_n \in \mathbb{R}^{Q \times Q}$ is then disseminated into the corresponding subgraph $\mathcal{B}_{nq}$ through node n (and the nodes behind it in $\mathcal{B}_{nq}$) and every node updates its compressor as

$$X^{i+1}_k = \begin{cases} X^{i+1}_q & \text{if } k = q, \\ X^i_k G^{i+1}_n & \text{if } k \in \mathcal{B}_{nq},\ n \in \mathcal{N}_q, \end{cases} \tag{2.53}$$

such that we again have $X^{(i+1)T}\mathbf{y} = \widetilde{X}^{*T}_q \widetilde{\mathbf{y}}^i_q$ and $X^{(i+1)T} B = \widetilde{X}^{*T}_q \widetilde{B}^i_q$.

From (2.53), we observe that we have parameterized the global variable X at node q and iteration i as

$$X = C^i_q \widetilde{X}_q = \begin{bmatrix} X^i_1 \boxed{G_{n(1)}} \\ \vdots \\ X^i_{q-1} \boxed{G_{n(q-1)}} \\ \boxed{X_q} \\ X^i_{q+1} \boxed{G_{n(q+1)}} \\ \vdots \\ X^i_K \boxed{G_{n(K)}} \end{bmatrix}, \tag{2.54}$$

where C^i_q is given in (2.50) and $n(k)$ is the neighbor $n \in \mathcal{N}_q$ of node q such that $k \in \mathcal{B}_{nq}$. This can be compared with (2.30) for the fully-connected case.

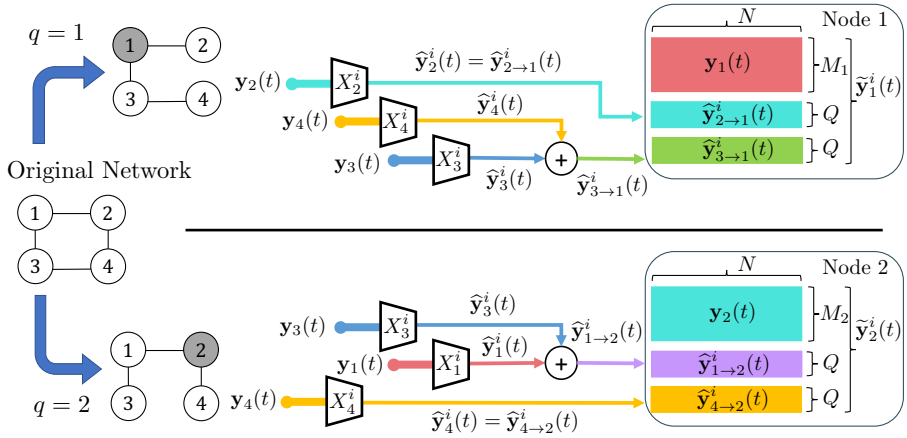
Since the optimization variable of (2.20) is partitioned as in (2.52), we can see from (2.54) that, similarly to the fully-connected case, the updating node q can "freely" update its filter X_q, while the filters of the other nodes can only change up to a right-hand side matrix multiplication with a G_n-matrix (the updating variables are the framed variables in (2.54)). In contrast to the fully-connected case in (2.30), some of the G_n's are constrained to be equal due to the network topology since $k, l \in \mathcal{B}_{nq} \Rightarrow G_{n(k)} = G_{n(l)}$. This implies that the degrees of freedom in the updating steps of the tree-based DASF algorithm are determined by the number of neighbors of the updating node.

Remark 2.6. Sometimes it is possible that a node cannot generate Q linearly independent output channels using (2.45). For example, this happens if node k is a leaf node and $M_k < Q$. In this case, node k will send its raw uncompressed sensor data $\mathbf{y}_k$ instead of $\widehat{\mathbf{y}}^i_{k\to n}$ defined in (2.45). The neighbor n that receives the raw data from node k will then treat this data as part of its own sensor signals, i.e., $\mathbf{y}_n$ is stacked with $\mathbf{y}_k$ and its sensor channel count becomes $M_n + M_k$. In other words, the data flow starting at the leaf nodes can initially consist of raw sensor channels until a node has more than Q channels to compress them into a $Q-$channel signal. Therefore, the number of transmitted channels is at most Q per node, but can also be less than Q. An analogous statement applies to the deterministic terms B_k.

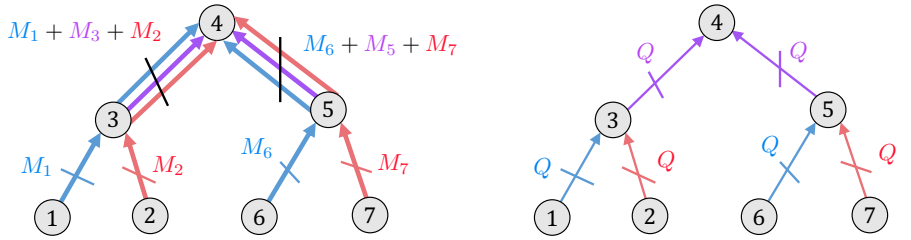
2.4.3 General connected graphs

Suppose the network is represented by a connected graph $\mathcal{G}$, which can potentially contain cycles. The main difference with the previous subsection is that there is more than one choice to forward the compressed data to the updating node q. We therefore propose to prune the graph $\mathcal{G}$ into a (different) tree at each iteration i, so that we can apply the same steps as the ones described for the tree topology case. The resulting tree is denoted as $\mathcal{T}^i(\mathcal{G}, q)$ to highlight the fact that the pruning function $\mathcal{T}^i$ depends on the current updating node q, while the dependence of the pruning function on the index i is for flexibility purposes. An example of the data flow in pruned networks for two different updating nodes is given in Figure 2.4a.

The choice of the mapping function $\mathcal{T}^i$ is a free design choice as long as the resulting tree remains connected. However, the convergence results in Chapter 3 that define the bound (2.8) will also assume that the pruning function does not remove the links between the updating node q and its neighbors. Indeed, if (2.8) is satisfied, then this rule ensures that $J \leq (1 + |\mathcal{N}_q|)Q^2$ at any updating



(a) Data flow for a 4−node network when nodes $q = 1$ (top) and $q = 2$ (bottom) are the updating node. The updating node has always access to its own data $\mathbf{y}_q$, while receiving fused signals from every other node in the network through its neighboring nodes.



(b) Comparison between a straightforward relaying approach (left) and the scalable fuse-and-forward approach the DASF algorithm uses (right). In this example, node 4 is the updating node.

Figure 2.4: Data flow of the DASF algorithm.

node q, which is one of the convergence conditions for the DASF algorithm (see Section 3.3.2). Furthermore, disconnecting node q from a neighbor would also reduce the convergence speed, since it would lead to a smaller number of resulting G_n's in (2.52)-(2.53) and therefore a lower number of degrees of freedom in the local optimization problem (2.20), typically leading to a smaller descent of the cost φ or f. In comparison, in the case of fully-connected networks, we were able to use $(K-1)$ of such G_n's in each iteration, which — as we will show in Section 2.5 — leads to the fastest convergence.

A simple distributed protocol to establish $\mathcal{T}^i$, i.e., to set up a tree that satisfies the aforementioned rule, is to let the updating node q broadcast a token to each neighbor. Once a node receives a token, it acknowledges a link to the node from which it received it, and it then broadcasts that token to its remaining

Algorithm 2: Topology-Independent Distributed Adaptive Signal Fusion (TI-DASF) algorithm
Code available in [110]

X^0 initialized randomly, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) The network $\mathcal{G}$ is pruned into a tree $\mathcal{T}^i(\mathcal{G}, q)$.
  2) Every node k collects a new batch of N samples of $\mathbf{y}_k$. All nodes compress these to N samples of $\widehat{\mathbf{y}}_k^i$ as in (2.14). $\widehat{B}_k^i$ is computed using (2.16).
  3) The nodes sum-and-forward their compressed data towards node q via the recursive rule (2.45) (and a similar rule for the $\widehat{B}_k^i$'s). Node q eventually receives N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ along with $\widehat{B}_{n\to q}^i$ given in (2.46)-(2.47) from all its neighbors $n \in \mathcal{N}_q$.
  at Node q **do**
    4a) Compute $\widetilde{X}_q^*$ as the solution of (2.20) where $\widetilde{\mathbf{y}}_q^i$, $\widetilde{B}_q^i$ and $\widetilde{X}_q$ are redefined as in (2.48) and (2.52). If the solution of (2.20) is not unique, select the solution which minimizes $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$ with $\widetilde{X}_q^i$ defined as in (2.23).
    4b) Partition $\widetilde{X}_q^*$ as in (2.52).
    4c) Disseminate G_n^{i+1} to all nodes in $\mathcal{B}_{nq}$, $\forall n \in \mathcal{N}_q$.
  end
  5) Every node updates X_k^{i+1} according to (2.53).
  $i \leftarrow i + 1$

Note: Each iteration uses a different batch of N samples in step 2, i.e., the iterations can be spread over different time segments in order to avoid retransmitting the same batch of N samples across the network (see also Remark 2.1). This makes that the sample time index t is coupled to the iteration index i.
Note: As in FC-DASF, the fused output signal $X^T\mathbf{y}(t) = \widehat{\mathbf{y}}(t)$ can be computed as $\widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i(t)$ at node q without extra transmissions (see also Remark 2.4).

neighbors from which it did not receive a token. This process continues until all nodes have received a token. If a node receives multiple tokens, it only connects to the parent node from which it received the token first (in case of a tie, it makes a random choice).

The full TI-DASF algorithm is given in Algorithm 2 and convergence analyses will be provided in Chapter 3. The same generalizations as explained in Section 2.3.3 apply for TI-DASF as well.

Remark 2.7. In the TI-DASF algorithm, each node k transmits only NQ samples of $\mathbf{y}_k$ per iteration, which is independent of the number of neighbors or the total number of nodes in the network. As opposed to the fully-connected case, the TI-DASF algorithm can reduce the total communication burden, even when $Q \geq M_k$. This is because naively relaying all the raw sensor data to a fusion center node for centralized processing would require most nodes to send more than NQ samples as they would also have to forward the data from their neighbors, and their neighbors' neighbors, etc (see Figure 2.4b). Obviously, such a relaying approach does not scale well with the network size, whereas the per-node bandwidth requirements in the TI-DASF algorithm are independent of the number of nodes.

Remark 2.8. We note that FC-DASF (Algorithm 1) is a special case of TI-DASF (Algorithm 2). Therefore, in the following sections of this text, we will not make the distinction between the two algorithms and simply refer to TI-DASF (Algorithm 2) as the DASF algorithm.

Remark 2.9. As mentioned in Section 2.3, we can consider a scenario where the number of channels $\widehat{M}$ of the compressed signals transmitted by each node is greater than Q, if the bandwidth constraints allow it. In that case, the local compressors F_k^i as defined in (2.9) can be taken as $F_k^i = [X_k^i, W_k^i] \in \mathbb{R}^{M_k \times \widehat{M}}$ for each node k, where the matrix $W_k^i \in \mathbb{R}^{M_k \times (\widehat{M}-Q)}$ corresponds to the filters of the additional channels. This creates additional degrees of freedom as the new $G-$matrices will be $\widehat{M} \times Q$ and can therefore improve the convergence speed of the DASF algorithm, which can also be explained by the fact that the compression ratio is smaller compared to the case $\widehat{M} = Q$. A possible solution for the matrix W_k^i at node k is to select its columns to be orthogonal to those of X_k^i. To guarantee convergence, the selection of W_k^i should deterministically depend on X_k^i, such that convergence of X_k^i implies convergence of W_k^i.

2.5 Examples and simulations

In this Section, we present examples of problems that fit the DASF framework and demonstrate the performance of the algorithm in various settings to gain

Table 2.3: Network topologies used in the simulations.

Name	Description	Acronym
Fully-connected networks	Networks where every node is linked to every other node.	FC
Randomly generated trees	Trees where each node has between 0 and 4 children with 1.7 children on average. The number of children nodes is selected randomly from $[0, 1, 2, 3, 4]$, which is distributed following the probability vector $[0.2, 0.3, 0.2, 0.2, 0.1]$.	Rand
Line graphs	Networks where each node has two neighbors except two which have a single neighbor.	Line
Erdős-Rényi random graphs	Networks generated using the Erdős-Rényi model, specified by the parameter p corresponding to the probability that any pair of nodes are connected.	ER(p)

insights into the convergence behavior as a function of Q, K, and the topology. The examples also serve as illustrations to familiarize the reader with how to recognize SFO problems of the form (2.3) or (2.5), and how to translate these into the DASF framework. It is noted that several existing distributed algorithms can be shown to be special cases of the proposed unified DASF framework (Table 2.1). Since these special cases have been validated already, and to show the generalizing properties of the framework, we will validate it on a few new problems that fit our framework. For this purpose, we have also published a companion toolbox [110] which allows to automatically generate and simulate a distributed algorithm for any arbitrary problem of the form (2.3) or (2.5). The only requirement is that the user provides a solver for the centralized problem (2.3) or (2.5). The software will use this solver to compute the updating step in each iteration of the DASF algorithm, as the update requires solving a compressed local instance of (2.3) or (2.5).

Throughout the experiments presented in this section, and in the ones of the following chapters, we will use various network topologies to compare convergence results. Therefore, we give in Table 2.3 a short description of every network topology we will use along with their acronyms used in the legends of the figures. Additionally, $\mathcal{T}^i(\cdot, q)$ is taken to be the shortest path pruning function for every experiment described in this work.

In this section, we consider two different sensor signals $\mathbf{y}$ and $\mathbf{v}$ following the mixture model given by

$$\mathbf{y}(t) = \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{2.55}$$

$$\begin{aligned}\mathbf{v}(t) &= \Pi_r \cdot \mathbf{r}(t) + \mathbf{y}(t) \\ &= \Pi_r \cdot \mathbf{r}(t) + \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t),\end{aligned} \tag{2.56}$$

with $\mathbf{r}(t)$, $\mathbf{s}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_r^2)$, $\mathbf{n}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_n^2)$ for every entry and time instant t. Π_s and Π_r are mixture matrices of which the entries will be independent of the time t and drawn from the uniform distribution within the interval $[-0.5, 0.5]$ throughout Sections 2.5.1 to 2.5.3, where we will assume stationarity in the experimental settings. An adaptive setting will be considered in Section 2.5.4, where the matrix Π_s will be time-dependent. We assume that both $\mathbf{y}$ and $\mathbf{v}$ are observable at the nodes (this is possible, e.g., if the source $\mathbf{r}$ has an on-off behavior). In all the simulations, we take the number of samples of the signals to be communicated between the nodes to be $N = 10^4$ and each node has an equal number of channels $M_k = M/K$ (where the total number of channels M and the total number of nodes K will vary). The convergence of the DASF algorithm is assessed by tracking the median squared (normalized) error (MedSE) ϵ:

$$\epsilon(i) = \text{median}\left(\frac{\|X^i - X^*\|_F^2}{\|X^*\|_F^2}\right), \tag{2.57}$$

where the median is taken over multiple Monte Carlo runs. In these experiments, we fix the number of runs to 100. The optimal value X^* is computed by solving the problems we present in the following paragraphs using centralized solvers. In case the centralized problem has multiple possible solutions, i.e., $|\mathcal{X}^*| > 1$, we select $X^* \in \mathcal{X}^*$ in (2.57) that best matches X^i in the final iteration of the simulation. This resolves the ambiguity of the solution, while the plots would still reveal non-convergence in case the algorithm would arrive in a limit cycle that switches between multiple accumulation points, i.e., we would observe subsequences of $(X^i)_i$ converging to different solutions of the problem. The parameters chosen for each following problem in Sections 2.5.1 to 2.5.3 are summarized in Table 2.4. In these experiments, we aim to observe the theoretical convergence result of the DASF algorithm and therefore consider stationary and ergodic signals. On the other hand, Section 2.5.4 has a slightly different experimental setting as we aim to demonstrate the adaptive properties of the DASF algorithm, in which case stationarity does not hold.

Table 2.4: Summary of parameters used in the simulations of the DASF algorithm.

Experiment	Section 2.5.1	Section 2.5.2	Section 2.5.3
Q	3	5	$\{1, 3, 5, 7\}$
K	$\{10, 25, 50\}$	30	
M, M_k	$M = 450$, $M_k = M/K$, $\forall k \in \mathcal{K}$		
Signal statistics	$\sigma_r^2 = \sigma_n^2 = 1$	$\sigma_r^2 = 0.5$, $\sigma_n^2 = 0.1$	$\sigma_r^2 = \sigma_n^2 = 1$
N	10000		
Monte Carlo runs	100		

2.5.1 Quadratically constrained quadratic problem

In this subsection, we will solve the following problem:

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \frac{1}{2}\mathbb{E}[\|X^T\mathbf{y}(t)\|^2] - \text{tr}(X^T A) \\ \text{subject to} \quad & \text{tr}(X^T X) \leq \alpha^2,\ X^T\mathbf{c} = \mathbf{d}, \end{aligned} \tag{2.58}$$

where we take $\sigma_n^2 = \sigma_r^2 = 1$ for the noise and signal variance. In this problem, we have three deterministic inner products of the form $X^T B$ where B is known a priori. Two of them are the terms $X^T A$ and $X^T\mathbf{c}$ where $A \in \mathbb{R}^{M \times Q}$ and $\mathbf{c} \in \mathbb{R}^M$. The third one comes from $X^T X$, which can be written as $(X^T I_M) \cdot (X^T I_M)^T$ which reveals the term $X^T B$ with $B = I_M$ (see also Remark 2.5). The values of $\alpha \in \mathbb{R}$ and $\mathbf{d} \in \mathbb{R}^Q$ have been chosen randomly while ensuring that $\alpha^2 \geq \|\mathbf{d}\|^2/\|\mathbf{c}\|^2$ which would otherwise make the problem infeasible (see Appendix B.7 for further details on this problem). We can write $\mathbb{E}[\|X^T\mathbf{y}(t)\|^2] = \text{tr}(X^T R_{\mathbf{yy}} X)$ where $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ is the covariance matrix of the signal $\mathbf{y}$. We note that the matrix $R_{\mathbf{yy}}$ is assumed to be unknown, as the signal statistics are to be learned by the algorithm, so it should not be seen as a deterministic term. Then, the local problem at iteration i that node q needs to solve is

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \frac{1}{2}\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} \widetilde{X}_q) - \text{tr}(\widetilde{X}_q^T \widetilde{A}_q^i) \\ \text{subject to} \quad & \text{tr}(\widetilde{X}_q^T C_q^{iT} C_q^i \widetilde{X}_q) \leq \alpha^2,\ \widetilde{X}_q^T \widetilde{\mathbf{c}}_q^i = \mathbf{d}, \end{aligned} \tag{2.59}$$

where $R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} = \mathbb{E}[\widetilde{\mathbf{y}}_q^i(t)\widetilde{\mathbf{y}}_q^{iT}(t)]$ is the correlation matrix of the locally available signal $\widetilde{\mathbf{y}}_q^i$ at node q and iteration i. Note that the compression matrix C_q^i

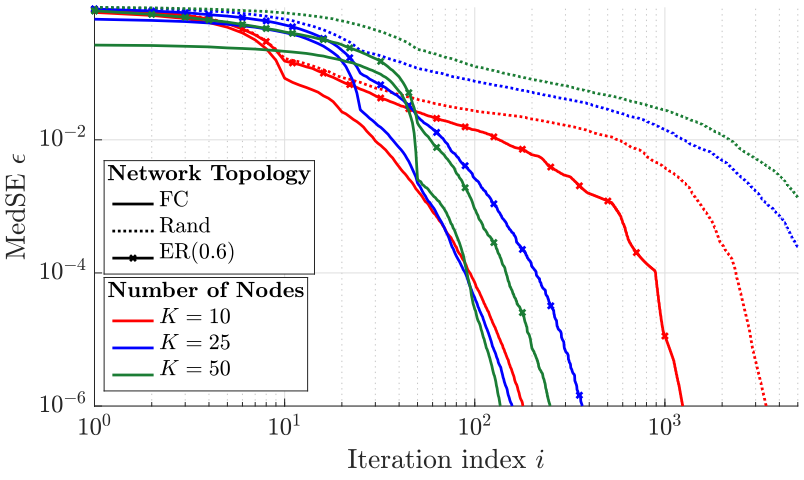


Figure 2.5: Convergence comparison of the DASF algorithm solving (2.58) on various network topologies (see Table 2.3 for a description) and for various network sizes.

appears in the quadratic constraints, which is indeed what one obtains when computing $\widetilde{B}_q^i$ with $B = I_M$, which directly follows from (2.26).

For this experiment, we take the number of channels to be $M = 200$, take $Q = 3$ and look at the behavior of the algorithm for a varying number of nodes in the network with $K \in \{10, 25, 50\}$ (the number of channels per node $M_k = M/K$ therefore changes for each K). The results are shown in Figure 2.5. For the fully-connected case, we see that the smaller networks, i.e., when K is small, converge faster in the first iterations. This is because they are able to do a full round update (over all nodes) in a smaller number of iterations compared to larger networks. However, the larger networks can eventually "catch up" and even surpass the convergence speed of their counterparts with smaller K after a certain number of iterations due to the larger number of degrees of freedom (i.e., a larger number of G_k matrices in each iteration). This property is however not observed for randomly generated trees as here the number of G_k matrices at each node is related to its number of neighbors, hence independent of the network size. Overall, the fully-connected topologies lead to faster convergence and the randomly generated trees are the slowest, which is consistent with the expectations based on the number of degrees of freedom (i.e., the number of G_k matrices in each update). Networks for which the topology is neither a tree nor fully-connected are expected to fall in between these two extreme cases.

2.5.2 The trace ratio optimization problem

We now consider the problem:

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & \frac{\mathbb{E}[\|X^T\mathbf{v}(t)\|^2]}{\mathbb{E}[\|X^T\mathbf{y}(t)\|^2]} = \frac{\text{tr}(X^T R_{\mathbf{vv}} X)}{\text{tr}(X^T R_{\mathbf{yy}} X)} \\ \text{subject to} \quad & X^T X = I_Q, \end{aligned} \tag{2.60}$$

often referred to as the trace ratio or trace quotient optimization (TRO) problem [94], which we will study in detail in Chapter 6. $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ are again spatial correlation matrices of $\mathbf{y}$ and $\mathbf{v}$ respectively, assumed to be unknown to the DASF algorithm. We take the signal and noise variance to be $\sigma_r^2 = 0.5$ and $\sigma_n^2 = 0.1$ respectively. Problem (2.60) can be solved using the solver in [94] by iteratively computing generalized eigenvalue decompositions.

At updating node q and iteration i, (2.60) is translated to

$$\begin{aligned} \underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{maximize}} \quad & \frac{\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}\widetilde{X}_q)}{\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q)} \\ \text{subject to} \quad & \widetilde{X}_q^T C_q^{iT} C_q^i \widetilde{X}_q = I_Q, \end{aligned} \tag{2.61}$$

where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ and $R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}$ are estimated in the same way as in the previous example. As detailed in Remark 2.5, the expression $\widetilde{X}_q^T C_q^{iT} C_q^i \widetilde{X}_q$ in the constraint is the local equivalent of $X^T\Gamma X$ when $\Gamma = I_M$ (or equivalently $(X^TB)\cdot(X^TB)^T$ for $B = I_M$). Therefore, the solver in [94] can be applied to solve the local problem (2.61) in step 4a of Algorithm 2, by replacing $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ by $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ and $R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}$ respectively, where q is the updating node at iteration i.

The experimental results are shown in Figure 2.6, where we highlight the differences in convergence depending on the topology of the network. In particular, we look at fully-connected networks, random trees, line topologies, and Erdős-Rényi models, generated using [119]. In each case, we keep the number of nodes, the number of channels, and the compression dimension Q constant, with $K = 30$, $M = 450$, and $Q = 5$ respectively. We observe that the more the network is connected the faster the DASF algorithm converges, the fastest being the fully-connected networks, while the slowest convergence is obtained for line graphs. This is again in line with the results and discussion on the degrees of freedom in Section 2.5.1.

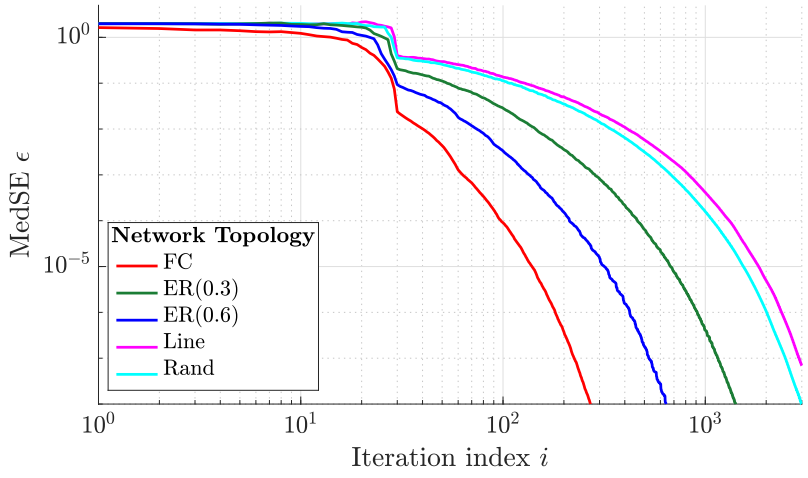


Figure 2.6: Convergence comparison of the DASF algorithm solving (2.60) for various network topologies (see Table 2.3 for a description).

2.5.3 Quadratic problem with a spherical constraint

Let us now consider:

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \frac{1}{2}\mathbb{E}[\|X^T \mathbf{y}(t)\|^2] + \operatorname{tr}(X^T A) \\ \text{subject to} \quad & \operatorname{tr}(X^T X) = 1, \end{aligned} \tag{2.62}$$

where $\sigma_n^2 = \sigma_r^2 = 1$ and the elements of A have been chosen independently at random. Note that this problem differs from (2.58) in the sense that it has a non-convex constraint set due to the non-linear equality constraint (see final paragraph in Appendix B.7). The local problem at node q and iteration i can be written as

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \frac{1}{2}\operatorname{tr}(\widetilde{X}_q^T R^i_{\tilde{\mathbf{y}}_q \tilde{\mathbf{y}}_q} \widetilde{X}_q) + \operatorname{tr}(\widetilde{X}_q^T \widetilde{A}_q^i) \\ \text{subject to} \quad & \operatorname{tr}(\widetilde{X}_q^T C_q^{iT} C_q^i \widetilde{X}_q) = 1. \end{aligned} \tag{2.63}$$

For this experiment, we fix $K = 30$, $M = 450$ and consider various number of filters $Q \in \{1, 3, 5, 7\}$. We observe in Figure 2.7 that the larger the number of filters Q, the faster the algorithm converges. This is expected as a larger value for Q implies that more information can be used at every node, however at the expense of a larger communication cost. As for the previous examples,

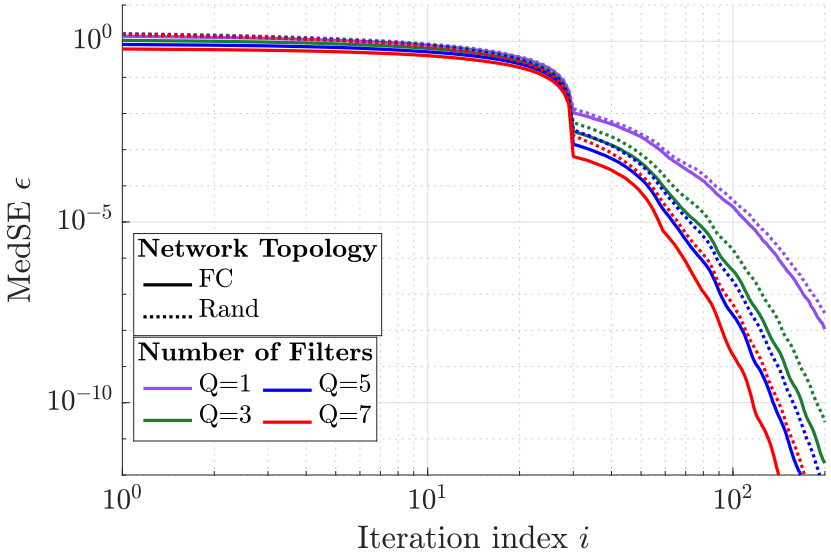


Figure 2.7: Convergence comparison of the DASF algorithm solving (2.62) on two network topologies (see Table 2.3 for a description) and for various number of filters Q.

we see that the DASF framework converges faster for fully-connected networks compared to randomly generated trees for Problem (2.62).

2.5.4 DASF in a tracking problem

In this final experiment, we consider the MMSE problem

$$\underset{\mathbf{x}\in\mathbb{R}^M}{\text{minimize}} \quad \mathbb{E}[|s(t) - \mathbf{x}^T\mathbf{y}(t)|^2], \tag{2.64}$$

where s is a one-dimensional signal, implying $Q = 1$. The problem is solved using the DASF algorithm on a randomly generated Erdős-Rényi network with connection probability 0.8. The network contains $K = 10$ nodes, each measuring a signal $\mathbf{y}_k$ with $M_k = 4$ channels. At each iteration i, $N = 10^4$ new time samples of $\mathbf{y}_k$ are used, such that $i = \lfloor t/N \rfloor$. The network-wide signal is given by

$$\mathbf{y}(t) = \mathbf{p}(t) \cdot s(t) + \mathbf{n}(t), \tag{2.65}$$

for each t, with $s(t) \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$ and $\mathbf{n}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.1)$ for every entry and time instant t. The main difference with (2.55) is that now the steering vector $\mathbf{p}$

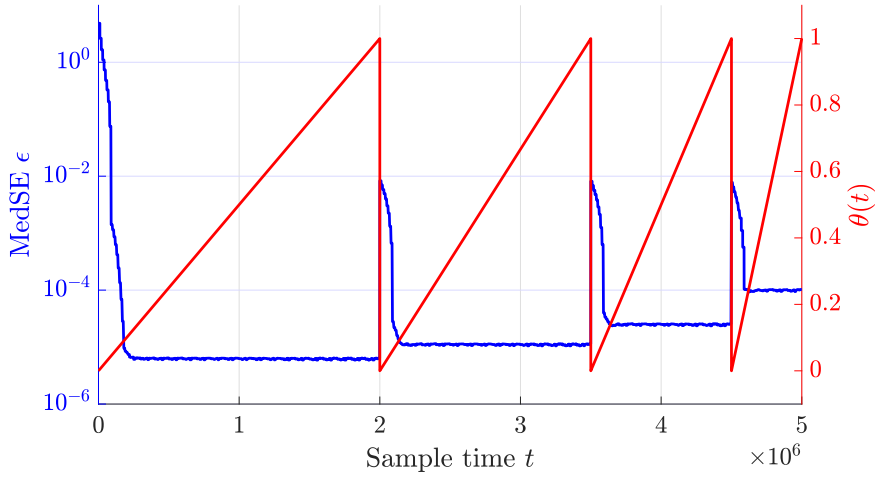


Figure 2.8: MedSE ϵ over time of the DASF algorithm in an adaptive setting (blue). The signal statistics change over time and depend on the function θ, represented in red. The relationship between the time t and the iterations i is given by $i = \lfloor t/N \rfloor$.

(corresponding to the mixture matrix Π_s in (2.55)) changes with time, implying that the statistical properties of $\mathbf{y}$ are time-dependent as well. At each time instant t, the solution of (2.64) is given by $\mathbf{x}^*(t) = (R_{\mathbf{yy}}(t))^{-1}\mathbf{r}_{\mathbf{y}s}(t)$, where $R_{\mathbf{yy}}(t) = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ and $\mathbf{r}_{\mathbf{y}s}(t) = \mathbb{E}[\mathbf{y}(t)s(t)]$ (see Appendix B.1). For each t, we have $\mathbf{p}(t) = (1-\theta(t)) \cdot \mathbf{p}_0 + \theta(t) \cdot (\mathbf{p}_0 + \Delta)$, where $\mathbf{p}_0$ and Δ are time-independent vectors of which the entries are drawn from $\mathcal{N}(0,1)$ and $\mathcal{N}(0,0.01)$ respectively.

The function θ used in this experiment is represented in Figure 2.8, where it can be observed that smooth, as well as abrupt, changes are modeled. At each iteration, the ground-truth solution of (2.64) is estimated as

$$\mathbf{x}^{*i} = (\widehat{R}^i_{\mathbf{yy}})^{-1}\widehat{\mathbf{r}}^i_{\mathbf{y}s}, \tag{2.66}$$

where

$$\widehat{R}^i_{\mathbf{yy}} = \frac{1}{N}\sum_{\tau=iN}^{iN+N-1}\mathbf{y}(\tau)\mathbf{y}^T(\tau),\ \widehat{\mathbf{r}}^i_{\mathbf{y}s} = \frac{1}{N}\sum_{\tau=iN}^{iN+N-1}\mathbf{y}(\tau)s(\tau). \tag{2.67}$$

Figure 2.8 shows the MedSE ϵ taken over 100 Monte Carlo runs:

$$\epsilon(i) = \text{median}\left(\frac{\|\mathbf{x}^i - \mathbf{x}^{*i}\|^2}{\|\mathbf{x}^{*i}\|^2}\right) \tag{2.68}$$

over time, where $i = \lfloor t/N \rfloor$. The algorithm can adapt to abrupt changes in signal statistics, i.e., when the value of θ suddenly changes, which translates to an initial jump in the error followed by a decrease. We observe that the DASF algorithm is also able to track slow changes in the statistical properties of the signal, shown by constant error values ϵ over the iterations, despite changes in the value of θ. Since the optimal solution $\mathbf{x}^{*i}$ changes at each iteration, the error ϵ settles around a certain threshold, which is higher for larger rates of change in θ and due to the approximation error.

2.6 Conclusion

In this chapter, we have proposed the DASF framework which contains a large number of well-known distributed spatial filter design problems and algorithms as a special case. For intelligibility purposes, we have first addressed the case of fully-connected networks (FC-DASF) and then generalized it to any network represented by a connected graph (TI-DASF). In order to reduce the communication burden, the nodes only communicate compressed data across the network. An interesting property of the resulting distributed algorithm is that the local problem to be solved at an updating node has the same structure as the original network-wide centralized problem, such that the same solver can be used. The convergence properties of the algorithm have been illustrated by means of four example instances of the (D)SFO problem (2.3) or its more general form in (2.5), one of the examples focusing on the adaptive properties of the DASF algorithm. A formal convergence analysis with convergence and optimality proofs, including examples of how the convergence conditions can be checked, will be provided in the next chapter. We have also provided a toolbox to automatically design and test the DASF algorithm for any user-defined instances of the (D)SFO problem (2.3) or (2.5), available in [110].

3 | The distributed adaptive signal fusion framework: Convergence properties

This chapter is largely 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.

All three co-authors contributed equally to the development of the convergence proof.

ABSTRACT | This chapter studies the convergence conditions and properties of the distributed adaptive signal fusion (DASF) algorithm, the framework itself having been introduced in Chapter 2. The DASF algorithm can be used to solve linear signal and feature fusion optimization problems in a distributed fashion, and is in particular well-suited for solving spatial filtering optimization problems encountered in wireless sensor networks. The convergence conditions and results are provided along with rigorous proofs and analyses, as well as various example problems to which they apply. Additionally, we describe procedures that can be added to the DASF algorithm to ensure convergence in specific cases where some of the technical convergence conditions are not satisfied.

3.1 Introduction

In this chapter, we provide a set of sufficient conditions for convergence and optimality of the DASF algorithm introduced in Chapter 2, based on which we can show that the DASF algorithm converges to the centralized solution of the problem despite the compression, as if all the raw sensor data were centrally available. The technical conditions required for convergence are akin to the well-known linear independence constraint qualifications in the optimization literature, which in the case of DASF lead to an upper bound on the number of constraints the global (centralized) problem is allowed to have. Furthermore, since the local problems in each node are compressed versions of the global problem, we assume that the local problems satisfy the same assumptions as the global problem as outlined in Section 2.2.3 of Chapter 2, which is generally the case as they are directly inherited. Finally, we impose a condition on the finiteness of the number of possible solutions that are achievable by the solver used to solve the local problems in each node. We will see that these conditions are often satisfied for spatial filtering and signal fusion problems in practical scenarios. Furthermore, we provide several examples and illustrations of how the convergence conditions can be checked either a priori or during operation of the algorithm. We will also show how the insights obtained from the convergence analysis can be used to design new strategies to enforce convergence in cases where a violation of the convergence conditions is detected.

The outline of the chapter is as follows. After a short review of the DASF framework in Section 3.2, we study the convergence and optimality guarantees of the DASF algorithm in Section 3.3. In particular, we show that under some technical conditions, accumulation points of the sequence of points produced by the algorithm are also fixed points, and that fixed points are solutions of the centralized problem. Examples of typical spatial filtering and signal fusion problems, such as minimum mean square error or minimum variance beamforming, and how the convergence conditions apply to these cases are discussed in Section 3.4. Finally, in the contrived cases where some of the technical requirements are violated, we describe methods to still achieve convergence for the DASF algorithm in Section 3.5. Conclusions are drawn in Section 3.6.

3.2 Review of the DASF framework

In this section, we briefly restate the scope and operation of the DASF framework, which was extensively described in Chapter 2. We do this for completeness and

with the purpose to re-introduce key equations and introduce some new ones that will be needed in the convergence analysis.

3.2.1 Problem description and assumptions

We consider a WSN consisting of a set of K nodes $\mathcal{K} = \{1, \ldots, K\}$ interconnected according to a network graph $\mathcal{G}$, which contains edges between nodes that are able to share data with each other. Each node k collects observations of an M_k-channel sensor signal $\mathbf{y}_k$. We define the network-wide sensor signal $\mathbf{y} \in \mathbb{R}^M$ as

$$\mathbf{y}(t) = [\mathbf{y}_1^T(t), \ldots, \mathbf{y}_K^T(t)]^T, \tag{3.1}$$

where t denotes the time index and $M = \sum_{k\in\mathcal{K}} M_k$. $\mathbf{y}$ is assumed to be (short-term) stationary and ergodic, such that its statistical properties can be properly estimated given a sufficiently large number of samples at different time instants, e.g., $\{\mathbf{y}(\tau)\}_{\tau=0}^{N-1}$, where N denotes the number of time samples. Our objective is to find a linear filter $X \in \mathbb{R}^{M\times Q}$ that optimizes in some sense the output signal of the linear signal fusion $X^T\mathbf{y}$. More specifically, we consider problems of the following form:

$$\begin{aligned} \mathbb{P}: \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \varphi\left(X^T\mathbf{y}(t), X^TB\right) \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) \leq 0 \;\; \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) = 0 \;\; \forall j \in \mathcal{J}_E, \end{aligned} \tag{3.2}$$

where φ and η_j's are differentiable real- and scalar-valued functions, introduced and referred to as ((D)SFO) problems in Chapter 2. The indices $j \in \mathcal{J}_I$ and $j \in \mathcal{J}_E$ again correspond to inequality and equality constraints respectively, and the joint index set is $\mathcal{J} = \mathcal{J}_I \cup \mathcal{J}_E$, such that the total number of constraints is $J = |\mathcal{J}|$.

B is again an additional deterministic parameter of the problem, partitioned as

$$B = [B_1^T, \ldots, B_K^T]^T \in \mathbb{R}^{M\times L} \tag{3.3}$$

where L is some problem-dependent constant. It is assumed that each node k has local access to its corresponding block B_k (i.e., they can be used in computations without being first requested from another node). In contrast to the signal $\mathbf{y}$, the matrix B is deterministic and independent of time. As noted in Section 2.3.3, Problem (3.2) can be generalized to more than one variable X, stochastic signal $\mathbf{y}$ and deterministic term B. Note that such a generalization covers deterministic quadratic terms in the form X^TAX, since two linear terms $X^TB^{(1)}$ and $X^TB^{(2)}$ can be combined into $(X^TB^{(1)}) \cdot (X^TB^{(2)})^T$, with $A = B^{(1)}B^{(2)T}$.

As the actual optimization variable is X, and by removing the time-dependence of the problem by stationarity of the random signal $\mathbf{y}$, we define the functions f and h_j, $j \in \mathcal{J}$, which express the objective and constraints as a function of X only:

$$f(X) \triangleq \varphi\left(X^T\mathbf{y}(t), X^T B\right), \tag{3.4}$$

$$h_j(X) \triangleq \eta_j\left(X^T\mathbf{y}(t), X^T B\right), \ \forall j \in \mathcal{J}, \tag{3.5}$$

which are assumed to be continuously differentiable with respect to the variable X over their respective domains. This allows us to write (3.2) compactly as

$$\begin{aligned} \mathbb{P}: \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & f(X) \\ \text{subject to} \quad & h_j(X) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & h_j(X) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{3.6}$$

Furthermore, we denote the constraint set of (3.2) or (3.6) as $\mathcal{S}$, the complete solution set as $\mathcal{X}^*$ and a single solution as X^*, i.e., $X^* \in \mathcal{X}^*$.

In order to guarantee the theoretical convergence of the DASF algorithm, we restrict its application to problems satisfying the following three general assumptions[1] (in Section 3.4, we explain how these assumptions can be checked in practical examples):

Assumption 1. The targeted instance of Problem (3.2) or (3.6) is well-posed, in the sense that the solution set is not empty and varies continuously with a change in the parameters of the problem.

The notion of (generalized Hadamard) well-posedness we require is based on [116,117]. The main difference is that we require the map from the space of inputs of the problem to the solution space to be continuous instead of upper semicontinuous, which is required for the convergence proof. We formally define the continuity of this map in Section 3.3. Even though this condition might seem restrictive, it applies to many practical instances of the problems of interest (see Section 3.4).

[1]Throughout this text, if assumptions or conditions are labeled as "**Xa**", it implies that this assumption/condition can be replaced with a different assumption/condition "**Xb**". When we only mention the label "**X**", we refer to either of the two.

Assumption 2. The linear independence constraint qualifications (LICQ) [118] hold at the solutions of Problem (3.2) or (3.6), i.e., the solutions satisfy the KKT conditions.

If X^* is a solution of Problem (3.2) or (3.6), Assumption 2 implies that, for $j \in \mathcal{J}^*$, the gradients $\nabla_X h_j(X^*)$ are linearly independent, where $\mathcal{J}^* \subseteq \mathcal{J}$ is the set of all indices j satisfying $h_j(X^*) = 0$. If the problem is unconstrained, we have $\nabla_X f(X^*) = 0$.

Assumption 3a. f has compact sublevel sets in $\mathcal{S}$, i.e., for all $m \in \mathbb{R}$, $\{X \in \mathcal{S} \mid f(X) \leq m\}$ is compact, i.e., closed and bounded.

It is noted here that this assumption can be further relaxed to an alternative Assumption 3b presented below. This relaxed version only requires that the DASF algorithm's initialization point is in a compact sublevel set, in which case not all sublevel sets of f in $\mathcal{S}$ should be compact.

Assumption 3b. The sublevel set of f $\{X \in \mathcal{S} \mid f(X) \leq f(X^0)\}$ corresponding to X^0 is compact.

As will be shown in Section 3.3.1, Assumption 3a or 3b is needed to ensure that the elements of the sequence $(X^i)_i$ obtained from the DASF algorithm lie in a compact set, which is required to show convergence.

In the remaining parts of this chapter, problems $\mathbb{P}$ which can be written as (3.2) or (3.6) will be satisfying the assumptions above, i.e., we will not repeat these assumptions in any of the convergence theorems. As discussed in Section 2.2.3, we also implicitly assume that there exists a centralized solver able to solve the targeted problem instance $\mathbb{P}$, which will be used by the DASF algorithm to solve local per-node compressed versions of the centralized problem $\mathbb{P}$.

3.2.2 The DASF algorithm

In order to solve Problem (3.2) in a distributed setting, we consider a partitioning of the global variable X into local variables:

$$X = [X_1^T, \ldots, X_K^T]^T, \tag{3.7}$$

where each local variable $X_k \in \mathbb{R}^{M_k \times Q}$ is assigned to a single node k. At any given iteration i, the nodes k all have an estimation X_k^i of their local variable X_k. At each iteration i, some node $q \in \mathcal{K}$ is selected to be the "updating node", and will solve a compressed version of Problem (3.2), which will be defined later in this section. We note that we choose a different updating node at each iteration. As the network graph $\mathcal{G}$ can contain loops, we prune the network into a tree, which we denote as $\mathcal{T}^i(\mathcal{G}, q)$, such that there is a unique path from any node $k \neq q$ to the updating node q (as explained in Section 2.4.3, the tree $\mathcal{T}^i(\mathcal{G}, q)$ should preserve all the neighbors of node q to maximize the degrees of freedom in the updating process). In the remaining parts of this chapter, the set $\mathcal{N}_k$ refers to the neighbors of node k in the pruned network, i.e., with respect to the graph $\mathcal{T}^i(\mathcal{G}, q)$.

At the beginning of each iteration, every node compresses N samples of its local signal $\mathbf{y}_k$ using its current estimate X_k^i of the local variable X_k to obtain the compressed $Q-$channel local signal:

$$\widehat{\mathbf{y}}_k^i(t) \triangleq X_k^{iT} \mathbf{y}_k(t) \in \mathbb{R}^Q, \tag{3.8}$$

while a similar operation is done to compress each B_k:

$$\widehat{B}_k^i \triangleq X_k^{iT} B_k \in \mathbb{R}^{Q \times L}. \tag{3.9}$$

A decentralized fuse-and-forward process is then started, in which the compressed data from all the nodes is summed on its way towards node q. This fuse-and-forward flow arises naturally when using a recursive computation rule, as explained in Section 2.4.2. Indeed, the data that node k sends to its neighbor n (which is closest to the updating node q) is defined as

$$\widehat{\mathbf{y}}_{k\to n}^i(t) \triangleq X_k^{iT} \mathbf{y}_k(t) + \sum_{l \in \mathcal{N}_k \setminus \{n\}} \widehat{\mathbf{y}}_{l\to k}^i(t), \tag{3.10}$$

which can be computed as soon as node k receives $\widehat{\mathbf{y}}_{l\to k}$ from all its neighbors $l \in \mathcal{N}_k$, except node n. We see that this recursive expression starts at leaf nodes (nodes with only one neighbor) and extends naturally to node q, which eventually receives

$$\widehat{\mathbf{y}}_{n\to q}^i(t) = X_n^{iT} \mathbf{y}_n(t) + \sum_{k \in \mathcal{N}_n \setminus \{q\}} \widehat{\mathbf{y}}_{k\to n}^i(t) = \sum_{k \in \mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i(t) \tag{3.11}$$

from all its neighbors $n \in \mathcal{N}_q$. $\mathcal{B}_{nq}$ is the subgraph rooted at node q that contains $n \in \mathcal{N}_q$, i.e., the set of nodes which would be disconnected from the subgraph containing node q if we were to cut the connection between node n and q (see Figure 2.3). A similar fuse-and-forward process is performed for the terms (3.9), resulting in fused parameters $\widehat{B}_{n\to q}^i$.

The data received by node q can then be structured in order to act as a local version of the global data. For this purpose, we define the $\widetilde{M}_q-$channel signal $\widetilde{\mathbf{y}}_q^i$ at node q as the local signal $\mathbf{y}_q$ stacked with the compressed signals it receives from its neighbors, where $\widetilde{M}_q = M_q + |\mathcal{N}_q| \cdot Q$, and given by

$$\widetilde{\mathbf{y}}_q^i(t) = [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_{n_1\to q}^{iT}(t), \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}(t)]^T. \tag{3.12}$$

The matrix $\widetilde{B}_q^i \in \mathbb{R}^{\widetilde{M}_q \times L}$ is defined similarly.

Then, we define the local variable $\widetilde{X}_q$ at node q as

$$\widetilde{X}_q = [X_q^T, G_{n_1}^T, \ldots, G_{n_{|\mathcal{N}_q|}}^T]^T \in \mathbb{R}^{\widetilde{M}_q \times Q}, \tag{3.13}$$

where $X_q \in \mathbb{R}^{M_q \times Q}$ and $G_n \in \mathbb{R}^{Q\times Q}$, $\forall n \in \mathcal{N}_q$. This variable acts as a spatial fusion filter on the locally available data at node q, analogous to the way X acts on $\mathbf{y}$ and B for the global problem. From (3.12) and (3.13), we can then write

$$\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t) = X_q^T \mathbf{y}_q(t) + \sum_{n\in\mathcal{N}_q} G_n^T \widehat{\mathbf{y}}_{n\to q}^i(t), \tag{3.14}$$

and replacing the signals $\widehat{\mathbf{y}}_{n\to q}^i$ by their definition in (3.11), we have

$$\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t) = X_q^T \mathbf{y}_q(t) + \sum_{n\in\mathcal{N}_q} \sum_{k\in\mathcal{B}_{nq}} G_n^T \widehat{\mathbf{y}}_k^i(t), \tag{3.15}$$

$$= X_q^T \mathbf{y}_q(t) + \sum_{n\in\mathcal{N}_q} \sum_{k\in\mathcal{B}_{nq}} (X_k^i G_n)^T \mathbf{y}_k(t). \tag{3.16}$$

A similar expression can be derived for $\widetilde{X}_q^T \widetilde{B}_q^i$.

These local counterparts of the global expressions lead us to define the local problem at node q using the previous parameterization of X, which is a compressed version of the global problem (3.2):

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \varphi(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) \\ \text{subject to} \quad & \eta_j(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) \le 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_j(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{3.17}$$

The fact that (3.2) and (3.17) have an equivalent structure implies that the same solver can be used for both.

From (3.16), we observe that the local inner product $\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t)$ at node q is related to the network-wide inner product $X^T \mathbf{y}(t)$ if X is defined as

$$X = C_q^i \widetilde{X}_q, \tag{3.18}$$

with

$$\begin{aligned} C_q(X) &= \left[\begin{array}{c|c|c} 0 & & \\ I_{M_q} & \Theta_{-q}(X) & \\ 0 & & \end{array}\right] \in \mathbb{R}^{M \times \widetilde{M}_q}, \\ C_q^i &\triangleq C_q(X^i), \end{aligned} \tag{3.19}$$

where I_{M_q} is placed in the $q-$th block-row, and $\Theta_{-q}(X)$ is a block matrix with K block-rows and $|\mathcal{N}_q|$ block-columns. Each block-column corresponds to one of the neighbors $n \in \mathcal{N}_q$ of q, which we re-index as $m_n \in \{1, \ldots, |\mathcal{N}_q|\}$. The block at the $k-$th block-row and m_n-th block-column is then defined as

$$\Big[\Theta_{-q}(X)\Big](k, m_n) = \begin{cases} X_k & \text{if } k \in \mathcal{B}_{nq}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.20}$$

This transition matrix allows us to relate the global data or global variables with their local counterparts:

$$\widetilde{\mathbf{y}}_q^i(t) = C_q^{iT} \mathbf{y}(t), \ \widetilde{B}_q^i = C_q^{iT} B, \ X = C_q^i \widetilde{X}_q \tag{3.21}$$

and also write the local problem (3.17) in a compact way:

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & f(C_q^i \widetilde{X}_q) \\ \text{subject to} \quad & h_j(C_q^i \widetilde{X}_q) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & h_j(C_q^i \widetilde{X}_q) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{3.22}$$

Moreover, denoting the constraint set of the global problem (3.2) or (3.6) as $\mathcal{S}$ and the constraint set of the local problem (3.17) or (3.22) as $\widetilde{\mathcal{S}}_q^i$, it can be shown that (see Lemma 2.1)

$$\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i \Leftrightarrow C_q^i \widetilde{X}_q \in \mathcal{S}, \tag{3.23}$$

i.e., a point $\widetilde{X}_q$ in the constraint set of the local problem (3.17) leads to a corresponding point which by definition is also in the constraint set of the global problem (3.2). Using this notation, we define the solution of the local problem

(3.17) or equivalently (3.22) as

$$\begin{aligned}\widetilde{X}_q^* &\triangleq \underset{\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}}\ f\left(C_q^i \widetilde{X}_q\right), \\ &= \underset{\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}}\ \varphi\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right)\end{aligned} \tag{3.24}$$

Considering instances where (3.17) or (3.22) would have multiple global minima, we choose $\widetilde{X}_q^*$ as the solution minimizing the distance $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$, where

$$\widetilde{X}_q^i \triangleq [X_q^{iT}, I_Q, \ldots, I_Q]^T. \tag{3.25}$$

Finally, the matrices $G_{n_1}^{i+1}, \ldots, G_{n_{|\mathcal{N}_q|}}^{i+1}$ obtained from the partitioning (3.13) of $\widetilde{X}_q^*$ need to be disseminated in the network, so that every node can update their local estimator X_k as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q, \\ X_k^i G_n^{i+1} & \text{if } k \neq q,\ k \in \mathcal{B}_{nq},\ n \in \mathcal{N}_q, \end{cases} \tag{3.26}$$

which follows from the parameterization (3.18) of X.

This process is then repeated at a different node at each iteration, as summarized in Algorithm 3.

3.3 Convergence and optimality

In this section, we analyze the convergence of the sequence of points $(X^i)_i$ that are generated by the DASF algorithm. Note that X^i is formed by stacking all the local X_k^i's at all nodes $k \in \mathcal{K}$, or equivalently, by using (3.18) which shows that $X^i = C_q^i \widetilde{X}_q^i$. We show that under some mild technical conditions (which are generally satisfied in practice), the DASF algorithm converges to a stationary point of (3.2), and with some additional conditions even to the globally optimal signal fusion rule X^*, i.e., $(X^i)_i \to X^* \in \mathcal{X}^*$. As a result, the fused output signal $X^{iT}\mathbf{y}(t)$, which can be computed as $\widetilde{X}_k^{iT}\widetilde{\mathbf{y}}_k^i(t)$ at any node k, will become equal to $(X^*)^T\mathbf{y}(t)$ when the sample time t is large (note that the iteration index i is linked to the time index t). This convergence is without any of the nodes knowing the full signal $\mathbf{y}$ or matrix B, and despite the compression and fusion performed at each node.

For mathematical tractability, we make abstraction of estimation errors appearing from approximating the signal statistics using a finite N, which

Algorithm 3: Distributed Adaptive Signal Fusion (DASF) algorithm
Code available in [110]

X^0 initialized randomly, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) The network $\mathcal{G}$ is pruned into a tree $\mathcal{T}^i(\mathcal{G}, q)$.
  2) Every node k collects a new batch of N samples of $\mathbf{y}_k$. All nodes compress these to N samples of $\widehat{\mathbf{y}}_k^i$ as in (3.8). $\widehat{B}_k^i$ is computed using (3.9).
  3) The nodes sum-and-forward their compressed data towards node q via the recursive rule (3.10) (and a similar rule for the $\widehat{B}_k^i$'s). Node q eventually receives N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ given in (3.11) along with $\widehat{B}_{n\to q}^i$ similarly defined, from all its neighbors $n \in \mathcal{N}_q$.
  at Node q **do**
    4a) Compute $\widetilde{X}_q^*$ as the solution of (3.17) where $\widetilde{\mathbf{y}}_q^i$, $\widetilde{B}_q^i$ and $\widetilde{X}_q^i$ are defined in (3.12) and (3.13). If the solution of (3.17) is not unique, select the solution which minimizes $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$ with $\widetilde{X}_q^i$ defined as in (3.25).
    4b) Partition $\widetilde{X}_q^*$ as in (3.13).
    4c) Disseminate G_n^{i+1} to all nodes in $\mathcal{B}_{nq}$, $\forall n \in \mathcal{N}_q$.
  end
  5) Every node updates X_k^{i+1} according to (3.26).
  $i \leftarrow i + 1$

Note: The fused output signal $\widehat{\mathbf{y}}(t)$ for the current batch of N samples can be computed at node q as $\widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i(t)$.

means the proofs only hold in the asymptotic case where $N \to +\infty$. In other words, we assume that all signal statistics are perfectly estimated, which is only an approximation of the practical case where the signal statistics are (re-)estimated based on blocks of N samples. In practice, a finite N has to be used, in which case the algorithm will hover around the optimal solution due to these aforementioned estimation errors in each iteration. We note that this is also the case for the centralized equivalent of the algorithm, in case the latter has to estimate the signal statistics over finite windows, e.g., in a tracking context.

3.3.1 Convergence of the objective

The following result states the convergence of the objective function under Algorithm 3's update rule.

Theorem 3.1. Let $(X^i)_i$ be any sequence of iterates satisfying Algorithm 3's update rule for an instance $\mathbb{P}$ of (3.2) or (3.6). Then, all $(X^i)_{i>0}$ belong to the constraint set $\mathcal{S}$ of (3.6) and $\big(f(X^i)\big)_{i>0}$ is a monotonically decreasing convergent sequence.

Proof. Because $\widetilde{X}_q^*$ satisfies the constraint set $\widetilde{\mathcal{S}}_q^i$ (see (3.24)) for any update i, we conclude from (3.23) that each X^i, satisfies the constraint set $\mathcal{S}$ of the network-wide problem (3.6) for any $i \geq 1$ (as also shown in Lemma 2.1). Furthermore, since $\widetilde{X}_q^*$ is the solution of (3.24), we have $f(C_q^i \widetilde{X}_q^*) \leq f(C_q^i \widetilde{X}_q)$ for any $\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i$. In particular, this inequality is verified for $\widetilde{X}_q = \widetilde{X}_q^i$ as defined in (3.25) for which $C_q^i \widetilde{X}_q^i = X^i$. This is because $\widetilde{X}_q^i$ indeed also belongs to the constraint set $\widetilde{\mathcal{S}}_q^i$ in (3.24), because of (3.23) and the fact that X^i belongs to $\mathcal{S}$ if $i \geq 1$ (see the beginning of the proof). Then, we have $f(C_q^i \widetilde{X}_q^*) = f(X^{i+1}) \leq f(C_q^i \widetilde{X}_q^i) = f(X^i)$. Since the sequence is monotonically decreasing and since it has a lower bound (defined by the global minimum of $\mathbb{P}$), it must converge. □

Theorem 3.1 guarantees that the sequence of objective values $\big(f(X^i)\big)_i$ associated with the iterates generated by the DASF algorithm converges and that the iterates correspond to feasible points of Problem (3.2) or (3.6). However, this does not imply convergence of the sequence $(X^i)_i$ itself, which is typically much more challenging to guarantee, even for centralized optimization algorithms such as line-search or trust region methods [120–124]. Moreover, even if the convergence of $(X^i)_i$ can be proven for the DASF algorithm, we still need to show that it converges to an "interesting" point, i.e., a stationary point of Problem (3.6), and preferably a global minimum. In the next two subsections, we will introduce two technical conditions (on top of the assumptions 1-3 in Section 3.2.1) which will be combined in Section 3.3.4 to state convergence and optimality results for $(X^i)_i$. We end this subsection with a corollary of Theorem 3.1 showing that the elements of $(X^i)_{i>0}$ lie in a compact set which is required to show convergence of the DASF algorithm (see Section 3.3.3 and Section 3.D).

Corollary 3.1 (Proof omitted)**.** If either Assumption 3a or 3b is satisfied, then the elements of the sequence $(X^i)_{i>0}$ obtained from the DASF algorithm lie in a compact set.

The proof of Corollary 3.1 comes directly from the monotonic decrease of the objective obtained in Theorem 3.1.

3.3.2 Technical conditions for stationarity of fixed points

The first condition comes in two versions, where either one of the two has to be satisfied in order to prove that the fixed points of the algorithm are stationary points. These conditions are akin to the linear independence constraint qualification (LICQ) in classical optimization theory [118], and can be seen as compressed versions of these. We define a fixed point $\overline{X}$ of the DASF algorithm as a point that is invariant under a DASF update step at any updating node q, i.e., $X^{i+1} = X^i = \overline{X}$ independently of the updating node q at iteration i.

Condition 1a. For a fixed point $\overline{X}$ of Algorithm 3, the elements of the set $\{\overline{X}^T \nabla_X h_j(\overline{X})\}_{j\in\mathcal{J}}$ are linearly independent.

Since the set $\{\overline{X}^T \nabla_X h_j(\overline{X})\}_{j\in\mathcal{J}}$ consists of J matrices of size $Q \times Q$, the number of constraints J is upper bounded as

$$J \leq Q^2, \tag{3.27}$$

in order to allow the set to be linearly independent. As we will see in the proof of Theorem 3.2, this condition ensures that the Lagrange multipliers of the local problem are unique at a fixed point, eventually leading to the global optimality conditions being satisfied. Note that Condition 1a is highly likely to be satisfied in practice if (3.27) is satisfied, as a linear dependency would be highly coincidental if there are fewer matrices in the set than entries in each matrix (the points where this condition is violated is then a discrete set of points within a continuum of points). It is noted that Condition 1a can sometimes be shown to be satisfied a priori, based on the structure of the constraint set in the DSFO problem to which it is applied, without knowing the fixed points of the algorithm. The following example demonstrates how this can be proven when the constraint set is the Stiefel manifold, i.e., $X^T X = I_Q$, which is the case in, e.g., principal component analysis.

Example 3.1. Let $\mathcal{S}$ be the Stiefel manifold, i.e., we have the constraints $X^TX = I_Q$. There are $J = \frac{Q(Q-1)}{2}$ distinct constraints, where each constraint function is written as $h_{ml}(X) = \mathbf{x}(m)^T\mathbf{x}(l) - \mathbb{1}\{m = l\}$, where $\mathbf{x}(m)$ is the $m-$th column of X, $m, l \in \{1, \dots, Q\}$, $l \leq m$ and $\mathbb{1}$ is the indicator function. The derivative with respect to X can be found to be

$$\nabla_X h_{ml}(X) = \mathbf{x}(m)\mathbf{e}_l^T + \mathbf{x}(l)\mathbf{e}_m^T, \tag{3.28}$$

where the $\mathbf{e}$'s are the standard basis vectors of $\mathbb{R}^Q$. Multiplying this expression by X^T from the left and applying the constraint $X^TX = I_Q$ (assuming $X \in \mathcal{S}$), we have

$$X^T\nabla_X h_{ml}(X) = \mathbf{e}_m\mathbf{e}_l^T + \mathbf{e}_l\mathbf{e}_m^T. \tag{3.29}$$

Note that the right-hand side of (3.29) is independent of X. Since the J elements of $\{\mathbf{e}_m\mathbf{e}_l^T + \mathbf{e}_l\mathbf{e}_m^T\}_{m,l}$, $l \leq m$, are linearly independent, Condition 1a is satisfied if $X \in \mathcal{S}$. From Lemma 2.1, every X^i that is produced by Algorithm 3 belongs to $\mathcal{S}$, therefore Condition 1a is satisfied for $i > 0$.

Condition 1a allows us to state a first important result:

Theorem 3.2. Let $\mathbb{P}$ be an instance of (3.2) or (3.6). Then, if Condition 1a is satisfied, any fixed point of Algorithm 3 must be a stationary point of $\mathbb{P}$ satisfying its KKT conditions.

Proof. See Section 3.A. □

By definition, convergence of an algorithm can only be towards a fixed point of the algorithm, hence Theorem 3.2 guarantees that *if* the DASF algorithm converges, it converges to a stationary point of the global problem. For the special case of problems that are unconstrained, we do not require Condition 1a to hold (as it would lead to an empty set), in which case Theorem 3.2 can be simplified as follows.

Corollary 3.2. Let $\mathbb{P}$ be an instance of (3.2) or (3.6) which is unconstrained. Then, any fixed point of Algorithm 3 must be a stationary point of $\mathbb{P}$ satisfying its KKT conditions.

Alternatively, we propose a less restrictive – albeit more complicated – condition for cases with more constraints than Q^2 (see (3.27)), which is especially of interest for problems where $Q = 1$, for which Condition 1a would only allow a single constraint.

Condition 1b. For a fixed point $\overline{X}$ of Algorithm 3, the elements of the set $\{D_{j,q}(\overline{X})\}_{j\in\mathcal{J}}$ are linearly independent for any q, where

$$D_{j,q}(\overline{X}) \triangleq \begin{bmatrix} \overline{X}_q^T \nabla_{X_q} h_j(\overline{X}) \\ \sum_{k\in\mathcal{B}_{n_1 q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \\ \vdots \\ \sum_{k\in\mathcal{B}_{n_{|\mathcal{N}_q|} q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \end{bmatrix}, \tag{3.30}$$

which is a block-matrix containing $(1 + |\mathcal{N}_q|)$ blocks of $Q \times Q$ matrices.

For a given node q, the elements of the set $\{D_{j,q}(\overline{X})\}_{j\in\mathcal{J}}$ are now $(1+|\mathcal{N}_q|)Q\times Q$ matrices and therefore their size depends on the nodes and the topology of the network. This means that we require the number of constraints J to satisfy:

$$J \leq (1 + \min_{k\in\mathcal{K}} |\mathcal{N}_k|) \cdot Q^2. \tag{3.31}$$

This condition assumes that the pruning of the network $\mathcal{T}^i(\mathcal{G}, q)$ preserves all the links with the neighbors of the updating node q. Furthermore, the proof in Section 3.B will reveal that the number of constraints should also satisfy a second bound, which is also necessary for Condition 1b to hold:

$$J \leq \frac{Q^2}{K-1} \sum_{k\in\mathcal{K}} |\mathcal{N}_k|. \tag{3.32}$$

The reason is less obvious, but is related to specific interdependencies between the $D_{j,q}$'s across different nodes q (see Section 3.B). It is noted that both bounds (3.31)-(3.32) are necessary, i.e., satisfying the first does not necessarily imply that the second one is satisfied and vice versa.

Similarly to the previous condition, Condition 1b is typically satisfied in practice when J satisfies both bounds in (3.31)-(3.32), i.e.,

$$J \leq \min\left(\frac{Q^2}{K-1} \sum_{k\in\mathcal{K}} |\mathcal{N}_k|,\ (1 + \min_{k\in\mathcal{K}} |\mathcal{N}_k|) \cdot Q^2 \right). \tag{3.33}$$

Nevertheless, it is still possible that there exists a fixed point that is "close" to violating this condition, in which case the convergence of the DASF algorithm might become very slow if it reaches a neighborhood of such a fixed point. We refer to Section 3.5 on how to deal with these rare situations.

The following example illustrates why Condition 1b is less restrictive than Condition 1a.

Example 3.2. Suppose that $\mathcal{S} = \{X \in \mathbb{R}^{M\times Q} \mid X^T B = A,\ A \in \mathbb{R}^{Q\times L}\}$, which is a typical constraint used in LCMV beamforming [9]. We have $J = QL$ constraints and each constraint function is given by $h_{ml}(X) = \mathbf{x}(m)^T\mathbf{b}(l) - A_{ml}$, where $\mathbf{x}(m)$ is the $m-$th column of X, $\mathbf{b}(l)$ is the $l-$th column of B and A_{ml} is the entry (m,l) of the matrix A. It is straightforward to show that requiring the elements $X^T\nabla_X h_{ml}(X) = X^T\mathbf{b}(l)\mathbf{e}_m^T$ to be linearly independent for every m and l, i.e., satisfying Condition 1a, is equivalent to requiring $\text{rank}(B) = L$. If $Q < L$, this condition cannot be satisfied. Looking now at Condition 1b, we have $X_k^T\nabla_{X_k} h_{ml}(X) = X_k^T\mathbf{b}_k(l)\mathbf{e}_m^T$ for every $k \in \mathcal{K}$, where $\mathbf{b}_k(l)$ is the block of $\mathbf{b}(l)$ corresponding to node k. Then, we can show that satisfying Condition 1b is equivalent to requiring that the matrix

$$[B_q^T X_q^i, \sum_{k\in\mathcal{B}_{n_1 q}} B_k^T X_k^i, \ldots, \sum_{k\in\mathcal{B}_{n_{|\mathcal{N}_q|} q}} B_k^T X_k^i]^T \tag{3.34}$$

has rank L at a fixed point X^i, where the B_k's are obtained from the partitioning of B as in (3.3). This is possible even when $Q < L$, i.e., if node q has sufficient neighbors such that $(1 + |\mathcal{N}_q|)\cdot Q \geq L$.

Theorem 3.3. Let $\mathbb{P}$ be an instance of (3.2) or (3.6). Then, if Condition 1b is satisfied, a fixed point of Algorithm 3 must be a stationary point of $\mathbb{P}$ satisfying its KKT conditions.

Proof. See Section 3.B. □

Finally, we note that these conditions are complementary in the sense that Condition 1a is not necessary for Condition 1b to hold, and vice versa.

3.3.3 Technical conditions for convergence

Conditions 1a and 1b are sufficient to show that fixed points of the DASF algorithm are stationary points. The next step is to show that accumulation points[2] of the algorithm are fixed points (and therefore stationary points of (3.2) or (3.6)), for which we require a second condition.

Condition 2. The local problems (3.17) or (3.22) satisfy Assumptions 1 to 3.

It is important to note here that this condition is usually satisfied in practice because the local problems have the same structure as the global problem (3.2), which was already assumed to satisfy Assumptions 1 to 3. It is therefore reasonable to assume that these local problems also inherit these same properties. In Section 3.4, we will give several examples to illustrate how Condition 2 can be checked in various problems. We will also present some examples of rare cases where this condition is not satisfied and provide fixes for it.

The well-posedness of the problem as required in Assumption 1 requires a continuity assumption on the point-to-set mapping from the space of inputs of the problem to its solution space. Formally, let $\widetilde{\mathcal{F}}_q : \mathbb{R}^{M\times Q} \rightrightarrows \mathbb{R}^{\widetilde{M}_q\times Q}$ be the following point-to-set mapping:

$$\widetilde{\mathcal{F}}_q(X) \triangleq \underset{\widetilde{W}_q : C_q(X)\widetilde{W}_q \in \mathcal{S}_q(X)}{\operatorname{argmin}} f(C_q(X)\widetilde{W}_q), \tag{3.35}$$

where $C_q(X)$ is defined in (3.19) and $\mathcal{S}_q$ is the point-to-set mapping corresponding to the local parameterized constraint set for X when node q is the updating node:

$$\mathcal{S}_q(X) = \{W \in \mathcal{S} \mid W_k \in \mathcal{C}(X_k), \quad \forall k \neq q\} \tag{3.36}$$

with the subscript k referring to the per-node partitioning of X and W as in (3.7) and $\mathcal{C}(X_k)$ the set of all matrices with the same size and the same column space as X_k. To appreciate how the set (3.36) relates to the local constraint set, note that $\mathcal{S}_q(X^i) = \{X = C_q^i \widetilde{X}_q^i \mid \widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i\}$. We require from our well-posedness property in Assumption 1 and Condition 2 that $\widetilde{\mathcal{F}}_q$ should be a continuous mapping (see [125, Definition 17.2] for a formal definition). Intuitively, this means that we expect that an arbitrarily small change in the

[2] We define an accumulation point of a sequence $(X^i)_{i\in\mathbb{N}}$ as the limit of a converging subsequence $(X^i)_{i\in\mathcal{I}}$ of $(X^i)_{i\in\mathbb{N}}$, with $\mathcal{I} \subseteq \mathbb{N}$.

inputs results in the addition or removal of points arbitrarily close to other points in the output set. In Section 3.4, we will illustrate on a few selected examples how the continuity of such a mapping can be checked.

Let the point-to-set mapping $\widetilde{\mathcal{M}}_q : \mathbb{R}^{M\times Q} \rightrightarrows \mathbb{R}^{\widetilde{M}_q\times Q}$ be defined as

$$\widetilde{\mathcal{M}}_q(X) \triangleq \underset{\widetilde{W}_q\in\widetilde{\mathcal{F}}_q(X)}{\operatorname{argmin}} \|W_q - X_q\|_F^2 + \sum_{k\neq q} \|W_k - I_Q\|_F^2, \tag{3.37}$$

where $\widetilde{W}_q = [W_q^T, W_1^T, \ldots, W_{q-1}^T, W_{q+1}^T, \ldots, W_K^T]^T$, i.e., $\widetilde{\mathcal{M}}_q$ selects the point in the set $\widetilde{\mathcal{F}}_q(X)$ that is closest to $[X_q^T, I_Q, \ldots, I_Q]^T$. We then define the point-to-set mapping $\mathcal{M}_q : \mathbb{R}^{M\times Q} \rightrightarrows \mathbb{R}^{M\times Q}$ as

$$\mathcal{M}_q(X) \triangleq C_q(X)\widetilde{\mathcal{M}}_q(X). \tag{3.38}$$

A single iteration of Algorithm 3 can then be summarized as

$$X^{i+1} \in \mathcal{M}_q(X^i). \tag{3.39}$$

In very contrived cases, it could happen that $\mathcal{M}_q(X^i)$ is not a singleton, i.e., there exists more than one solution at a certain iteration i which are equidistant to the previous estimate X^i. In that case, selecting by any means one particular solution is sufficient to resolve this ambiguity.

Theorem 3.4. Suppose that for an instance $\mathbb{P}$ of (3.2) or (3.6), under the updates of Algorithm 3, Condition 2 is satisfied. Then:

1. Any accumulation point $\overline{X}$ of $(X^i)_i$ is a fixed point of the map $\mathcal{M}_q : \mathbb{R}^{M\times Q} \rightrightarrows \mathbb{R}^{M\times Q}$ for any q.
2. $\lim_{i\to+\infty} \|X^{i+1} - X^i\|_F = 0$.

Proof. See Section 3.D. □

An important corollary is that any accumulation point is a fixed point of the full DASF algorithm as it is a fixed point for an update at any node q. However, note that Theorem 3.4 still does not guarantee convergence to a single point. The latter can be established if we assume the following condition:

Condition 3a. The number of stationary points of the global problem $\mathbb{P}$ is finite.

Theorem 3.5. If Conditions 3a and 2 are satisfied, then $(X^i)_i$ converges to a single point.

Proof. See Section 3.E. □

The condition on the finiteness of the number of stationary points can be relaxed to the following condition:

Condition 3b. The number of solutions of each local problem (3.22) is finite or the solver of the local problems (3.22) can only obtain a finite subset of the solutions of (3.22).

In other words, we only require the finiteness of the number of solutions *obtainable* through the solver used for solving the local problems. For example, when maximizing $\operatorname{tr}(X^T AX)$ over $X^T BX = I$, there are infinitely many options for a solution X^* represented as $X^* = V^*U$ where U is an orthogonal matrix and V^* contains the principal generalized eigenvectors of the matrix pair (A, B). However, a solver using a generalized eigenvalue decomposition to solve this problem can only output V^* itself up to a sign change of its columns, hence the solver is only able to select solutions from a finite subset of the complete solution set.

Theorem 3.6 (Proof omitted)**.** If Conditions 2 and 3b are satisfied, then $(X^i)_i$ converges to a single point.

The proof of Theorem 3.5 can be straightforwardly applied to Theorem 3.6 as well.

3.3.4 Convergence to stationary points and global minima

We can now combine all of the previous results to eventually obtain complete convergence results of the DASF algorithm to stationary points of Problem (3.6).

Theorem 3.7. Suppose that for an instance $\mathbb{P}$ of (3.2) or (3.6), under the updates of Algorithm 3, Conditions 1 to 3 (for 1 and 3 either the form a or b) are satisfied, then $(X^i)_i$ converges and $\lim_{i\to+\infty} X^i = \overline{X}$, where $\overline{X}$ is a stationary point of $\mathbb{P}$ satisfying its KKT conditions.

Proof. From Theorems 3.5 and 3.6, $(X^i)_i$ converges to a single point $\overline{X}$. Therefore, $\overline{X}$ is a fixed point of Algorithm 3 (Theorem 3.4). From Theorems 3.2 and 3.3, fixed points of the DASF algorithm are stationary points of $\mathbb{P}$ satisfying its KKT conditions, proving the theorem. □

Theorem 3.7 can lead to stronger convergence guarantees if all minima of the problem $\mathbb{P}$ are global minima (i.e., the value of the objective function is the same in all minima), which will be explained next. It is noted that many of the common spatial filtering design criteria satisfy this assumption, including PCA, canonical correlation analysis, minimum variance beamformers, generalized eigenvalue decomposition, and trace ratio optimization.

Theorem 3.8. Under the same settings of Theorem 3.7, if all minima of $\mathbb{P}$ are global minima, the only stable fixed points of Algorithm 3 are in $\mathcal{X}^*$.

Proof. For any fixed point $\overline{X} \notin \mathcal{X}^*$, there exists a descent direction in $\mathcal{S}$ (as $\overline{X}$ cannot be a minimum), and therefore there exists a perturbation ΔX, with $X + \Delta X \in \mathcal{S}$, such that $f(X + \Delta X) \leq f(X)$. Due to the monotonic decrease of $(f(X^i))_i$ (see Theorem 3.1), the sequence $(X^i)_i$ is kicked out of equilibrium and cannot return to it, hence the equilibrium is unstable. Therefore, in the absence of local minima, the only stable fixed points of Algorithm 3 must be in $\mathcal{X}^*$. □

Corollary 3.3. If all minima of f over $\mathcal{S}$ are global minima and all conditions of Theorem 3.7 are satisfied, $(X^i)_i$ converges to the global minimum of (3.2) or (3.6) with high probability.

Proof. This corollary follows immediately from Theorems 3.7 and 3.8. Indeed, from Theorem 3.7, we know that $(X^i)_i$ converges to a stationary point. Since all fixed points that are not in $\mathcal{X}^*$ are unstable (Theorem 3.8), the algorithm will eventually escape from the neighborhoods of such unstable equilibria and will end up in a point of $\mathcal{X}^*$. □

The phrasing "with high probability" here refers to the fact that it is expected that the algorithm cannot end up in an unstable equilibrium, as it would always escape from it due to numerical or estimation noise.

Corollary 3.4. If the global problem (3.2) or (3.6) is a convex problem with a strongly convex objective f and all conditions of Theorem 3.7 are satisfied, $(X^i)_i$ converges to the unique global minimum X^*.

Proof. Theorem 3.7 guarantees convergence to a stationary point of Problem (3.6). The result comes from the fact that for a convex problem with a strongly convex objective, the unique stationary point is the global minimum. □

The previous corollary can be further relaxed by removing the requirements of Conditions 1a and 1b for a problem which is unconstrained (see Corollary 3.2):

Corollary 3.5. If the global problem (3.2) or (3.6) is unconstrained with a strongly convex objective f and Condition 2 is satisfied, $(X^i)_i$ converges to the unique global minimum X^*.

3.4 Selected examples

In this section, we illustrate how the different convergence results and conditions translate to some commonly encountered spatial filtering problems, and how the problems can be manipulated in order to satisfy the conditions in case they are violated. In the following, $\mathbf{y}$ and $\mathbf{d}$ are stochastic signals, and we denote their covariance and cross-covariance matrices as $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ and $R_{\mathbf{yd}} = \mathbb{E}[\mathbf{y}(t)\mathbf{d}^T(t)]$. We also define the corresponding compressed matrices $R^i_{\tilde{\mathbf{y}}_q\tilde{\mathbf{y}}_q} = \mathbb{E}[\tilde{\mathbf{y}}^i_q(t)\tilde{\mathbf{y}}^{iT}_q(t)] = C^{iT}_q R_{\mathbf{yy}} C^i_q$ and $R^i_{\tilde{\mathbf{y}}_q\mathbf{d}} = \mathbb{E}[\tilde{\mathbf{y}}^i_q(t)\mathbf{d}^T(t)] = C^{iT}_q R_{\mathbf{yd}}$ for the local problems.

As will be shown, we typically require $R_{\mathbf{yy}}$ to be non-singular for Condition 2 to be satisfied, in particular for the local problem to be well-posed (Assumption 1), as Assumptions 3a and 2 (or 3b) are satisfied automatically if the global problem satisfies these. We will see that a question that will arise is whether the local $R^i_{\tilde{\mathbf{y}}_q\tilde{\mathbf{y}}_q}$ is non-singular. If this is not satisfied, we cannot ignore situations where a small change in the local problem parameters leads to discontinuous changes in their solution sets, making Condition 2 (in particular Assumption 1)

invalid. The following result relates the (non-)singularity of $C_q^{iT} R C_q^i$ to the (non-)singularity of R.

Lemma 3.1. Suppose that a matrix $R \in \mathbb{R}^{M \times M}$ is non-singular. Then, given $C_q^i \in \mathbb{R}^{M \times \widetilde{M}_q}$ as defined in (3.19), if C_q^i has full rank, the matrix $C_q^{iT} R C_q^i$ is also non-singular.

The proof is omitted as this is a well-known property of the rank of a matrix. Lemma 3.1 will be used in many of the examples that are discussed in this section.

Remark 3.1. Note that — by construction — the matrix C_q^i has full rank unless one of the matrices X_k has linearly dependent columns. The latter is a contrived case and is to be avoided anyway, since it would imply that redundant data is transmitted via $\widehat{\mathbf{y}}_k$ (in which case the algorithm should only transmit the non-redundant part). The result of Lemma 3.1 can therefore be interpreted in such a way that if the global problem is well-posed, then so is the local one because from this result, we have a guarantee that the local problems' covariance matrix will also be non-singular.

3.4.1 Least squares / minimum mean square error and ridge regression

The least squares (LS) / minimum mean square error (MMSE) problem can be written as

$$\underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \; \mathbb{E}[\|X^T \mathbf{y}(t) - \mathbf{d}(t)\|^2], \tag{3.40}$$

which is an unconstrained problem with a convex quadratic objective, since $R_{\mathbf{yy}}$ is positive semi-definite by definition. A solution X^* of (3.40) needs to satisfy the normal equations given as $R_{\mathbf{yy}} X^* = R_{\mathbf{yd}}$ (see Appendix B.1). If additionally $R_{\mathbf{yy}}$ is positive definite, then it is invertible, and the objective is strongly convex leading to the unique solution $X^* = R_{\mathbf{yy}}^{-1} R_{\mathbf{yd}}$ of the global problem. In this case, (3.40) is well-posed and satisfies **Assumptions 1 and 3**, while **Assumption 2** is automatically satisfied as there are no constraints. We write the corresponding local problem (3.17) as

$$\underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \; \mathbb{E}[\|\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t) - \mathbf{d}(t)\|^2]. \tag{3.41}$$

Since (3.41) does not have any constraints, **Condition 1a** is trivially satisfied (see also Corollary 3.2). Similarly to the centralized case, $\widetilde{X}_q$ is a solution of (3.41) if and only if it satisfies the normal equations

$$R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q = R^i_{\widetilde{\mathbf{y}}_q\mathbf{d}}. \tag{3.42}$$

In general cases, $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} = C_q^{iT} R_{\mathbf{yy}} C_q^i$ is invertible from Lemma 3.1, and the solution of the local problem at iteration i and node q is unique (satisfying **Condition 3a**) and equal to

$$\widetilde{X}_q^* = (R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})^{-1} R^i_{\widetilde{\mathbf{y}}_q\mathbf{d}}. \tag{3.43}$$

We then have $\widetilde{\mathcal{F}}_q(X^i) = \{\widetilde{X}_q^*\}$, which is continuous since matrix inversion is continuous (see Cramer's rule). Hence, **Condition 2** is satisfied if $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ is non-singular. Since (3.40) is unconstrained, we satisfy the conditions of **Corollary 3.5**, and we conclude that the DASF algorithm applied to the LS / MMSE problem converges to the optimal solution X^*.

In cases where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ is singular, the normal equations have infinitely many solutions and (3.41) therefore admits infinitely many solutions as well. However, a small change in inputs can lead to $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ suddenly becoming non-singular, reducing the number of solutions from infinitely many to a single one, which would not satisfy **Condition 2**. We can resolve this problem by additionally requiring the column space of $\widetilde{X}_q$ to be orthogonal to the null space of $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$, which corresponds to the solution with minimum norm [126, Chapter 3, Section 2]. The solution of this surrogate problem is always uniquely defined and varies continuously with $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ and $R^i_{\widetilde{\mathbf{y}}_q\mathbf{d}}$, even at points where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ becomes singular. In practice though, $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ will never be singular (see also Remark 3.1), and the additional constraint is always trivially satisfied, making the surrogate problem equivalent to the original one.

If $R_{\mathbf{yy}}$ is singular (therefore implying that **Assumptions 1 and 3** do not hold in the general case for (3.40)), one can consider adding an ℓ_2-norm constraint or penalty to (3.40), leading to the ridge regression problem. As this can be rewritten as a least squares problem where $R_{\mathbf{yy}}$ is replaced by $R_{\mathbf{yy}} + \alpha I_M$, the resulting matrix becomes non-singular. In this case, both the local and global problems will satisfy the well-posedness assumption, and therefore **Corollary 3.5** straightforwardly applies, such that convergence to the global solution is guaranteed.

3.4.2 Linearly constrained minimum variance

The linearly constrained minimum variance (LCMV) problem is a convex problem often used in beamforming applications [9], and written as

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \mathbb{E}[\|X^T\mathbf{y}(t)\|^2] = \text{tr}(X^T R_{\mathbf{yy}} X) \\ \text{subject to} \quad & X^T B = A, \end{aligned} \tag{3.44}$$

with the linear term $B \in \mathbb{R}^{M\times L}$, $M > L$. If $R_{\mathbf{yy}}$ is positive definite and $\text{rank}(B) = L$, Problem (3.44) has a strongly convex objective and the unique solution is given by $X^* = R_{\mathbf{yy}}^{-1}B(B^T R_{\mathbf{yy}}^{-1}B)^{-1}A^T$ (see Appendix B.2). Problem (3.44) is then well-posed and satisfies **Assumption 1**. As shown in Example 3.2, the gradient of each constraint function h_{ml} of (3.44) is given by $\nabla_X h_{ml}(X) = \mathbf{b}(l)\mathbf{e}_m$, where $\mathbf{b}(l)$ corresponds to the $l-$th column of B. Therefore, **Assumption 2** is satisfied when B is full (column) rank. Finally, since the objective is continuous and the objective strongly convex, the sublevel sets of the objective are compact. Adding the constraints $X^T B = A$ preserves compactness as the intersection of a closed set with a compact one is compact, satisfying **Assumption 3**. The local LCMV problem is written as

$$\begin{aligned} \underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \quad & \text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q) \\ \text{subject to} \quad & \widetilde{X}_q^T\widetilde{B}_q^i = A, \end{aligned} \tag{3.45}$$

at iteration i and node q. As discussed in Example 3.2, **Condition 1a** is satisfied if B has rank L, or alternatively **Condition 1b** is satisfied if the matrix given in (3.34) has rank L. Excluding the rare cases where C_q^i is rank deficient (which can be dealt with, see Remark 3.1), $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ is invertible from Lemma 3.1, and it can be shown from the property that B is full rank that $\widetilde{B}_q^{iT}(R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})^{-1}\widetilde{B}_q^i$ is also invertible. Therefore, the unique solution of the local problem (implying **Condition 3a** is satisfied) is given by an analogous expression to the one of the global LCMV problem:

$$\widetilde{X}_q^* = (R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})^{-1}\widetilde{B}_q^i[\widetilde{B}_q^{iT}(R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})^{-1}\widetilde{B}_q^i]^{-1}A^T. \tag{3.46}$$

From the continuity of matrix inversion, **Condition 2** is satisfied. This implies that we can apply **Corollary 3.4** to conclude that the DASF algorithm will converge to the globally optimal LCMV solution. We note that **Condition 1b** is satisfied in practice with high probability when J satisfies the upper bound (3.33) as illustrated in Figure 3.1. However, there still exist situations where

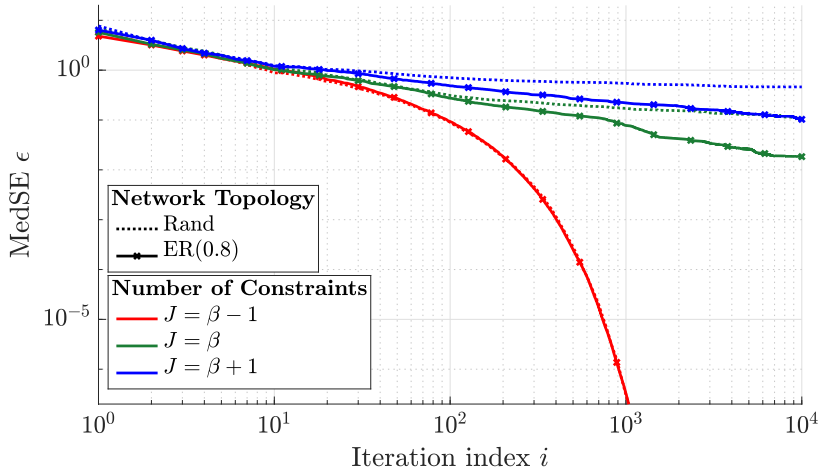


Figure 3.1: Convergence comparison of the DASF algorithm solving the LCMV problem (3.44) on two network topologies (see Table 2.3 for a description). Each point has been obtained by computing the MedSE across 100 Monte Carlo runs. β represents the upper bound given in (3.33) on the number of constraints J.

slow convergence is observed in cases where **Condition 1b** is "close" to being violated. We propose a fix for those situations in Section 3.5.1. Note that Condition 1b is sufficient for showing convergence to the optimal point but it is not a necessary condition, as in Figure 3.1, we can still see (slow) convergence for some cases when J does not satisfy (3.33).

Before continuing with the following examples, we define two expressions that will be used throughout this text.

Definition 1. Let $A, B \in \mathbb{R}^{M\times M}$ be symmetric matrices, with B positive semidefinite. Then we define

$$V = \mathrm{EVC}_Q(A) \tag{3.47}$$

as an $M \times Q$ matrix containing in its columns Q eigenvectors (EVCs) of A corresponding to its Q largest eigenvalues (EVLs), normalized such that $V^TV = I_Q$.

Similarly, the matrix

$$W = \mathrm{GEVC}_Q(A, B) \tag{3.48}$$

is defined as an $M \times Q$ matrix containing in its columns Q generalized eigenvectors (GEVCs) of the pair (A, B) corresponding to its Q largest generalized eigenvalues (GEVLs), normalized such that $W^TBW = I_Q$.

3.4.3 Generalized eigenvalue decomposition and principal component analysis

Let us consider the problem:

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & -\mathbb{E}\left[\|X^T\mathbf{y}(t)\|^2\right] = -\mathrm{tr}(X^TR_{\mathbf{yy}}X) \\ \text{subject to} \quad & \mathbb{E}[X^T\mathbf{v}(t)\mathbf{v}^T(t)X] = X^TR_{\mathbf{vv}}X = I_Q, \end{aligned} \tag{3.49}$$

where $\mathbf{y}$ and $\mathbf{v}$ are $M-$dimensional time signals. A global solution of this problem is obtained by computing the Q principal generalized eigenvectors when computing the generalized eigenvalue decomposition (GEVD) of the matrix pair $(R_{\mathbf{yy}}, R_{\mathbf{vv}})$ (see Appendix B.3). This GEVD problem is often encountered in discriminant analysis or max-SNR filtering [9, 19, 127]. It also contains the standard eigenvalue decomposition (EVD) problem or principal component analysis (PCA) as a special case, which is obtained when the constraint set of (3.49) is replaced with $X^TX = I_Q$, i.e., the Stiefel manifold (or equivalently, when $\mathbf{v}$ is a white noise process). Therefore, the discussions below apply to the EVD and PCA problems as well.

If both $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ are positive definite and the $Q + 1$ largest GEVLs of $(R_{\mathbf{yy}}, R_{\mathbf{vv}})$ are all distinct, Problem (3.49) is well-posed [128], and therefore satisfies **Assumption 1**. A solution of (3.49) is then given by $X^* = \mathrm{GEVC}_Q(R_{\mathbf{yy}}, R_{\mathbf{vv}})$, where $\mathrm{GEVC}_Q(A, B)$ is defined as in Definition 1. We note that the solution of this problem is not unique, and applying any orthogonal transformation on $X^* = \mathrm{GEVC}_Q(R_{\mathbf{yy}}, R_{\mathbf{vv}})$ is also a valid solution. Similarly

to Example 3.1, we have $\nabla_X h_{ml}(X^*) = R_{\mathbf{vv}}(\mathbf{x}^*(m)\mathbf{e}_l^T + \mathbf{x}^*(l)\mathbf{e}_m^T)$. It can be shown that the linear independence of the set $\{\nabla_X h_{ml}(X^*)\}_{m,l}$ is equivalent to the linear independence of the columns of $R_{\mathbf{vv}}X^*$. Under Assumption 1, $R_{\mathbf{vv}}$ is positive definite, hence invertible, and since X^* contains GEVCs of $(R_{\mathbf{yy}}, R_{\mathbf{vv}})$ in its columns, it is by definition full column rank. Therefore, the solutions of (3.49) satisfy the LICQ conditions hence **Assumption 2** is satisfied. Additionally, the sublevel sets of the objective of (3.49) are closed, while the constraint of (3.49) defines a compact set. From the compactness of their intersection, we satisfy **Assumption 3**. From (3.49), we observe that the corresponding local problem (3.17) is given by

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & -\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} \widetilde{X}_q) \\ \text{subject to} \quad & \widetilde{X}_q^T R^i_{\widetilde{\mathbf{v}}_q \widetilde{\mathbf{v}}_q} \widetilde{X}_q = I_Q, \end{aligned} \tag{3.50}$$

and for any fixed iteration i and node q, a solution of the local problem is

$$\widetilde{X}_q^* = \text{GEVC}_Q(R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q},\ R^i_{\widetilde{\mathbf{v}}_q \widetilde{\mathbf{v}}_q}). \tag{3.51}$$

Similarly to Example 3.1, we can show that $\nabla_X h_{ml}(X) = R_{\mathbf{vv}}(\mathbf{x}(m)\mathbf{e}_l^T + \mathbf{x}(l)\mathbf{e}_m^T)$. Therefore, for any X satisfying the constraints of (3.49), we have $X^T \nabla_X h_{ml}(X) = \mathbf{e}_m \mathbf{e}_l^T + \mathbf{e}_l \mathbf{e}_m^T$, which, for every (m, l), form a linearly independent set hence **Condition 1a** is satisfied at any iteration.

On the other hand, we do not have a guarantee that the algorithm does not converge to a local problem where the $Q+1$ largest GEVLs of the local matrix pair are all distinct. This would lead to a violation of the continuity of the problem as required in **Condition 2**, which is otherwise satisfied. This event, however improbable, can be monitored and a particular fix is described in Section 3.5.2.

As noted previously, there exists infinitely many solutions of Problem (3.49) therefore **Condition 3a** cannot be satisfied. However, suppose the solver we choose to solve the local problems (3.50) computes the GEVD of the matrix pair $(R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q},\ R^i_{\widetilde{\mathbf{v}}_q \widetilde{\mathbf{v}}_q})$ as in (3.51). Then, the solver can only output one of the 2^Q possible solutions of the local problems at each iteration, namely one of the matrices containing the Q most significant GEVCs of the matrix pair $(R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q},\ R^i_{\widetilde{\mathbf{v}}_q \widetilde{\mathbf{v}}_q})$, which are equal up to a sign change of the columns. This allows to eliminate all other solutions of (3.50) from the set of candidate solutions, making the solution set obtainable by the solver finite and leading to **Condition 3b** being satisfied.

Under these conditions, we conclude from **Theorem 3.7** that the DASF algorithm converges to a stationary point of the GEVD problem. As all

minima/maxima of (3.49) are global minima/maxima, we obtain convergence to the global solution according to **Corollary 3.3**.

3.4.4 Trace ratio optimization

The trace ratio optimization (TRO) problem [94, 129] is defined as

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & -\frac{\mathbb{E}\left[\|X^T\mathbf{v}(t)\|^2\right]}{\mathbb{E}\left[\|X^T\mathbf{y}(t)\|^2\right]} = -\frac{\operatorname{tr}(X^T R_{\mathbf{vv}} X)}{\operatorname{tr}(X^T R_{\mathbf{yy}} X)} \\ \text{subject to} \quad & X^T X = I_Q. \end{aligned} \tag{3.52}$$

Considering that $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ are positive definite, there exists a scalar ρ such that a solution of this problem is given by $X^* = \mathrm{EVC}_Q(R_{\mathbf{vv}} - \rho R_{\mathbf{yy}})$, where $\mathrm{EVC}_Q(A)$ is defined in Definition 1 (see [94] and Appendix B.4). This solution returned by the TRO solver defined in [94] is unique up to a sign change of its columns if the $Q+1$ EVLs of $R_{\mathbf{vv}} - \rho R_{\mathbf{yy}}$ are distinct, in which case the problem is well-posed, satisfying **Assumption 1**. It can be shown in a similar fashion as to 3.4.3 that (3.52) satisfies **Assumptions 2 and 3**. Additionally, From Example 3.1, we know that the Stiefel manifold satisfies **Condition 1a**. The local problem that node q solves at iteration i is written as

$$\begin{aligned} \underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \quad & -\frac{\operatorname{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}\widetilde{X}_q)}{\operatorname{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q)} \\ \text{subject to} \quad & \widetilde{X}_q^T\widetilde{\Gamma}_q^i\widetilde{X}_q = I_Q, \end{aligned} \tag{3.53}$$

with $\widetilde{\Gamma}_q^i = C_q^{iT} C_q^i$ (see Remark 2.5). There exists a scalar ρ_q^i such that the solution of the local problem is given by

$$\widetilde{X}_q^* = \mathrm{GEVC}_Q\left(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho_q^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q},\, \widetilde{\Gamma}_q^i\right), \tag{3.54}$$

if both matrices of the pair are positive definite (this holds if C_q^i is full rank from Lemma 3.1) and the $Q+1$ largest GEVLs are distinct (in Section 3.5.2, we will cover the case where this is not satisfied). Therefore, the local problems are generally well-posed, satisfying **Condition 2**. Similarly to the GEVD example, a solver using (3.54) to solve (3.53) satisfies **Condition 3b**. As there are no local minima, in practical cases we obtain convergence to a global minimum from **Corollary 3.3**.

3.4.5 Canonical correlation analysis

The goal of canonical correlation analysis (CCA) is to find two spatial filters for two different multi-channel signals such that their outputs are maximally correlated [22, 23]. The CCA problem can be (re)written as [23]

$$\begin{aligned} \underset{X,W\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & -\mathbb{E}\Big[\text{tr}\left(X^T\mathbf{y}(t)\mathbf{v}^T(t)W\right)\Big] = -\text{tr}(X^T R_{\mathbf{yv}} W) \\ \text{subject to} \quad & \mathbb{E}[X^T\mathbf{y}(t)\mathbf{y}^T(t)X] = X^T R_{\mathbf{yy}} X = I_Q, \\ & \mathbb{E}[W^T\mathbf{v}(t)\mathbf{v}^T(t)W] = W^T R_{\mathbf{vv}} W = I_Q. \end{aligned} \tag{3.55}$$

Assuming that $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ are positive definite the solution of Problem (3.55) is given by $X^* = \text{GEVC}_Q(R_{\mathbf{yv}}R_{\mathbf{vv}}^{-1}R_{\mathbf{vy}},\ R_{\mathbf{yy}})$ and $W^* = R_{\mathbf{vv}}^{-1}R_{\mathbf{vy}}X^*\Lambda^{-1/2}$, where Λ is a $Q\times Q$ diagonal matrix containing the Q largest GEVLs of the pair $(R_{\mathbf{yv}}R_{\mathbf{vv}}^{-1}R_{\mathbf{vy}},\ R_{\mathbf{yy}})$ (see Appendix B.5). If the $Q+1$ largest GEVLs of this pair are all distinct, the CCA problem is well-posed, satisfying **Assumption 1**. Similarly to 3.4.3, it can be shown that (3.55) satisfies **Assumptions 2 and 3**. As shown previously, **Condition 1a** is satisfied at any iteration i. We can write the local problems as

$$\begin{aligned} \underset{\widetilde{X}_q,\widetilde{W}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \quad & -\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{v}}_q}\widetilde{W}_q) \\ \text{subject to} \quad & \widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q = I_Q, \\ & \widetilde{W}_q^T R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}\widetilde{W}_q = I_Q. \end{aligned} \tag{3.56}$$

From Lemma 3.1 and Remark 3.1, we conclude that both $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ and $R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}$ are non-singular. The solution of the local problems is then

$$\widetilde{X}_q^* = \text{GEVC}_Q\left(R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{v}}_q}(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q})^{-1}R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{y}}_q},\ R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\right) \tag{3.57}$$

$$\widetilde{W}_q^* = (R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q})^{-1}R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q^*(\Lambda_q^i)^{-1/2}, \tag{3.58}$$

where Λ_q^i is the $Q\times Q$ diagonal matrix containing the Q largest GEVLs of the pair given in (3.57). If the $Q+1$ GEVLs are all distinct, the local problem (3.56) is again well-posed and **Condition 2** is satisfied. If this is not the case, the same fixes we discussed in the examples of the GEVD and TRO problems can be applied. As in these latter examples, **Condition 3b** is also satisfied when choosing a solver which computes the solutions of (3.56) using (3.57)-(3.58). Therefore, the corresponding DASF algorithm will converge to the centralized CCA solution from **Corollary 3.3**.

3.5 Fixes in case of violations of conditions

3.5.1 Avoiding violations of Condition 1b

In certain cases, the linear independence requirements of Condition 1a or 1b cannot be a priori guaranteed. In such cases, even though they are expected to hold (since they are only violated in a discrete set of points within a continuum of possibilities), it is possible that the DASF algorithm passes a neighborhood of a fixed point $\overline{X}$ where the conditions are not satisfied, especially when J is close to the upper bound (3.33) or even larger (see Figure 3.1). When an updating node q observes that the elements of $\{D_{j,q}(X^i)\}_j$ are close to being linearly dependent, the algorithm is potentially converging to a fixed point that does not satisfy the linear independence requirements. This can be checked at the node itself, e.g., when the $J-$th singular value of $[\text{vec}(D_{1,q}(X^i)), \ldots, \text{vec}(D_{J,q}(X^i))]$ is close to zero[3]. In that case, we propose that the neighbors k of node q split their local X_k^i into a sum of two matrices $X_{k,a}^i$ and $X_{k,b}^i$ (which both have linearly independent columns) such that $X_k^i = X_{k,a}^i + X_{k,b}^i$, and start to temporarily communicate two sets of compressed signals $\widehat{\mathbf{y}}_{k,a}^i = X_{k,a}^{iT}\mathbf{y}_k$ and $\widehat{\mathbf{y}}_{k,b}^i = X_{k,b}^{iT}\mathbf{y}_k$. This implies that the elements of $\{D_{j,q}(X^i)\}_j$ have grown in dimension thereby making the linear independence requirement of the set $\{D_{j,q}(X^i)\}_j$ more likely to be met. As soon as the algorithm escapes the suboptimal point, node k can again merge $X_{k,a}$ and $X_{k,b}$ for future iterations, returning to the minimal communication bandwidth setting.

3.5.2 Avoiding violations of Condition 2

In the very contrived cases where the algorithm would produce a subsequence converging to a stationary point at which Condition 2 does not hold, convergence of the overall sequence cannot be guaranteed anymore and the algorithm will possibly oscillate between points which are solutions of the local problems, but not necessarily corresponding to global solutions. Indeed, without Condition 2, we cannot guarantee that $\lim_{X \to \overline{X}} \mathcal{M}_q(X)$ is well-defined (i.e., is unique and a singleton). A practical and pragmatic fix for such scenarios is to monitor both the potential oscillatory behavior and the continuity of $\widetilde{\mathcal{F}}_q(X^i)$ and skip the update at the node where a problem occurs. To detect such an oscillation,

[3]Note that, depending on the problem, constructing $D_{j,q}$'s at node q might require some additional data exchange, namely each node k should compute and transmit $X_k^{iT}\nabla_{X_k} h_j(X^i)$'s towards node q. However, this would not add a significant burden to the communication cost as these matrices are $Q \times Q$, while there are also many cases where the compressed gradients are already required to be communicated due to the problem's structure.

select an arbitrarily small $\varepsilon > 0$, and monitor whether $|f(X^{i+1}) - f(X^i)| \cdot \|X^{i+1} - X^i\|_F^{-1} > \varepsilon$ which can be interpreted as a sufficient decrease condition. If this condition is violated, a potential oscillation is flagged. In addition, one could monitor whether a particular near-discontinuity condition is met, which is problem specific. This condition can be interpreted as a sufficient decrease condition. In the case of GEVD and TRO problems discussed above, monitoring the difference between the $Q-$th and $(Q+1)-$th largest eigenvalue is sufficient to detect such a discontinuity. If both such a near-discontinuity and an insufficient decrease are flagged, the update at that node should be skipped. We refer the reader to Section 6.C for a detailed discussion on this issue for the specific case of the TRO problem.

3.6 Conclusion

In this chapter, we have analyzed the technical convergence properties and conditions of the DASF algorithm and provided formal proofs of convergence. The conditions required for convergence were shown to be satisfied in many practical problems, assuming some bounds on the dimension of the problem are satisfied, which depend on the number of constraints and the network topology. We have provided some illustrative examples to demonstrate how the — sometimes rather technical — conditions can be validated in practice. These examples also showed in which contrived cases a problem could occur, e.g., in case of singularities or eigenvalue collisions in sensor signal covariance matrices, which are expected to be rare in practice. Nevertheless, we have discussed various methods to fix these convergence problems for the rare cases where these should occur.

The technical convergence analysis of the DASF algorithm presented in this chapter concludes the first part of this thesis. The second part is dedicated to various extensions and improvements of the DASF framework in the aim of further generalizing it to new problems and making it more efficient in multiple aspects.

Appendices

3.A Fixed points are KKT points under Condition 1a

This section provides a proof for Theorem 3.2.

Let us write the KKT conditions of Problem (3.6), as mentioned in Assumption 2 [130]:

$$\nabla_X \mathcal{L}(X, \boldsymbol{\lambda}) = 0, \tag{3.59}$$

$$h_j(X) \leq 0 \ \ \forall j \in \mathcal{J}_I,\ h_j(X) = 0 \ \ \forall j \in \mathcal{J}_E, \tag{3.60}$$

$$\lambda_j \geq 0 \ \ \forall j \in \mathcal{J}_I, \tag{3.61}$$

$$\lambda_j h_j(X) = 0 \ \ \forall j \in \mathcal{J}_I, \tag{3.62}$$

where

$$\mathcal{L}(X, \boldsymbol{\lambda}) \triangleq f(X) + \sum_{j \in \mathcal{J}} \lambda_j h_j(X) \tag{3.63}$$

is the Lagrangian with $\lambda_j \in \mathbb{R}$ the Lagrange multiplier corresponding to the constraint h_j and $\boldsymbol{\lambda}$ in bold is used as a shorthand notation for the set of all Lagrange multipliers. These KKT conditions can also be formalized for the local optimization problem (3.22) defined at the updating node q. At iteration i, the updating node q solves its local problem (3.22) which has the following Lagrangian

$$\widetilde{\mathcal{L}}(\widetilde{X}_q, \widetilde{\boldsymbol{\lambda}}(q)) \triangleq f(C_q^i \widetilde{X}_q) + \sum_{j \in \mathcal{J}} \lambda_j(q) h_j(C_q^i \widetilde{X}_q), \tag{3.64}$$

where $\widetilde{\boldsymbol{\lambda}}(q)$ represents the set of Lagrange multipliers $\lambda_j(q)$ corresponding to the local problem (3.22) at node q and iteration i. Since $\widetilde{X}_q^*$ solves the local problem (3.22), it must satisfy the KKT conditions of the local problem. In particular, we can write the stationarity condition as

$$\nabla_{\widetilde{X}_q} \widetilde{\mathcal{L}}(\widetilde{X}_q^*, \widetilde{\boldsymbol{\lambda}}(q)) = 0. \tag{3.65}$$

From the parameterization $X = C_q^i \widetilde{X}_q$, we obtain

$$\widetilde{\mathcal{L}}(\widetilde{X}_q, \widetilde{\boldsymbol{\lambda}}(q)) = \mathcal{L}(C_q^i \widetilde{X}_q, \widetilde{\boldsymbol{\lambda}}(q)). \tag{3.66}$$

By applying the chain rule on (3.65), we have (see Example A.4)

$$C_q^{iT} \nabla_X \mathcal{L}(C_q^i \widetilde{X}_q^*, \widetilde{\boldsymbol{\lambda}}(q)) = 0. \tag{3.67}$$

The KKT conditions for optimality of the local problem (3.22) can then be written as

$$C_q^{iT}\nabla_X\left[f\left(C_q^i\widetilde{X}_q^*\right)+\sum_{j\in\mathcal{J}}\lambda_j(q)h_j\left(C_q^i\widetilde{X}_q^*\right)\right]=0, \tag{3.68}$$

$$h_j\left(C_q^i\widetilde{X}_q^*\right)\leq 0\ \ \forall j\in\mathcal{J}_I,\ h_j\left(C_q^i\widetilde{X}_q^*\right)=0\ \ \forall j\in\mathcal{J}_E, \tag{3.69}$$

$$\lambda_j(q)\geq 0\ \ \forall j\in\mathcal{J}_I, \tag{3.70}$$

$$\lambda_j(q)h_j\left(C_q^i\widetilde{X}_q^*\right)=0\ \ \forall j\in\mathcal{J}_I, \tag{3.71}$$

where the $\lambda_j(q)$'s are the Lagrange multipliers at updating node q and iteration i. Since (3.69) is exactly the same as (3.60), we conclude that the local primal feasibility condition is also satisfied globally (which we already knew from (3.23) and Lemma 2.1). Let us now look at the three other equations and assume that the algorithm has reached a fixed point, i.e., $X^{i+1}=X^i=\overline{X}$. From this fixed point assumption, we can replace $C_q^i\widetilde{X}_q^*=X^{i+1}$ with $C_q^i\widetilde{X}_q^i=X^i=\overline{X}$ in (3.68) such that the local stationarity condition can be rewritten as

$$C_q^{iT}\nabla_X\left[f(\overline{X})+\sum_{j\in\mathcal{J}}\lambda_j(q)h_j(\overline{X})\right]=0. \tag{3.72}$$

Selecting the first M_q rows of (3.72), we have (note that from (3.19), the matrix C_q^{iT} selects the $q-$th block-row from X as the first M_q rows)

$$\nabla_{X_q}f(\overline{X})=-\sum_{j\in\mathcal{J}}\lambda_j(q)\nabla_{X_q}h_j(\overline{X}). \tag{3.73}$$

Since this result is valid for any node q due to the fixed point assumption, we may stack the variations of equation (3.73) for every node $q\in\mathcal{K}$:

$$\begin{bmatrix}\nabla_{X_1}f(\overline{X})\\ \vdots\\ \nabla_{X_K}f(\overline{X})\end{bmatrix}=\nabla_X f(\overline{X})=-\begin{bmatrix}\sum_{j\in\mathcal{J}}\lambda_j(1)\nabla_{X_1}h_j(\overline{X})\\ \vdots\\ \sum_{j\in\mathcal{J}}\lambda_j(K)\nabla_{X_K}h_j(\overline{X})\end{bmatrix}. \tag{3.74}$$

Multiplying (3.72) from the left by $\widetilde{X}_q^{iT}$ defined in (3.25) and using the fact that $C_q^i\widetilde{X}_q^i=X^i=\overline{X}$ (this follows from (3.25) and the definition of C_q^i in (3.19)-(3.20)), we obtain

$$\overline{X}^T\nabla_X f(\overline{X})=-\sum_{j\in\mathcal{J}}\lambda_j(q)\overline{X}^T\nabla_X h_j(\overline{X}). \tag{3.75}$$

From Condition 1a, the set $\{\overline{X}^T \nabla_X h_j(\overline{X})\}_j$ is linearly independent and therefore the Lagrange multipliers $\{\lambda_j(q)\}_j$ that satisfy (3.75) are unique. Moreover, since the left-hand side of (3.75) does not depend on the node q, we have that $\lambda_j(q) = \lambda_j$ for any node q. Therefore, (3.74) becomes

$$\nabla_X f(\overline{X}) = -\sum_{j\in\mathcal{J}} \lambda_j \nabla_X h_j(\overline{X}), \tag{3.76}$$

which implies that $(\overline{X}, \{\lambda_j\}_j)$ satisfy the global stationarity conditions (3.59). Since $(\overline{X}, \{\lambda_j\}_j)$ satisfies the local dual feasibility and the local complementary slackness conditions (3.70) and (3.71) (with $C_q^i \widetilde{X}_q^* = X^{i+1}$ replaced by $\overline{X}$ due to it being a fixed point), it also satisfies their global counterparts (3.61) and (3.62). Hence, $(\overline{X}, \{\lambda_j\}_j)$ satisfies all the global KKT optimality conditions. This proves that any fixed point of Algorithm 3 is a stationary point of the global problem (3.6). □

3.B Fixed points are KKT points under Condition 1b

This section provides a proof for Theorem 3.3.

The arguments in this proof are very similar to the ones in the proof of Theorem 3.2, where the main difference is to show the uniqueness of the Lagrange multipliers when $\{D_{j,q}(\overline{X})\}_j$ is a linearly independent set for any $q \in \mathcal{K}$ at fixed points $\overline{X} = X^{i+1} = X^i$. Therefore, we only make changes to that part of the proof.

From the definition of C_q^i in (3.19)-(3.20), left-multiplying each block-row of (3.72) by the corresponding block-column of $\widetilde{X}_q^{*T}$ as structured in (3.13), we have

$$X_q^{(i+1)T} \nabla_{X_q} \Big[f(X^i) + \sum_{j\in\mathcal{J}} \lambda_j(q) h_j(X^i) \Big] = 0, \tag{3.77}$$

$$G_n^{(i+1)T} \sum_{k\in\mathcal{B}_{nq}} X_k^{iT} \nabla_{X_k} \Big[f(X^i) + \sum_{j\in\mathcal{J}} \lambda_j(q) h_j(X^i) \Big] = 0, \tag{3.78}$$

$\forall n \in \mathcal{N}_q$. Note that we here again assume that the algorithm has reached a fixed point for which $X^{i+1} = X^i = \overline{X}$ (as this was also assumed to derive (3.72)), which implies that we can make the substitutions $X_q^{i+1} = X_q^i = \overline{X}_q$ and $G_n^{i+1} = I_Q$ for all $n \in \mathcal{N}_q$ (see (3.26)) within (3.77)-(3.78). By doing so and,

from the definition (3.30) of $D_{j,q}(\overline{X})$, (3.77) and (3.78) become

$$C_{X_q}^{iT}\nabla_X f(\overline{X}) + \sum_{j\in\mathcal{J}} \lambda_j(q) D_{j,q}(\overline{X}) = 0, \tag{3.79}$$

where $C_{X_q}^i$ is the matrix C_q^i but the identity matrix of the first block-column in (3.19) has been replaced by $X_q^i = \overline{X}_q$. From (3.79) and the linear independence assumption over the set $\{D_{j,q}(\overline{X})\}_j$ (see Condition 1b), the Lagrange multipliers $\lambda_j(q)$ for all $j \in \mathcal{J}$ that satisfy (3.79) must be unique. We can repeat the same argument for any updating node, which implies that (3.77)-(3.79) holds for any node q, each time with its own unique set of Lagrange multipliers. We will now prove that this unique set of Lagrange multipliers is the same for any updating node q.

Slightly rewriting (3.77)-(3.78) (with $X^{i+1} = X^i = \overline{X}$ and $G_n^{i+1} = I_Q$) gives

$$\overline{X}_q^T \nabla_{X_q} f(\overline{X}) = -\sum_{j\in\mathcal{J}} \lambda_j(q) \overline{X}_q^T \nabla_{X_q} h_j(\overline{X}), \tag{3.80}$$

$$\sum_{k\in\mathcal{B}_{nq}} \overline{X}_k^T \nabla_{X_k} f(\overline{X}) = -\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{B}_{nq}} \lambda_j(q) \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}), \tag{3.81}$$

where (3.80)-(3.81) holds for every $q \in \mathcal{K}$ and for all $n \in \mathcal{N}_q$. Substituting (3.80) into (3.81), we obtain

$$\begin{aligned} &\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{B}_{nq}} \lambda_j(k) \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \\ =&\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{B}_{nq}} \lambda_j(q) \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}). \end{aligned} \tag{3.82}$$

Vectorizing the matrices in (3.82) such that $\mathbf{h}_{j,k} = \text{vec}\left(\overline{X}_k^T \nabla_{X_k} h_j(\overline{X})\right) \in \mathbb{R}^{Q^2}$ and defining $H_k = [\mathbf{h}_{1,k}, \ldots, \mathbf{h}_{J,k}] \in \mathbb{R}^{Q^2 \times J}$ $\forall k \in \mathcal{K}$, we obtain

$$\sum_{k\in\mathcal{B}_{nq}} H_k \boldsymbol{\lambda}(k) = \left(\sum_{k\in\mathcal{B}_{nq}} H_k\right) \boldsymbol{\lambda}(q), \tag{3.83}$$

where $\boldsymbol{\lambda}(k) = [\lambda_1(k), \ldots, \lambda_J(k)]^T$. At node q and for its corresponding neighbor $n \in \mathcal{N}_q$, we can then write the following linear system of equations:

$$\mathbf{H}_{nq} \cdot \boldsymbol{\lambda}_{\mathcal{K}} = 0, \tag{3.84}$$

where $\boldsymbol{\lambda}_{\mathcal{K}} = [\boldsymbol{\lambda}^T(1), \ldots, \boldsymbol{\lambda}^T(K)]^T \in \mathbb{R}^{KJ}$ and $\mathbf{H}_{nq} \in \mathbb{R}^{Q^2 \times KJ}$ is a block-column matrix where the $l-$th block of size $Q^2 \times J$ is given by

$$\mathbf{H}_{nq}(l) = \begin{cases} -\sum_{k \in \mathcal{B}_{nq}} H_k & \text{if } l = q, \\ H_l & \text{if } l \in \mathcal{B}_{nq}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.85}$$

Stacking vertically the matrices $\mathbf{H}_{nq}$, for every neighbor $n \in \mathcal{N}_q$ and every node $q \in \mathcal{K}$, results in $\mathbf{H} \in \mathbb{R}^{Q^2 \sum_k |\mathcal{N}_k| \times KJ}$ and we write

$$\mathbf{H} \cdot \boldsymbol{\lambda}_{\mathcal{K}} = 0. \tag{3.86}$$

Note that from (3.85), the sum of all $Q^2 \times J$ block-columns of $\mathbf{H}_{nq}(l)$ must be equal to the zero matrix. Therefore, every $\boldsymbol{\lambda}_{\mathcal{K}}$ such that $\boldsymbol{\lambda}(1) = \cdots = \boldsymbol{\lambda}(K)$ is in the null space of $\mathbf{H}$ and would satisfy (3.86). The dimension of the set $\{\boldsymbol{\lambda}_{\mathcal{K}} \in \mathbb{R}^{KJ} \mid \boldsymbol{\lambda}(1) = \cdots = \boldsymbol{\lambda}(K)\}$ is equal to J and therefore $\text{rank}(\mathbf{H}) \leq KJ - J$. To ensure that these are the only solutions of (3.86), we require $\text{rank}(\mathbf{H}) = KJ - J$. Note that a necessary condition to satisfy this is that $KJ - J \leq Q^2 \sum_k |\mathcal{N}_k|$, i.e., $KJ - J$ is less than the number of rows of $\mathbf{H}$, or equivalently $J \leq \frac{Q^2}{K-1} \sum_k |\mathcal{N}_k|$, leading to the upper bound given in (3.32).

Lemma 3.2. If Condition 1b holds, then $\text{rank}(\mathbf{H}) = KJ - J$.

Proof. The proof of this lemma is provided in Section 3.C as it is too elaborate and would break the flow of the text. □

Since $\text{rank}(\mathbf{H}) = KJ - J$, the set $\{\boldsymbol{\lambda}_{\mathcal{K}} \in \mathbb{R}^{KJ} \mid \boldsymbol{\lambda}(1) = \cdots = \boldsymbol{\lambda}(K)\}$ contains the full null space of $\mathbf{H}$ and hence all the solutions of (3.86). Because we now have established that all the Lagrange multipliers are the same, we can conclude that (3.74) holds in this case as well, allowing us to obtain the same result as in (3.76). The remaining arguments of the proof of Theorem 3.2 can be applied here to conclude that each fixed point is a point satisfying the KKT conditions (3.59)-(3.62) of the original problem (3.6). □

3.C $\text{rank}(\mathbf{H}) = KJ - J$

This section provides a proof for Lemma 3.2.

We want to show that $\text{rank}(\mathbf{H}) = KJ - J$ which implies that the equation $\mathbf{H} \cdot \boldsymbol{\lambda}_{\mathcal{K}} = 0$ in (3.86) can only have solutions $\boldsymbol{\lambda}_{\mathcal{K}} = [\boldsymbol{\lambda}^T(1), \ldots, \boldsymbol{\lambda}^T(K)]^T$ of the form $\boldsymbol{\lambda}(1) = \cdots = \boldsymbol{\lambda}(K)$ when Condition 1b is satisfied.

Condition 1b. *For a fixed point $\overline{X}$ of Algorithm 3, the elements of the set $\{D_{j,q}(\overline{X})\}_{j\in\mathcal{J}}$ are linearly independent for any q, where*

$$D_{j,q}(\overline{X}) = \begin{bmatrix} \overline{X}_q^T \nabla_{X_q} h_j(\overline{X}) \\ \sum_{k\in\mathcal{B}_{n_1 q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \\ \vdots \\ \sum_{k\in\mathcal{B}_{n_{|\mathcal{N}_q|} q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \end{bmatrix}, \tag{3.87}$$

which is a block-matrix containing $(1 + |\mathcal{N}_q|)$ blocks of $Q \times Q$ matrices.

The proof will be accompanied by an example network topology, given in Figure 3.A, to visualize the structure of some large matrices in the proof, yet we keep the proof itself generic.

At a fixed point $\overline{X} = X^{i+1} = X^i$ of the DASF algorithm, we have shown in Section 3.B that at each node $q \in \mathcal{K}$ the local stationarity conditions can be written as

$$\overline{X}_q^T \nabla_{X_q} f(\overline{X}) = -\sum_{j\in\mathcal{J}} \lambda_j(q) \overline{X}_q^T \nabla_{X_q} h_j(\overline{X}), \tag{3.88}$$

$$\sum_{k\in\mathcal{B}_{nq}} \overline{X}_k^T \nabla_{X_k} f(\overline{X}) = -\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{B}_{nq}} \lambda_j(q) \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}), \tag{3.89}$$

$\forall n \in \mathcal{N}_q$, leading to the equations

$$\begin{aligned} &\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{B}_{nq}} \lambda_j(k) \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \\ &= \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{B}_{nq}} \lambda_j(q) \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}), \ \forall n \in \mathcal{N}_q. \end{aligned} \tag{3.90}$$

For clarity, we vectorize the matrices $\overline{X}_k^T \nabla_{X_k} h_j(\overline{X})$ such that $\mathbf{h}_{j,k} = \text{vec}\left(\overline{X}_k^T \nabla_{X_k} h_j(\overline{X})\right) \in \mathbb{R}^{Q^2}$ and create the matrix $H_k = [\mathbf{h}_{1,k}, \ldots, \mathbf{h}_{J,k}] \in \mathbb{R}^{Q^2 \times J}$ $\forall k \in \mathcal{K}$. Note that the linear independence condition of $\{D_{j,q}\}_j$ for any

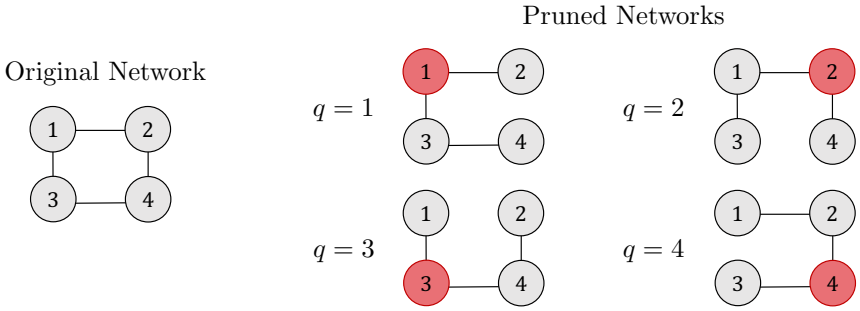


Figure 3.A: The example 4−node network we will use to illustrate the equations.

node q is then equivalent to

$$\mathbf{D_q} = \begin{bmatrix} H_q \\ \sum_{k\in\mathcal{B}_{n_1 q}} H_k \\ \vdots \\ \sum_{k\in\mathcal{B}_{n_{|\mathcal{N}_q|} q}} H_k \end{bmatrix} \tag{3.91}$$

being full rank, i.e., $\text{rank}(\mathbf{D_q}) = J$. Then, (3.90) can be rewritten as

$$\sum_{k\in\mathcal{B}_{nq}} H_k \boldsymbol{\lambda}(k) = \left(\sum_{k\in\mathcal{B}_{nq}} H_k \right) \boldsymbol{\lambda}(q), \ \forall n \in \mathcal{N}_q, \tag{3.92}$$

where $\boldsymbol{\lambda}(k) = [\lambda_1(k), \ldots, \lambda_J(k)]^T$. For example, in the example network from Figure 3.A, we have for $q = 1$:

$$H_2 \boldsymbol{\lambda}(2) = H_2 \boldsymbol{\lambda}(1), \tag{3.93}$$

$$H_3 \boldsymbol{\lambda}(3) + H_4 \boldsymbol{\lambda}(4) = (H_3 + H_4) \boldsymbol{\lambda}(1). \tag{3.94}$$

Equation (3.92) corresponds to the linear system given as

$$\mathbf{H}_{nq} \cdot \boldsymbol{\lambda}_{\mathcal{K}} = 0, \tag{3.95}$$

with $\mathbf{H}_{nq}$ a block-row matrix, where each block corresponds to a node $l \in \mathcal{K}$:

$$\mathbf{H}_{nq}(l) = \begin{cases} -\sum_{k\in\mathcal{B}_{nq}} H_k & \text{if } l = q, \\ H_l & \text{if } l \in \mathcal{B}_{nq}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.96}$$

We can then stack vertically every $\mathbf{H}_{nq}$ for every neighbor $n \in \mathcal{N}_q$ of q, and for every node q to obtain

$$\mathbf{H} \cdot \boldsymbol{\lambda}_{\mathcal{K}} = 0. \tag{3.97}$$

In the example network of Figure 3.A, we have the matrix given in (3.98), where we separated by horizontal lines the blocks corresponding to each different node for clarity.

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{2,1} \\ \mathbf{H}_{3,1} \\ \hline \mathbf{H}_{1,2} \\ \mathbf{H}_{4,2} \\ \hline \mathbf{H}_{1,3} \\ \mathbf{H}_{4,3} \\ \hline \mathbf{H}_{2,4} \\ \mathbf{H}_{3,4} \end{bmatrix} = \left[\begin{array}{cccc} -H_2 & H_2 & 0 & 0 \\ -H_3 - H_4 & 0 & H_3 & H_4 \\ \hline H_1 & -H_1 - H_3 & H_3 & 0 \\ 0 & -H_4 & 0 & H_4 \\ \hline H_1 & 0 & -H_1 & 0 \\ 0 & H_2 & -H_2 - H_4 & H_4 \\ \hline H_1 & H_2 & 0 & -H_1 - H_2 \\ 0 & 0 & H_3 & -H_3 \end{array}\right] \tag{3.98}$$

Note that $\mathbf{H}$ is a $Q^2 \sum_{k \in \mathcal{K}} |\mathcal{N}_k| \times KJ$ matrix and since $\forall q \in \mathcal{K}$, $n \in \mathcal{N}_q$,

$$\sum_{k \in \mathcal{K} \setminus \{l\}} \mathbf{H}_{nq}(k) = -\mathbf{H}_{nq}(l) \tag{3.99}$$

$\forall l \in \mathcal{K}$, all $\boldsymbol{\lambda}_{\mathcal{K}} \in \mathbb{R}^{KJ}$ such that $\boldsymbol{\lambda}(1) = \cdots = \boldsymbol{\lambda}(K)$ is a solution of (3.97). A basis for this solution space is given by

$$\mathcal{E} = \left\{ \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_1 \end{bmatrix}, \begin{bmatrix} \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_2 \end{bmatrix}, \dots \begin{bmatrix} \mathbf{e}_J \\ \vdots \\ \mathbf{e}_J \end{bmatrix} \right\} \subset \mathbb{R}^{KJ}, \tag{3.100}$$

where $\mathbf{e}_k$'s represent the standard basis for $\mathbb{R}^J$. Since $\mathcal{E}$ spans a $J-$dimensional space, we have

$$\text{rank}(\mathbf{H}) \leq KJ - J. \tag{3.101}$$

To show that the solution of (3.97) all satisfy $\boldsymbol{\lambda}(1) = \cdots = \boldsymbol{\lambda}(K)$, we need to show that all solutions are in span($\mathcal{E}$), i.e., rank($\mathbf{H}$) $= KJ - J$, which is stated in Lemma 3.2, repeated below and followed by a proof. By the dimensions of $\mathbf{H}$, this is only possible if we have $Q^2 \sum_{k \in \mathcal{K}} |\mathcal{N}_k| \geq KJ - J$ or equivalently $J \leq \frac{Q^2}{K-1} \sum_{k \in \mathcal{K}} |\mathcal{N}_k|$.

Lemma 3.2. *If Condition 1b holds, then rank*($\mathbf{H}$) $= KJ - J$.

Proof. Let us take the bottom part of the matrix $\mathbf{H}$ corresponding to the neighbors of the last node K, and take the sum over its block-rows $\mathbf{H}_{nK}$,

$n \in \mathcal{N}_K$, defined in (3.96), which results in a new summed block row

$$\mathbf{H}_{\Sigma K}(l) = \begin{cases} -\sum_{k\in\mathcal{K}\setminus\{K\}} H_k & \text{if } l = K \\ H_l & \text{if } l \neq K. \end{cases} \tag{3.102}$$

In the example network, we have $K = 4$, therefore

$$\mathbf{H}_{\Sigma 4} = \left[H_1 \;\middle|\; H_2 \;\middle|\; H_3 \;\middle|\; -H_1 - H_2 - H_3 \right]. \tag{3.103}$$

We then insert vertically the matrices $\mathbf{H}_{\Sigma K}$ after each block $\mathbf{H}_{n_{|\mathcal{N}_q|}q}$, $q \neq K$,

$$\widetilde{\mathbf{H}} = \begin{bmatrix} \mathbf{H}_{2,1} \\ \mathbf{H}_{3,1} \\ \mathbf{H}_{\Sigma 4} \\ \hline \mathbf{H}_{1,2} \\ \mathbf{H}_{4,2} \\ \mathbf{H}_{\Sigma 4} \\ \hline \mathbf{H}_{1,3} \\ \mathbf{H}_{4,3} \\ \mathbf{H}_{\Sigma 4} \end{bmatrix} = \left[\begin{array}{c|c|c|c} -H_2 & H_2 & 0 & 0 \\ -H_3 - H_4 & 0 & H_3 & H_4 \\ H_1 & H_2 & H_3 & -H_1 - H_2 - H_3 \\ \hline H_1 & -H_1 - H_3 & H_3 & 0 \\ 0 & -H_4 & 0 & H_4 \\ H_1 & H_2 & H_3 & -H_1 - H_2 - H_3 \\ \hline H_1 & 0 & -H_1 & 0 \\ 0 & H_2 & -H_2 - H_4 & H_4 \\ H_1 & H_2 & H_3 & -H_1 - H_2 - H_3 \end{array}\right] \tag{3.104}$$

of $\mathbf{H}$. Note that this insertion cannot change the rank of the matrix, as all the inserted rows are sums of existing rows in $\mathbf{H}$. We also remove the block-rows $[\mathbf{H}^T_{n_1 K}, \ldots, \mathbf{H}^T_{n_{|\mathcal{N}_K|}K}]^T$, i.e., the block-rows corresponding to node K, to obtain a new matrix $\widetilde{\mathbf{H}}$. For the example in Figure 3.A, this results in the matrix given in (3.104). Note that we have

$$\operatorname{rank}(\mathbf{H}) \geq \operatorname{rank}(\widetilde{\mathbf{H}}) \tag{3.105}$$

since the removal of rows can only reduce the rank.

We will first look at the rank of $\widetilde{\mathbf{H}}$ and derive from it the rank of $\mathbf{H}$. For this, we will apply the Gaussian elimination method using elementary row operations (EROs) which are known to not change the rank. Referring to the block decomposition of example (3.104), $\widetilde{\mathbf{H}}$ has K block-columns, and each of these block-columns of $\widetilde{\mathbf{H}}$ will be referred to as **the block-column at position** $k \in \mathcal{K}$. We refer to the matrix $[\mathbf{H}^T_{n_1 q}, \ldots, \mathbf{H}^T_{n_{|\mathcal{N}_q|}q}, \mathbf{H}_{\Sigma K}]^T$, $q \neq K$, and the resulting matrices obtained by applying EROs to it as **the submatrix corresponding to node** q. For example, in (3.104) the block-column at position 3 of the submatrix corresponding to node 2 is equal to $[H_3^T, 0, H_3^T]^T$.

For each $q \neq K$, let us sum all block-rows $\mathbf{H}_{nq}$, $n \in \mathcal{N}_q$. The result is then subtracted from the block-row $\mathbf{H}_{\Sigma K}$ leading to the final block-row of the submatrix corresponding to each node q being of the form

$[0|\dots|0|\sum_{k\in\mathcal{K}} H_k|0|\dots|0|-\sum_{k\in\mathcal{K}} H_k]$, where the first non-zero matrix is at position $q \neq K$. For our example network, we obtain

$$\left[\begin{array}{c|c|c|c} -H_2 & H_2 & 0 & 0 \\ -H_3-H_4 & 0 & H_3 & H_4 \\ \sum_{k=1}^4 H_k & 0 & 0 & -\sum_{k=1}^4 H_k \\ \hline H_1 & -H_1-H_3 & H_3 & 0 \\ 0 & -H_4 & 0 & H_4 \\ 0 & \sum_{k=1}^4 H_k & 0 & -\sum_{k=1}^4 H_k \\ \hline H_1 & 0 & -H_1 & 0 \\ 0 & H_2 & -H_2-H_4 & H_4 \\ 0 & 0 & \sum_{k=1}^4 H_k & -\sum_{k=1}^4 H_k \end{array}\right]. \tag{3.106}$$

An important observation is that, for each $q \neq K$, the block-column at position q corresponding to the submatrix of node q can be obtained by applying EROs to $\mathbf{D_q}$ defined in (3.91), i.e., it is equal to $\mathbf{D_q}$ up to EROs. Since $\text{rank}(\mathbf{D_q}) = J$ by Condition 1b, we can apply the necessary EROs to obtain the following reduced echelon form for the submatrix corresponding to node $q = 1$:

$$\left[\begin{array}{c|c|c|c|c} I_J & * & \cdots & * & * \\ 0 & & & & \end{array}\right]. \tag{3.107}$$

We can use the J pivots in the first J columns of (3.107) to create zeros at all the entries underneath (in the submatrices corresponding to $q \neq 1$) using EROs. As a result, we obtain

$$\left[\begin{array}{c|c|c|c|c} I_J & * & \cdots & * & * \\ 0 & & & & \\ \hline 0 & \mathcal{R}\mathbf{D_2} & \cdots & * & * \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline 0 & * & \cdots & \mathcal{R}\mathbf{D_{K-1}} & * \end{array}\right], \tag{3.108}$$

where $\mathcal{R}\mathbf{D_k}$ is a matrix equal to $\mathbf{D_k}$ up to EROs. To see why (3.108) holds, i.e., that each block-column at position $k \notin \{1, K\}$ of the submatrix corresponding to node k is indeed equal to $\mathbf{D_k}$ up to EROs, we note that, initially, every row above this block is either full of zeros or a row of H_k (see (3.106) for a visual example). Therefore, the EROs we applied to the full matrix to create zeros at all entries underneath the J pivots in the block-column corresponding to node $q = 1$ do not change the fact that the block-column at position k of the submatrix corresponding to node k is equal to $\mathbf{D_k}$ up to EROs. Since EROs do not change the rank of a matrix, the submatrix $\mathcal{R}\mathbf{D_2}$ should again have rank J

and so we can again create J pivots to create zeros underneath. Repeating this process for $2 \leq q \leq K-1$, we obtain

$$\left[\begin{array}{c|c|c|c|c} \begin{matrix} I_J \\ 0 \end{matrix} & * & \cdots & * & * \\ \hline 0 & \begin{matrix} I_J \\ 0 \end{matrix} & \cdots & * & * \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline 0 & 0 & \cdots & \begin{matrix} I_J \\ 0 \end{matrix} & * \end{array}\right]. \tag{3.109}$$

Since there are at least $K-1$ block-columns containing J pivots, $\operatorname{rank}(\widetilde{\mathbf{H}}) \geq KJ - J$. We previously established in (3.105) that $\operatorname{rank}(\mathbf{H}) \geq \operatorname{rank}(\widetilde{\mathbf{H}})$, hence $\operatorname{rank}(\mathbf{H}) \geq KJ - J$. We also already established in (3.101) that $\operatorname{rank}(\mathbf{H}) \leq KJ - J$, and therefore it should hold that $\operatorname{rank}(\mathbf{H}) = KJ - J$, which is what had to be proven. □

3.D Accumulation points are fixed points

This section provides a proof for Theorem 3.4

From Corollary 3.1, all points in $(X^i)_i$ remain in a compact set. Since each compact set has at least one accumulation point $\overline{X}$, there exists an infinite subsequence of $(X^i)_i$ which converges to $\overline{X}$. Because the number of nodes is finite, there exists a node $k \in \mathcal{K}$ that acts as an updating node in an infinite number of iterations that are sampled in this subsequence. In other words, we can find some node k such that there exists a set of iteration indices $\mathcal{I}_k \subseteq \mathbb{N}$ such that $(X^i)_{i\in\mathcal{I}_k}$ converges to $\overline{X}$ and $(i \mod K)_{i\in\mathcal{I}_k} = (k)_{i\in\mathcal{I}_k}$, i.e., the iteration indices $\mathcal{I}_k$ correspond only to iterations where node k is the updating node in Algorithm 3. From Berge's Maximum Theorem [131], the continuity of $\widetilde{\mathcal{F}}_k$ and $g(W,V) \triangleq \|W - V\|_F$ implies that $\widetilde{\mathcal{M}}_k$ and thus $\mathcal{M}_k = C_k\widetilde{\mathcal{M}}_k$ as in (3.38) are upper semicontinuous. This implies that any accumulation point $\overline{X}^{(+1)}$ of the set $(X^{i+1})_{i\in\mathcal{I}_k}$ must satisfy

$$\overline{X}^{(+1)} \in \mathcal{M}_k(\overline{X}). \tag{3.110}$$

Lemma 3.3. Let $\mathcal{I}_k \subseteq \mathbb{N}$ be such that $(X^i)_{i\in\mathcal{I}_k}$ converges to $\overline{X}$ and $(i \mod K)_{i\in\mathcal{I}_k} = (k)_{i\in\mathcal{I}_k}$, i.e., we only consider iterates related to some node k. Then if $\widetilde{\mathcal{F}}_k : \mathbb{R}^{M\times Q} \rightrightarrows \mathbb{R}^{\widetilde{M}_Q\times Q}$ is a continuous mapping, $\overline{X}$ is a fixed point of the map $\mathcal{M}_k$, i.e., $\mathcal{M}_k(\overline{X}) = \{\overline{X}\}$.

The proof of this lemma is given at the end of this section. From Lemma 3.3, we have $\mathcal{M}_k(\overline{X}) = \{\overline{X}\}$ (i.e., $\overline{X}$ is a fixed point of $\mathcal{M}_k$), (3.110) implies that

$$\overline{X}^{(+1)} = \overline{X}. \tag{3.111}$$

Inductively applying the above argument for the new subsequence $(X^{i+1})_{i\in\mathcal{I}_k}$ and accumulation point $\overline{X}^{(+1)}$ and for node $k+1 \mod K$ yields

$$\overline{X}^{(+l)} = \overline{X}, \quad \forall l \geq 0. \tag{3.112}$$

As $\overline{X}^{(+l)}$ is *any* accumulation point $(X^{i+l})_{i\in\mathcal{I}_k}$, all the accumulation points of $(X^{i+l})_{i\in\mathcal{I}_k}$ are equal to $\overline{X}$ and all the sequences

$$(X^i)_{i\in\mathcal{I}_{(k+l \mod K)}} \triangleq (X^{i+l})_{i\in\mathcal{I}_k} \tag{3.113}$$

converge to the same point $\overline{X}$. From Lemma 3.3 (here applied to the node $k+l \mod K$ instead of k), $\overline{X}^{(+l)}$ is a fixed point of $\mathcal{M}_{(q+l \mod K)}$ and $\overline{X}$ is therefore a fixed point of $\mathcal{M}_k$ for any k, proving the first part of the theorem.

We now prove that $\lim_{i\to+\infty} \|X^{i+1} - X^i\|_F = 0$ by contradiction. Let us assume that the above statement is not true. We first note that $X, W \to \|X - W\|_F$ is a continuous mapping, and X^i and X^{i+1} both live in a compact set (see beginning of the proof). Since the continuous image of a compact set is itself a compact set, $(\|X^{i+1} - X^i\|_F)_i$ has at least one accumulation point. There must therefore be some index set $\mathcal{I}$ such that

$$\lim_{i\in\mathcal{I}\to\infty} \|X^{i+1} - X^i\|_F > 0, \tag{3.114}$$

that is, there is one convergent subsequence converging to a point different from zero. Indeed, if zero was the only accumulation point, the sequence would be convergent (see Lemma 3.4). Furthermore, based on the same reasoning as the beginning of this proof, there is some $\mathcal{I}'_k \subseteq \mathcal{I}$ such that $(X^i)_{i\in\mathcal{I}'_k}$ is a convergent sequence such that $(i \mod K)_{i\in\mathcal{I}'_k} = (k)_{i\in\mathcal{I}'_k}$ for some k. We have shown above that the convergence of such a sequence $(X^i)_{i\in\mathcal{I}'_k}$ implied that

$$\lim_{i\in\mathcal{I}'_k\to\infty} X^i = \lim_{i\in\mathcal{I}'_k\to\infty} X^{i+1}. \tag{3.115}$$

Therefore, by continuity of the Frobenius norm, it must be that

$$\lim_{i \in \mathcal{I}'_k \to \infty} \|X^{i+1} - X^i\|_F = 0. \tag{3.116}$$

As (3.116) contradicts (3.114), every convergent subsequence of $(\|X^{i+1} - X^i\|_F)_{i\in\mathbb{N}}$ converges to 0 and $(\|X^{i+1} - X^i\|_F)_{i\in\mathbb{N}}$ is therefore a convergent sequence. □

Proof of Lemma 3.3. From Corollary 3.1, all points in $(X^i)_i$ remain in a compact set therefore $(X^{i+1})_{i\in\mathcal{I}_k}$ has an accumulation point $\overline{X}^{(+1)}$. The continuity of $\widetilde{\mathcal{F}}_k$, and thus of $C_k(X)\widetilde{\mathcal{F}}_k$, implies that it is also upper semicontinuous. Therefore, we have (by definition, see [125])

$$\overline{X}^{(+1)} \in C_k(\overline{X})\widetilde{\mathcal{F}}_k(\overline{X}). \tag{3.117}$$

We can now prove that

$$\min_{W \in \mathcal{S}_k(\overline{X})} f(W) = f(\overline{X}^{(+1)}) = f(\overline{X}). \tag{3.118}$$

The first equality directly follows from (3.117) and the definition (3.35) of $\widetilde{\mathcal{F}}_k$, that is

$$\min_{W \in \mathcal{S}_k(\overline{X})} f(W) = \min_{\widetilde{W}_q : C_q(\overline{X})\widetilde{W}_q \in \mathcal{S}_k(\overline{X})} f(C_q(\overline{X})\widetilde{W}_q) = f(X) \tag{3.119}$$

$\forall X \in C_k(\overline{X})\widetilde{\mathcal{F}}_k(\overline{X})$. The second equality in (3.118) follows from the fact that $\overline{X}$ is an accumulation point and that f is continuous together with the fact that $(f(X^i))_i$ is monotonically decreasing (Theorem 3.1) (i.e., all accumulation points have the same objective value). Because of (3.118), and since $\overline{X}$ is by definition in $\mathcal{S}_k(\overline{X})$, it must be that $\overline{X} \in C_k(\overline{X})\widetilde{\mathcal{F}}_k(\overline{X})$ and thus $[\overline{X}_k^T, I_Q, \ldots, I_Q]^T \in \widetilde{\mathcal{F}}_k(\overline{X})$. Using this in (3.37) with X replaced by $\overline{X}$ results in $\{\overline{X}\} = \mathcal{M}_k(\overline{X})$. □

3.E Proof of convergence to a single point

This section provides a proof for Theorem 3.5.

Let us assume that the mapping (3.39) corresponding to the DASF algorithm has a finite set of fixed points denoted Φ. As the set of fixed points is finite, it must be that there exists some $\delta > 0$ such that for any pair of fixed points $\overline{X}, \overline{W}$ $\|\overline{X} - \overline{W}\|_F > \delta$.

Lemma 3.4. In a compact metric space, a sequence converges to the set of its accumulation points. Additionally, the sequence converges if and only if it has a single accumulation point.

The proof of this lemma is given at the end of this section. From Lemma 3.4 (see the end of this appendix for a proof), as the sublevel sets of f are compact, $(X^i)_i$ converges to the set of its accumulation points $\mathcal{A}$. From Theorem 3.4, this set $\mathcal{A}$ must be a subset of Φ, and therefore finite.

We will now show that $\mathcal{A}$ must be a singleton. From Lemma 3.4, we have

$$\forall \varepsilon, \exists i_\varepsilon > 0 : \inf_{W \in \mathcal{A}} \|W - X^i\|_F < \varepsilon, \quad \forall i > i_\varepsilon, \tag{3.120}$$

while Theorem 3.4 implies

$$\forall \varepsilon, \exists i_\varepsilon > 0 : \|X^{i+1} - X^i\|_F < \varepsilon, \quad \forall i > i_\varepsilon. \tag{3.121}$$

From (3.120) and (3.121), there exists an $i_\varepsilon > 0$ such that

$$\begin{aligned} \forall i > i_\varepsilon \; \exists \overline{X}, \overline{X}^{(+1)} \in \mathcal{A} : & \|\overline{X} - X^i\|_F < \delta/3, \\ & \|\overline{X}^{(+1)} - X^{i+1}\|_F < \delta/3, \\ & \|X^{i+1} - X^i\|_F < \delta/3. \end{aligned} \tag{3.122}$$

We then have from the triangle inequality

$$\begin{aligned} \|\overline{X}^{(+1)} - \overline{X}\|_F &\leq \|\overline{X}^{(+1)} - X^i\|_F \\ &+ \left|\left|X^{i+1} - X^i\right|\right|_F + \|X^{i+1} - \overline{X}^{(+1)}\|_F < \delta. \end{aligned} \tag{3.123}$$

If $\overline{X} \neq \overline{X}^{(+1)}$ then this would imply that $\|\overline{X}^{(+1)} - \overline{X}\|_F > \delta$ (by definition of δ). However, this would be a contradiction with (3.123). Therefore, $\overline{X}$ and $\overline{X}^{(+1)}$ must be equal. Since (3.122) holds for any i, we find by induction that $\mathcal{A}$ is a singleton. From Lemma 3.4, this results in the convergence of $(X^i)_i$. □

Proof of Lemma 3.4. Let $(X^i)_i$ be some sequence in a compact metric space. Let $\mathcal{A}$ denote the set of accumulation points of $(X^i)_i$. We have $\forall X^* \in \mathcal{A}, \exists \mathcal{I} \subseteq \mathbb{N} : \forall \varepsilon > 0, \exists i_\varepsilon > 0 : \|X^* - X^i\|_F < \varepsilon, \quad \forall i \in \mathcal{I} > i_\varepsilon$. We wish to prove that $\forall \varepsilon > 0, \exists i_\varepsilon > 0 : \inf_{X \in \mathcal{A}} \|X - X^i\|_F < \varepsilon, \quad \forall i > i_\varepsilon$. Let us assume that our claim is not true. Then

$$\exists \varepsilon > 0, \forall i_\varepsilon > 0, \exists i > i_\varepsilon : \inf_{X \in \mathcal{A}} \|X - X^i\|_F \geq \varepsilon, \tag{3.124}$$

which implies that there exists some infinite set $\mathcal{I} \subseteq \mathbb{N}$ such that

$$\exists i_\varepsilon > 0 : \inf_{X \in \mathcal{A}} \|X - X^i\|_F \geq \varepsilon, \quad \forall i \in \mathcal{I} > i_\varepsilon. \tag{3.125}$$

Since the space is compact, the subsequence $(X^i)_{i \in \mathcal{I}}$ has itself a convergent subsequence converging to a point in $\mathcal{A}$, contradicting (3.125).

The convergence in the case of a single accumulation point follows directly from the previous result and the converse is a well-known result. □

Part II

Extensions of the DASF framework

4 | Improving the tracking performance and adaptivity of the DASF algorithm

This chapter is largely based on **C. A. Musluoglu**, M. Moonen, and A. Bertrand, "Improved Tracking for Distributed Signal Fusion Optimization in a Fully-Connected Wireless Sensor Network," in Proceedings of the *2022 30th European Signal Processing Conference (EUSIPCO)*, pp. 1836-1840, 2022.

ABSTRACT | As discussed in Chapters 2 and 3, although the DASF algorithm adaptively learns the relevant second order statistics from the collected sensor data, accuracy problems can arise if the spatial covariance structure of the signals is rapidly changing. In this chapter, we propose a method to improve the tracking or convergence speed of the DASF algorithm in a setting where signal statistics change relatively fast. While the improved tracking increases communication cost, we demonstrate that this tradeoff is efficient in the sense that an L-fold increase in bandwidth results in an R times faster convergence with $R \gg L$. We focus on fully-connected sensor networks, where a broadcast communication protocol can be used, and refer to [132, Chapter 3.5.2] for an extension of this method to more general topologies.

4.1 Introduction

In practice, the iterations of the DASF algorithm are spread over time, i.e., over different batches of signal observations (samples), thereby exploiting the stationarity of the underlying signals. This results in an adaptive algorithm that is able to track the changes in the signal statistics. However, this is a valid argument as long as the changes in the statistical parameters of the network-wide signal $\mathbf{y}$ (e.g., their spatial covariance matrix) are slower than the algorithm convergence rate, which is not always the case in practice. In this chapter, we propose a modified scheme for the DASF algorithm in a fully-connected WSN, which improves the tracking speed by re-using and re-transmitting previously measured observations in a bandwidth-efficient way, avoiding the centralization of the full data measured at each node. Although this leads to additional communication costs, we show that the resulting tradeoff can be made efficient by synergistically combining the data exchange protocol with the computations within the DASF algorithm. In the last section of this chapter, we demonstrate the improvements made by the proposed method on two different algorithms from the DASF family.

4.2 Efficient communication for improved tracking

In the DASF algorithm, each batch of N samples of the local signal $\mathbf{y}_k$ measured at node k is used in only one iteration i, i.e., the batch $\{\mathbf{y}_k(\tau)\}_{\tau=iN}^{(i+1)N-1}$ at iteration i. This might lead to a slow convergence/tracking speed, in particular for large networks (i.e., large K, with K the number of nodes in the network). To see this, note that it takes at least K iterations before each local variable X_k has been updated once by its corresponding node k, which happens only after sample time $t = KN$ (i.e., K batches of N samples). In order to improve the tracking performance, one could perform $R > 1$ iterations per $N-$sample batch. However, a straightforward realization of such a scheme would result in an $R-$fold increase in the communication cost. We will show that a specific re-transmission scheme that exploits the broadcast nature of the transmissions, and which synergistically combines the re-transmissions with the computations within the DASF algorithm, can actually make this tracking-vs-bandwidth tradeoff very efficient in the sense that the bandwidth increase is smaller than R.

Let us define $\mathcal{T}_m \triangleq \{(m-1)N, \ldots, mN-1\}$, where m is a strictly positive integer, and let R be the number of iterations over which the signal batch $\{\mathbf{y}(\tau)\}_{\tau\in\mathcal{T}_m}$ will be used. This means that the $m-$th batch $\{\mathbf{y}(\tau)\}_{\tau\in\mathcal{T}_m}$ will be

used over the iterations $\mathcal{I}_m \triangleq \{(m-1)R, \ldots, mR-1\}$ of the DASF algorithm, hence $m = \lceil (i+1)/R \rceil$. In Algorithm 1, we had $R = 1$ and therefore $m = i+1$ at every iteration, i.e., the batch index m and iteration i always update at the same pace. If $R > 1$, each new batch of N samples will lead to multiple DASF iterations, leading to faster convergence/tracking properties, i.e., more nodes can update per batch.

For $R > 1$, let us first consider a straightforward approach where we apply the steps of Algorithm 1 algorithm while re-using the same batch R times. As $X_k^{iT}\mathbf{y}_k$ is a $Q-$channel signal, each broadcast has a communication cost in $\mathcal{O}(NQ)$ per node, where $\mathcal{O}$ represents the Landau notation, and $\mathcal{O}(NQK)$ over the full network. This approach would then cost $\mathcal{O}(NQKR)$ per batch since each node would have to re-compress and re-transmit the same batch of samples R times. In this case, the communication cost increases linearly with R.

In the remainder of this section, we propose a more efficient scheme, which results in a much smaller and more scalable increase in communication cost. For each *new* batch $\{\mathbf{y}_k(\tau)\}_{\tau\in\mathcal{T}_m}$, i.e., for each increment of m, all nodes k first broadcast the compressed data batch $\{X_k^{iT}\mathbf{y}_k(\tau)\}_{\tau\in\mathcal{T}_m}$ to the entire network of nodes, where i is the index corresponding to the first iteration where this new batch of samples is used. Assuming a broadcast communication protocol, this requires only one transmission per node, which is the same as in a single iteration of Algorithm 1. Every node in the network then has access to

$$\{\widehat{\mathbf{y}}_k^i(\tau)\}_{\tau\in\mathcal{T}_m} = \{X_k^{iT}\mathbf{y}_k(\tau)\}_{\tau\in\mathcal{T}_m},\ \forall k. \tag{4.1}$$

$\widehat{B}_k^i = X_k^{iT}B_k$ is also broadcast initially. After this initial broadcast, the first updating node (say node q) solves its local problem

$$\begin{aligned} \widetilde{\mathbb{P}}_q^i : \underset{\widetilde{X}_q}{\text{minimize}} \quad & \varphi\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i\right) \\ \text{subject to} \quad & \eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i\right) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i\right) = 0 \ \ \forall j \in \mathcal{J}_E, \end{aligned} \tag{4.2}$$

obtaining $\widetilde{X}_q^*$. Partitioning $\widetilde{X}_q^*$ as

$$\widetilde{X}_q^* = [X_q^{(i+1)T}, G_{1,q}^{(i+1)T}, \ldots, G_{q-1,q}^{(i+1)T}, G_{q+1,q}^{(i+1)T}, \ldots, G_{K,q}^{(i+1)T}]^T \tag{4.3}$$

then gives a new X_q^{i+1} and matrices $G_{k,q}^{i+1}$, such that every node $k \in \mathcal{K}\backslash\{q\}$ updates their variable as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q, \\ X_k^i G_{k,q}^{i+1} & \text{if } k \neq q, \end{cases} \tag{4.4}$$

Algorithm 4: DASF in a fully-connected network with data re-use

X^0 initialized randomly, $i \leftarrow 0$, $m \leftarrow 1$.
repeat
  1) Every node k collects $\{\mathbf{y}_k(\tau)\}_{\tau\in\mathcal{T}_m}$, compresses each sample as $\widehat{\mathbf{y}}_k^i(\tau) = X_k^{iT}\mathbf{y}_k(\tau)$ to obtain $\{\widehat{\mathbf{y}}_k^i(\tau)\}_{\tau\in\mathcal{T}_m}$ and broadcasts the compressed signals to all other nodes along with $\widehat{B}_k^i = X_k^{iT}B_k$.
  repeat R times
    $q \leftarrow (i \mod K) + 1$.
    at Node q **do**
      2a) $\widetilde{X}_q^* \leftarrow \operatorname{argmin} \widetilde{\mathbb{P}}_q^i$.
      If the solution of (4.2) is not unique, select the solution which minimizes $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$.
      2b) Partition $\widetilde{X}_q^*$ as in (4.3).
      2c) Broadcast $\{\widehat{\mathbf{y}}_q^{i+1}(\tau) = X_q^{(i+1)T}\mathbf{y}_q(\tau)\}_{\tau\in\mathcal{T}_m}$, $\widehat{B}_q^{i+1} = X_q^{(i+1)T}B_q$ and $\{G_{k,q}^{i+1}\}_{k\neq q}$ to all other nodes.
    end
    3) For all k, every node updates X_k^{i+1} according to (4.4) and recomputes $\{\widehat{\mathbf{y}}_k^{i+1}(\tau)\}_{\tau\in\mathcal{T}_m} = \{G_{k,q}^{(i+1)T}\widehat{\mathbf{y}}_k^i(\tau)\}_{\tau\in\mathcal{T}_m}$ and $\widehat{B}_k^{i+1} = G_{k,q}^{(i+1)T}\widehat{B}_k^i$.
    $i \leftarrow i + 1$
  $m \leftarrow m + 1$

For the other nodes to be aware of these changes, node q broadcasts its compressed signal samples and deterministic matrix obtained using its new estimation X_q^{i+1}, as well as the matrices $G_{k,q}^{i+1}$, i.e., node q broadcasts

$$\begin{aligned}&\{\widehat{\mathbf{y}}_q^{i+1}(\tau) = X_q^{(i+1)T}\mathbf{y}_q(\tau)\}_{\tau\in\mathcal{T}_m},\\ &\widehat{B}_q^{i+1} = X_q^{(i+1)T}B_q \text{ and } \{G_{k,q}^{i+1}\}_{k\neq q},\end{aligned} \tag{4.5}$$

so that the next updating node has access to the batch of samples of node q as well as $\widehat{B}_q^{i+1}$, this time compressed by the new X_q^{i+1} and can update the compressed signals and deterministic matrix of the other nodes (received in the previous iteration i) using:

$$\begin{aligned}\forall k \neq q : &\{X_k^{(i+1)T}\mathbf{y}_k(\tau)\}_{\tau\in\mathcal{T}_m} = \{G_{k,q}^{(i+1)T}X_k^{iT}\mathbf{y}_k(\tau)\}_{\tau\in\mathcal{T}_m},\\ &X_k^{(i+1)T}B_k = G_{k,q}^{(i+1)T}X_k^{iT}B_k.\end{aligned} \tag{4.6}$$

Table 4.1: Per batch communication cost for the original, straightforward, and proposed methods.

Original	Straightforward	Proposed
$\mathcal{O}(NQK)$	$\mathcal{O}(NQKR)$	$\mathcal{O}(NQ(K+R-1))$

In the remaining $R-1$ iterations in $\mathcal{I}_m$, the only transmissions done in the network are the broadcasts (4.5) performed by the updating node (which changes at each iteration). Each node therefore needs to re-compress and re-broadcast the same batch of samples only when it becomes an updating node. The modifications are presented in Algorithm 4.

As mentioned previously, $X_k^{iT}\mathbf{y}_k$ is a $Q-$channel signal, hence the initial broadcast has a communication cost in $\mathcal{O}(NQ)$ per node and $\mathcal{O}(NQK)$ over the full network. Then, for each iteration $i \in \mathcal{I}_m$ (except the last one), the extra data to be transmitted (only by a single node each time) is given in (4.5), which has a cost in $\mathcal{O}(NQ)$ (assuming N is large[1]). Note that the final updating node does not have to broadcast (4.5) as the next iteration starts with a new batch. Thus, over the R iterations in $\mathcal{I}_m$, the total communication cost in the network is in $\mathcal{O}(NQ(K+R-1))$ per batch, as opposed to a cost of $\mathcal{O}(NQKR)$ in the case of the "straightforward" approach mentioned earlier. Algorithm 4 thus allows to perform $R>1$ iterations per batch of N samples, albeit with a larger communication bandwidth than Algorithm 1 (see Table 4.1). To appreciate why this is an efficient and desirable tradeoff, consider the case where $R=K$. In this case, the bandwidth of Algorithm 4 is doubled compared to Algorithm 1, yet the former can perform K iterations in the time when the latter can only perform a single iteration (note that a batch increment is directly coupled to the sample time of the sensors). As a result, the convergence or tracking speed is K times faster, whereas the bandwidth is only doubled. Moreover, the communication cost is $K/2$ times more efficient than the "straightforward" approach.

4.3 Simulations

In this section, we demonstrate the performance of the proposed method for two different spatial filtering algorithms: LCMV beamforming and the EVD problem. In both cases, we consider a WSN with $K=5$ nodes, each measuring

[1] In practice, the number of observations in a single batch is large in order to estimate the statistics of $\widetilde{\mathbf{y}}_k$ (see Section 2.2.2). As a result, the transmission of $\widehat{B}_k$'s and the parameters $G_{k,q}$ becomes negligible. We therefore neglect these in the asymptotic communication cost.

the signal $\mathbf{y}_k$ on $M_k = 5$ channels. The network-wide signal $\mathbf{y}$ has then $M = 25$ channels and is modeled as

$$\mathbf{y}(t) = \Pi \cdot \mathbf{d}(t) + \mathbf{n}(t), \tag{4.7}$$

where $\mathbf{d}$ represents an $S-$dimensional source signal, Π is an $M \times S$ mixture matrix and $\mathbf{n} \in \mathbb{R}^M$ is an additive noise signal. The values over time of $\mathbf{d}$ have been chosen independently at random, following the zero-mean Gaussian distribution with variance σ_d^2, i.e., $\mathcal{N}(0, \sigma_d^2)$. Similarly, each entry of $\mathbf{n}$ independently follows $\mathcal{N}(0, \sigma_n^2)$. On the other hand, the entries of the mixture matrix Π are drawn from $\mathcal{N}(0, \sigma_\Pi^2)$. An overview of the values taken by the different parameters of the problems we will consider can be found in Table 4.2.

In our experiments, the spatial covariance matrix $K_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ is estimated using batches of N samples of $\mathbf{y}$ during the time intervals $\mathcal{T}_m$ with the following estimation:

$$K_{\mathbf{yy}} \approx \frac{1}{N} \sum_{\tau \in \mathcal{T}_m} \mathbf{y}(\tau)\mathbf{y}^T(\tau). \tag{4.8}$$

Each estimation of the covariance matrix is then used R times before using newly collected data to obtain a new estimation of $K_{\mathbf{yy}}$. Additionally, we also change the entries of Π at random points in time by adding random numbers drawn from $\mathcal{N}(0, \sigma_\Delta^2)$, to simulate a change in signal statistics and to assess the tracking performances for different values of R (see further). The metric we use to compare different approaches that we present below is the MedSE:

$$\epsilon(i) = \text{median}\left(\frac{\|X^i - X^{*i}\|_F^2}{\|X^{*i}\|_F^2}\right), \tag{4.9}$$

where X^i is only evaluated at the last update of the batch, i.e., $i = mR - 1$, and where X^{*i} represents a solution of the global optimization problem. Note that X^{*i} depends on the iteration i since it depends on $K_{\mathbf{yy}}$ which changes during the simulations due to the additive noise on Π we mentioned previously. For each batch, X^{*i} is computed by replacing $K_{\mathbf{yy}}$ with the approximation (4.8).

4.3.1 LCMV beamforming

Let us consider the following LCMV beamforming problem

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \mathbb{E}\left[\|X^T\mathbf{y}(t)\|^2\right] = \text{trace}(X^T K_{\mathbf{yy}} X) \\ \text{subject to} \quad & B^T X = F, \end{aligned} \tag{4.10}$$

Table 4.2: Summary of parameters used in the tracking simulations.

Experiment	Section 4.3.1	Section 4.3.2
Q	3	2
K	5	
M	25	
S	10	4
Signal Statistics	$\sigma_d^2 = 1$, $\sigma_n^2 = 0.2$, $\sigma_\Pi^2 = 0.1$, $\sigma_\Delta^2 = 0.01$	$\sigma_d^2 = 0.5$, $\sigma_n^2 = 0.1$, $\sigma_\Pi^2 = 0.1$, $\sigma_\Delta^2 = 0.01$
N	100000	
Monte Carlo Runs	100	

where the steering matrix $B \in \mathbb{R}^{M\times J}$ contains the first J columns of the mixture matrix Π, where we fix $J = Q = 3$, and the elements of $F \in \mathbb{R}^{J\times Q}$ are drawn independently at random from $\mathcal{N}(0,1)$. In this experiment, we take the number of sources to be $S = 10$. The unique solution of (4.10) is given by $X^* = K_{\mathbf{yy}}^{-1} B \left(B^T K_{\mathbf{yy}}^{-1} B\right)^{-1} F$ (see Appendix B.2).

4.3.2 EVD problem

We consider the EVD problem

$$\begin{aligned} &\underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad \mathbb{E}\left[\|X^T\mathbf{y}(t)\|_F^2\right] = \operatorname{tr}(X^T K_{\mathbf{yy}} X) \\ &\text{subject to} \quad X^T X = I_Q, \end{aligned} \tag{4.11}$$

where $Q = 2$ and $S = 4$. Problem (4.11) has infinitely many solutions, however, a common choice is to take an orthogonal matrix containing in its columns the eigenvectors of the covariance matrix $K_{\mathbf{yy}}$ corresponding to its $Q = 2$ largest eigenvalues (with arbitrarily chosen signs for each column), i.e., $X^* = \mathrm{EVD}_2(K_{\mathbf{yy}})$.

4.3.3 Results

Figure 4.1 shows the results for $R \in \{1, 2, 3, 4, K = 5\}$ for both problems. For the case of $R = 1$, the results are obtained using Algorithm 1, whereas we used

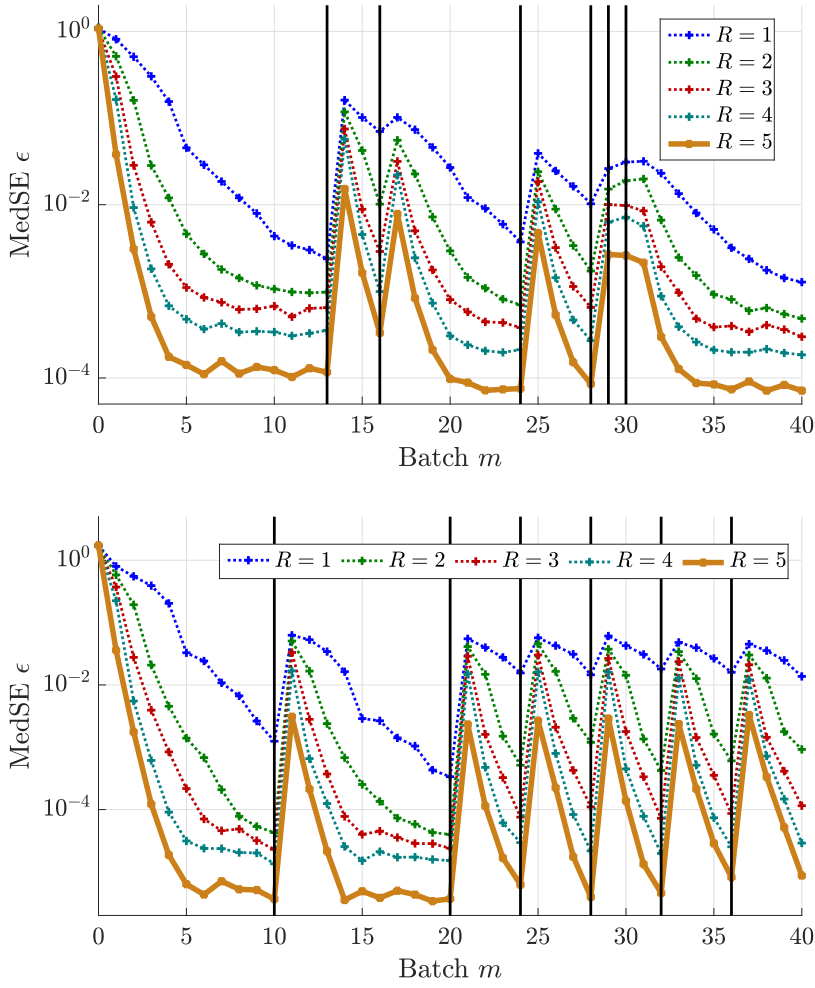


Figure 4.1: Comparison of the MedSE for different values of R in a network with $K = 5$ nodes. The vertical lines in black represent time instants where Π changes. *Top*: Solving the LCMV problem described in Section 4.3.1. *Bottom*: Solving the EVD problem described in Section 4.3.2.

Algorithm 4 when $R \geq 2$. The plots represent the MedSE ϵ over 100 Monte-Carlo runs. The time instants where Π changes are indicated by black vertical lines. We observe that for a fixed Π, the minimal MedSE value is achieved faster for larger values of R. After every change in statistics, a slight increase in the

MedSE can be observed but can be corrected if Π does not change too rapidly. A case worth highlighting is when $R = K = 5$ which is particularly robust to the changes of Π. On the other hand, if the signal statistics change too fast, a larger value of R is necessary for tracking performances to stay accurate. In the case of $R = 1$, i.e., the original (FC-)DASF algorithm (Algorithm 1), the MedSE does not attain its minimal value when the signal statistics change very frequently. Note that the bandwidth is only doubled in the case of $R = 5$ (vs. the case $R = 1$), while achieving a $5-$fold faster convergence, thereby allowing to track well the changes in Π.

4.4 Conclusion

In this chapter, we have proposed a scheme to improve the convergence or tracking speed of the DASF algorithm for a fully-connected WSN, which is important for making the algorithm robust to settings where the statistical properties of the signals change fast, as could be the case in practical settings. This is done by efficiently communicating the data across the network so as to re-use the same batch of observations over multiple iterations while keeping the communication cost as low as possible. Results obtained from various simulations allowed us to confirm our claims.

An extension of this scheme for general network topologies is possible, as explained in [132, Chapter 3.5.2]. The main difference with the fully-connected case is that a retransmission is required at each level of the tree, starting at the updating node q. This is because the $G-$matrices of the child nodes of the neighbors of node q depend on the update done at node q (see (3.26)).

5 | The DASF framework for node-specific problems

This chapter is largely based on **C. A. Musluoglu** and A. Bertrand, "A Distributed Adaptive Algorithm for Node-Specific Signal Fusion Problems in Wireless Sensor Networks," Accepted in Proceedings of the *2023 31st European Signal Processing Conference (EUSIPCO)*, pp. 1-5, 2023.

ABSTRACT | The DASF framework presented in Chapters 2 and 3 assumes that there is a common goal across the nodes, i.e., all nodes collaborate to optimize the same network-wide objective function. However, some applications require node-specific objectives, which are linked via a common latent target signal subspace. In this chapter, we propose the distributed adaptive node-specific signal fusion (DANSF) algorithm which builds upon the DASF framework and extends it to allow for such node-specific spatial filtering problems.

5.1 Introduction

Until this point, we have considered that the nodes in a WSN estimate a *common* spatial filter, translated mathematically as an optimization problem that is common to every node in the network. However, a node-specific spatial filter can be desired, e.g., when each node is interested in different source signals or differently filtered versions of the same source signal(s) [10, 133, 134]. Each node then has a different optimization problem to solve, i.e., the problem is node-specific, yet can be related, e.g., via a common latent signal model, in which case a joint processing is desirable.

Distributed algorithms for some particular node-specific problems have been studied, such as MMSE [88, 89, 135] and LCMV beamforming [40, 114, 136, 137], although each problem has been treated separately in the literature. Other distributed algorithms for node-specific problems have been proposed in [138, 139], but can generally not be applied to spatial filtering due to the way the data is partitioned across the network.

Although the DASF algorithm considers a single common optimization problem to be solved across the network, i.e., it does not allow for node-specific optimization problems, it can be extended to include the latter. In this chapter, we propose the distributed adaptive node-specific signal fusion (DANSF) algorithm, which builds upon the DASF framework to solve generic node-specific problems in a distributed fashion, where the node-specific objectives are linked via a common latent target signal subspace. The proposed method contains specific algorithms previously described for the MMSE and LCMV problems [40, 88, 89, 114, 135, 136] as special cases and allows for a generalization over a larger class of problems, such as robust variations of the MMSE/LCMV, regularized total least squares, ℓ_2 regularizations terms, etc. The DANSF algorithm converges to the optimal solution of each node-specific problem under the same assumptions as the original DASF algorithm.

5.2 Problem setting

In this section, we define the problem setting of the DANSF algorithm. Relevant definitions and statements from previous chapters are repeated here for completeness and to avoid ambiguities. We consider a sensor network with K nodes given in the set $\mathcal{K} = \{1, \ldots, K\}$ and connected following a topology given by a graph $\mathcal{G}$, where each link between two nodes k and l implies that nodes k and l can share data with each other. Every node senses an M_k-channel

signal $\mathbf{y}_k$ so that the network-wide signal can be defined as

$$\mathbf{y}(t) = [\mathbf{y}_1^T(t), \ldots, \mathbf{y}_K^T(t)]^T \in \mathbb{R}^M, \tag{5.1}$$

at every time sample t and where $M = \sum_k M_k$. As in previous chapters, the signal $\mathbf{y}$ should be viewed as a multivariate stochastic variable, assumed to be ergodic and (short-term) stationary. Each node k acts as a data sink and is interested in finding its own optimal (network-wide) spatial filter $X_k \in \mathbb{R}^{M\times Q}$ and the corresponding filter output $X_k^T\mathbf{y}(t)$, which should satisfy a node-specific optimality condition.

Remark 5.1. In this chapter, and in contrast to the previous (and next) ones, X_k does not correspond to the block of the common variable X corresponding to node k, but to the "global" variable at node k. The block of X_k corresponding to node l will be denoted as X_{kl}, as defined further.

We envisage a generic problem statement where we assume that the optimal filter X_k is the solution of an optimization problem of the following form (for node k):

$$\begin{aligned} \mathbb{P}_k: \underset{X_k\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \varphi_k\big(X_k^T\mathbf{y}(t), X_k^TB\big) \\ \text{subject to} \quad & \eta_{k,j}\big(X_k^T\mathbf{y}(t), X_k^TB\big) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_{k,j}\big(X_k^T\mathbf{y}(t), X_k^TB\big) = 0 \ \ \forall j \in \mathcal{J}_E, \end{aligned} \tag{5.2}$$

where the sets $\mathcal{J}_I$ and $\mathcal{J}_E$ represent the index sets for inequality and equality constraints respectively. Some examples of problems of the form (5.2) are shown in Table 5.1. The functions φ_k and $\eta_{k,j}$, $j \in \mathcal{J}_I \cup \mathcal{J}_E$ are real and scalar-valued functions, while the subscript k specifies that a function or variable is specific for node k. As in previous chapters, the filter output $(X_k^T\mathbf{y}(t))$ is a stochastic variable, hence the functions in (5.2) should contain an operator to extract a real-valued deterministic quantity from this term, such as an expectation operator. Note that a solution $X_k^*(t)$ of (5.2) depends on the time sample t, as the statistics of the signal $\mathbf{y}$ are allowed to change in time. The proposed DANSF algorithm will act as a block-adaptive filter that estimates and tracks the changes in the data statistics, similar to the DASF algorithm. However, from short-term stationarity, we assume that a solution $X_k^*(t)$ of (5.2) changes slowly in time compared to the convergence rate of the DANSF algorithm. Therefore, we omit the time-dependence of solutions of (5.2) for mathematical tractability and assume stationarity within convergence time of the algorithm. Additionally, from the ergodicity of the signal $\mathbf{y}$, we assume that the statistical

Table 5.1: Examples of problems with node-specific objectives as in (5.2).

Problem	Cost function to minimize at node k	Constraints
MMSE	$\mathbb{E}[\|\mathbf{d}_k(t) - X_k^T\mathbf{y}(t)\|^2]$	—
LCMV	$\mathbb{E}[\|X_k^T\mathbf{y}(t)\|^2]$	$X_k^T B = H_k$

quantities related to $\mathbf{y}$ (and other signals if present) can be estimated using a large enough number of samples N (see Section 2.2.2).

B is again a deterministic $M \times L$ matrix independent of the time index t, commonly encountered to enforce a structure on the variable X_k (as in, e.g., LCMV in Table 5.1). Similarly to the partitioning of $\mathbf{y}$ in (5.1), these deterministic matrices are assumed to be obtained by stacking $M_k \times L$ matrices B_k, where B_k is supposed to be available at node k, i.e., $B = [B_1^T, \ldots, B_K^T]^T$. Moreover, for a fixed node k, we also allow Problem (5.2) to have multiple variables, signals, and deterministic matrices (in addition to X_k, $\mathbf{y}$ and B, respectively), which are however not represented in (5.2) for conciseness (see discussion in Section 2.3.3). We assume that every parameter in (5.2), except $\mathbf{y}$ and B, is available at node k. For example, the signal $\mathbf{d}_k$ and parameter H_k in the MMSE and LCMV examples of Table 5.1 should be available at node k.

For each node k, let us denote by $\mathcal{X}_k^*$ the solution set of $\mathbb{P}_k$ and $X_k^* \in \mathcal{X}_k^*$ a specific solution. We then make the following assumption on the set of problems $\mathbb{P}_k$, which links the solutions across the nodes.

Assumption 4. There exists a set of invertible $Q \times Q$ matrices $\{D_{k,l}\}_{(k,l)\in\mathcal{K}^2}$ such that for any pair (k, l) of nodes, the solutions $X_k^* \in \mathcal{X}_k^*$ and $X_l^* \in \mathcal{X}_l^*$ satisfy $X_k^* = X_l^* \cdot D_{k,l}$.

These properties were also exploited in [88, 89, 114, 135–137] for the design of distributed fusion algorithms for MMSE and LCMV problems. Note that Assumption 4 implies that the solutions at the different nodes are instantaneous mixtures of each other. However, this also allows modeling convolutive mixtures if the problem is formulated in the short-time Fourier transform domain, as for example in [88, 136]. Taking the example of the MMSE problem in Table 5.1,

Assumption 4 is satisfied if $\mathbf{d}_k(t) = D_{k,l}^T \mathbf{d}_l(t)$. Indeed, we can then write

$$\begin{aligned} X_k^* &= R_{\mathbf{yy}}^{-1} R_{\mathbf{yd}_k} = R_{\mathbf{yy}}^{-1} \cdot \mathbb{E}[\mathbf{y}(t)\mathbf{d}_k^T(t)] \\ &= R_{\mathbf{yy}}^{-1} \cdot \mathbb{E}[\mathbf{y}(t)\mathbf{d}_l^T(t) D_{k,l}] = R_{\mathbf{yy}}^{-1} R_{\mathbf{yd}_l} D_{k,l} = X_l^* \cdot D_{k,l}, \end{aligned} \tag{5.3}$$

where $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ and $R_{\mathbf{yd}_k} = \mathbb{E}[\mathbf{y}(t)\mathbf{d}_k^T(t)]$ for every node k. The assumption that $\mathbf{d}_k(t) = D_{k,l}^T \mathbf{d}_l(t)$ is true if, e.g., the desired signals at the different nodes are all different mixtures from the same set of latent sources. This is common in, e.g., hearing aids where the acoustic mixing process needs to be preserved for spatial hearing [88, 136]. For the LCMV example of Table 5.1, Assumption 4 is satisfied if $H_k = D_{k,l}^T H_l$, which is a common case in wireless acoustic sensor networks [40, 136, 137].

In this chapter, we propose a unifying algorithmic framework, which has [40, 88, 89, 114, 135, 136] as special cases, while also admitting new problems (see, e.g., Section 5.4), assuming they can be written in the form (5.2).

5.3 DASF for node-specific problems

In this section, we derive the DANSF algorithm which extends the DASF framework to also admit node-specific problems of the form (5.2). We refer to Chapter 2 for a thorough presentation of the DASF algorithm, from which we here only extract the essential ingredients to allow us to define the proposed DANSF algorithm.

At each iteration i, an updating node $q \in \mathcal{K}$ is selected and the network represented by the graph $\mathcal{G}$ is temporarily pruned to a tree $\mathcal{T}^i(\mathcal{G}, q)$, such that there is a unique path between any pair of nodes in the network. The pruning function is a free design choice, however it should not remove any links between node q and its neighbors $n \in \mathcal{N}_q$ as discussed in Section 2.4.3, where $\mathcal{N}_q$ denotes the set of neighboring nodes of node q. As in the previous chapters, the updating node q changes at each iteration. In the remaining parts of the algorithm derivation, the neighbors of any node are defined to be the ones after pruning the network, i.e., based on the edges of $\mathcal{T}^i(\mathcal{G}, q)$.

Let us partition each X_k, i.e., the network-wide spatial filter that generates the desired node-specific output signal for node k, as

$$X_k = [X_{k1}^T, \ldots, X_{kK}^T]^T, \tag{5.4}$$

such that each X_{kl} is $M_l \times Q$. For each $k \in \mathcal{K}$, we define X_{kk}, the $k-$th block of X_k, to be the compressor at node k. At iteration i, every node $k \neq q$ uses its

current estimate of X_{kk} to compress the local M_k-dimensional sensor signal $\mathbf{y}_k$ into a $Q-$dimensional one, with $Q < M_k$. A similar compression applied to node k's matrix B_k leads to

$$\widehat{\mathbf{y}}_k^i \triangleq X_{kk}^{iT}\mathbf{y}_k,\ \widehat{B}_k^i \triangleq X_{kk}^{iT}B_k, \tag{5.5}$$

where X_{kk}^0 is initialized randomly for each k. The nodes will then fuse and forward their compressed data (5.5) towards node q as explained next. As was also the case in previous algorithms of the DASF framework, each node k should transmit a batch of N time samples of $\widehat{\mathbf{y}}_k^i$, where N should be chosen such that there are enough samples to estimate the relevant statistics of $\widehat{\mathbf{y}}_k^i$ that are used in the objective and constraints of (5.2) (in most practical examples, this consists of all the second-order statistics). At each iteration, a different block of N samples is used, so that in the case of changes in statistics of the signal $\mathbf{y}$, the proposed method can adaptively track these changes.

The data flow and processing done at the nodes are analogous to the DASF algorithm, however we repeat these steps in this chapter to avoid ambiguities related to the new notation we use (the additional index for node-specificity). Each node k first waits until receiving data from all of its neighbors except one, which we denote as n. Node k then fuses the data received from its neighbors $l \in \mathcal{N}_k \backslash \{n\}$ with its compressed data (5.5) and the result is then transmitted to node n, which receives N samples of

$$\widehat{\mathbf{y}}_{k\to n}^i \triangleq X_{kk}^{iT}\mathbf{y}_k + \sum_{l\in\mathcal{N}_k\backslash\{n\}} \widehat{\mathbf{y}}_{l\to k}^i, \tag{5.6}$$

where $\widehat{\mathbf{y}}_{l\to k}^i$ is the data received by node k from its neighbor l. Note that the second term of (5.6) is recursive and vanishes for nodes with a single neighbor, i.e., leaf nodes, implying that the recursion in (5.6) is bootstrapped at the leaf nodes of the tree $\mathcal{T}^i(\mathcal{G}, q)$. The fused data eventually reaches node q, which receives N samples of

$$\widehat{\mathbf{y}}_{n\to q}^i = X_{nn}^{iT}\mathbf{y}_n + \sum_{k\in\mathcal{N}_n\backslash\{q\}} \widehat{\mathbf{y}}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i, \tag{5.7}$$

from all its neighbors $n \in \mathcal{N}_q$. In (5.7), $\mathcal{B}_{nq}$ is defined to be the subgraph of $\mathcal{T}^i(\mathcal{G}, q)$ which contains node n, obtained after cutting the link between nodes n and q (see Figure 2.3). A similar recursion applies for the compressed matrices $\widehat{B}_k^i$, and we define $\widehat{B}_{n\to q}^i$ as the matrix analogous to (5.7) received by node q. Defining $\mathcal{N}_q \triangleq \{n_1, \ldots, n_{|\mathcal{N}_q|}\}$, the data collected at node q can be structured as

$$\begin{aligned} \widetilde{\mathbf{y}}_q^i &\triangleq [\mathbf{y}_q^T, \widehat{\mathbf{y}}_{n_1\to q}^{iT}, \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T \in \mathbb{R}^{\widetilde{M}_q}, \\ \widetilde{B}_q^i &\triangleq [B_q^T, \widehat{B}_{n_1\to q}^{iT}, \ldots, \widehat{B}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T \in \mathbb{R}^{\widetilde{M}_q\times L} \end{aligned} \tag{5.8}$$

where $\widetilde{M}_q = M_q + |\mathcal{N}_q| \cdot Q$. Using the local data in (5.8), node q creates a compressed version of its original problem $\mathbb{P}_q$:

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \varphi_q\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\big) \\ \text{subject to} \quad & \eta_{q,j}\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\big) \leq 0 \;\; \forall j \in \mathcal{J}_I, \\ & \eta_{q,j}\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\big) = 0 \;\; \forall j \in \mathcal{J}_E. \end{aligned} \tag{5.9}$$

Note that (5.9) has the same objective and constraint functions as (5.2) (for $k = q$), hence a solver for (5.2) can also be used locally at node q to solve the compressed problem (5.9). We saw this to be an interesting feature of the DASF framework, which is inherited in DANSF as well.

Node q then solves its local problem (5.9) to obtain $\widetilde{X}_q^*$:

$$\widetilde{X}_q^* \triangleq \underset{\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}} \; \varphi_q\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\big), \tag{5.10}$$

where $\widetilde{\mathcal{S}}_q^i$ denotes the constraint set of (5.9). In the cases where (5.9) has multiple solutions, we choose $\widetilde{X}_q^*$ by minimizing $\|\widetilde{X}_q - \widetilde{X}_q^i\|_F$ over all solutions of (5.9), with

$$\widetilde{X}_q^i \triangleq [X_{qq}^{iT}, I_Q, \ldots, I_Q]^T \tag{5.11}$$

and where X_{qq}^i corresponds to the current estimate of the compressor X_{qq} of node q. The optimal solution $\widetilde{X}_q^*$ is then partitioned as

$$\widetilde{X}_q^* = [X_{qq}^{(i+1)T}, G_{qn_1}^{(i+1)T}, \ldots, G_{qn_{|\mathcal{N}_q|}}^{(i+1)T}]^T, \tag{5.12}$$

with each $G-$matrix being $Q \times Q$. The new estimate of the variable $X_q = [X_{q1}^T, \ldots, X_{qK}^T]^T$ at iteration i is then

$$X_{qk}^{i+1} = \begin{cases} X_{qq}^{i+1} & \text{if } k = q, \\ X_{kk}^i G_{qn}^{i+1} & \text{if } k \in \mathcal{B}_{nq},\ n \in \mathcal{N}_q. \end{cases} \tag{5.13}$$

For nodes $k \neq q$, a new estimate of their variable X_k can be estimated in the following way. Node q first transmits

$$\mathbf{z}_q^{i+1}(t) \triangleq \widetilde{X}_q^{*T} \widetilde{\mathbf{y}}_q^i(t), \quad Z_q^{i+1} \triangleq \widetilde{X}_q^{*T} \widetilde{B}_q^i, \text{ and } X_q^{i+1} \tag{5.14}$$

to each node k, which can either be broadcast by node q or transmitted following the pruned network topology (see [135] for a discussion on efficient ways to

achieve this). Note that again N samples of $\mathbf{z}_q^{i+1}$ should be sent by node q to the other nodes. Node k then solves

$$\begin{aligned} \underset{F_{kq}\in\mathbb{R}^{Q\times Q}}{\text{minimize}} \quad & \varphi_k\big(F_{kq}^T\mathbf{z}_q^{i+1}(t), F_{kq}^T Z_q^{i+1}\big) \\ \text{subject to} \quad & \eta_{k,j}\big(F_{kq}^T\mathbf{z}_q^{i+1}(t), F_{kq}^T Z_q^{i+1}\big) \leq 0 \;\; \forall j \in \mathcal{J}_I, \\ & \eta_{k,j}\big(F_{kq}^T\mathbf{z}_q^{i+1}(t), F_{kq}^T Z_q^{i+1}\big) = 0 \;\; \forall j \in \mathcal{J}_E, \end{aligned} \tag{5.15}$$

such that a new estimate of its variable X_k at iteration i is

$$X_k^{i+1} = X_q^{i+1} F_{kq}^{i+1}, \tag{5.16}$$

where F_{kq}^{i+1} is a solution of (5.15) at node k. Note that the compressor at node k, i.e., X_{kk} is part of X_k, i.e., the compression matrix of node k is also updated by (5.16).

It is also possible to consider the case where only the updating node q obtains a new estimate of its variable X_q as in (5.13), while the other nodes keep their current estimate, i.e., $X_k^{i+1} = X_k^i$ for $k \neq q$. Such an updating scheme is encountered in, e.g., [89]. The additional communication of the terms in (5.14), and the computation of F_{kq}^{i+1} by solving (5.15) at each node $k \neq q$ could then be omitted. This alternative therefore creates a tradeoff between computational and communication efficiency, and convergence speed.

Remark 5.2. Similar to Lemma 2.1 and its proof, it can be shown that for each node k, X_k^i defined as in (5.13) and (5.16) always satisfies the constraints of the corresponding problem (5.2) at node k for every $i > 0$. Furthermore, $\widetilde{X}_q^* \in \widetilde{\mathcal{S}}_q^i \Leftrightarrow X_q^{i+1} \in \mathcal{S}_q$, where $\widetilde{\mathcal{S}}_q^i$ and $\mathcal{S}_q$ denote the constraint sets of (5.9) and (5.2), respectively, for node q.

Since only the compressor matrices X_{kk} (for all k) play a role within the algorithm (as these define the transmitted signals), the update of the blocks X_{kl} with $k \neq l$ can be omitted, unless the nodes are explicitly interested in knowing the coefficients of the full matrix X_k. However, in most applications, the filter output signal $\mathbf{z}_k(t) = X_k^T\mathbf{y}(t)$ is sought after, rather than the filter X_k itself, which can be computed at each node k as

$$\mathbf{z}_k^{i+1}(t) \triangleq \begin{cases} \widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i(t) & \text{if } k = q, \\ F_{kq}^{(i+1)T}\mathbf{z}_q^{i+1}(t) & \text{if } k \neq q, \end{cases} \tag{5.17}$$

without keeping track of other subblocks X_{kq} for $k \neq q$. This is because the filtering of subblocks X_{kq} is done at node q instead of node k, using the compressor X_{qq}, of which the output is transformed with F_{kq} at node k.

Algorithm 5: Distributed Adaptive Node-Specific Signal Fusion (DANSF) algorithm

X_{kk}^0 initialized randomly for each k, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) The network $\mathcal{G}$ is pruned into a tree $\mathcal{T}^i(\mathcal{G}, q)$.
  2) Each node k collects N samples of $\mathbf{y}_k$, compress these to N samples of $\widehat{\mathbf{y}}_k^i$ and also compute $\widehat{B}_k^i$ as in (5.5).
  3) The nodes sum-and-forward their compressed data towards node q via the recursive rule (5.6) (and a similar rule for the $\widehat{B}_k^i$'s). Node q eventually receives N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ given in (5.7) and similarly, $\widehat{B}_{n\to q}^i$, from all its neighbors $n \in \mathcal{N}_q$.
  at Node q **do**
    4a) Compute the solution of (5.9) to obtain $\widetilde{X}_q^*$. If the solution is not unique, select $\widetilde{X}_q^*$ which minimizes $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$ with $\widetilde{X}_q^i$ defined in (5.11).
    4b) Partition $\widetilde{X}_q^*$ as in (5.12).
    4c) Update the estimate of X_q as in (5.13).
    4d) Disseminate Z_q^{i+1} and N samples of $\mathbf{z}_q^{i+1}$ as in (5.14) within the tree to each data sink node.
  end
  5) Nodes $k \neq q$ update X_k according to (5.16) by solving (5.15) and can estimate their filtered output as in (5.17).
  $i \leftarrow i + 1$

At each iteration, this process is repeated by selecting a different updating node. Algorithm 5 summarizes the steps of the DANSF algorithm described above. We note that [88,89,114,136] are special cases of this algorithm. The method is able to adapt to and track changes in the signal statistics of $\mathbf{y}$, as is the case for the original DASF algorithm. This is because a new block of N samples, e.g., $\{\mathbf{y}(\tau)\}_{\tau=iN}^{(i+1)N-1}$, is measured and used at each iteration to solve (5.9), i.e., different iterations of the DANSF algorithm are spread over different sample blocks across the time dimension, similar to an adaptive filter, making each X_k^i an estimate of $X_k^*(t)$, where $t = iN$. Note that the communication cost per node and per iteration is at most $2NQ$ samples, independent of the topology (note that $Q < M_k$). In a setting where all the raw data is relayed to a fusion center via multi-hop transmissions, this cost is much higher (i.e., PNM_k, where P is the number of hops between node k and the fusion center).

Theorem 5.1 below gives a convergence guarantee in cost for the DANSF algorithm. The full convergence for each node k to a solution $X_k^* \in \mathcal{X}_k^*$ of $\mathbb{P}_k$ is described afterwards in Theorem 5.2. The proof of the latter is omitted but follows similar steps as in Chapter 3.

Theorem 5.1. Let us denote by $(\varphi_k^i)_i$ the sequence of function values $\varphi_k\left(X_k^{iT}\mathbf{y}(t), X_k^{iT}B\right)$, $\forall k \in \mathcal{K}$, obtained from Algorithm 5. Then, the sequence $(\varphi_k^i)_i$ is non-increasing and converges for each k, i.e., the cost function at each node is monotonically decreasing.

Proof. For conciseness, we omit the matrix B in this proof (it can be treated similarly to $\mathbf{y}$). For the updating node q, at iteration i, we have that $\varphi_q\big(\widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i(t)\big) \leq \varphi_q\big(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\big)$ for any $\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i$, since $\widetilde{X}_q^*$ solves the local problem (5.9). Moreover, since $\widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i$ (see Remark 5.2), we have $\varphi_q\big(\widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i(t)\big) \leq \varphi_q\big(\widetilde{X}_q^{iT}\widetilde{\mathbf{y}}_q^i(t)\big)$. This shows a monotonic decrease of the cost at the current updating node q. For the case $k \neq q$, let us (hypothetically) assume that the updating node q would have used the cost function of node k instead, i.e., it solves

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \varphi_k\big(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\big) \\ \text{subject to} \quad & \eta_{k,j}\big(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\big) \leq 0 \;\; \forall j \in \mathcal{J}_I, \\ & \eta_{k,j}\big(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\big) = 0 \;\; \forall j \in \mathcal{J}_E, \end{aligned} \tag{5.18}$$

instead of (5.9). As mentioned earlier, (5.9) and (5.18) are compressed versions of the problem (5.2) for node q and node k respectively, i.e., the data $\mathbf{y}$ of (5.2) is replaced by $\widetilde{\mathbf{y}}_q^i$ in (5.9) and (5.18). Therefore, the local problems (5.9)-(5.18) satisfy Assumption 4, i.e., there exists a matrix $\widetilde{D}_{q,k}$ such that $\widetilde{X}_q^* = \widetilde{X}_k^* \cdot \widetilde{D}_{q,k}$. This implies that node q also optimizes φ_k up to a transformation with a $Q \times Q$ matrix. The latter transformation is compensated for by finding a proper transformation F_{kq} at node k by solving (5.15). Since this argument holds for any iteration i, and from the relationship $X_k^{i+1} = X_q^{i+1}F_{kq}^{i+1}$ in (5.16), we have $\varphi_k(X_k^{(i+1)T}\mathbf{y}(t)) \leq \varphi_k(X_k^{iT}\mathbf{y}(t))$ even though node q optimizes φ_q instead of φ_k at iteration i. The sequence $(\varphi_k^i)_i$ is therefore non-increasing for each k. Since these sequences are respectively lower bounded by the minimal value of φ_k achieved for X_k^* over the constraint set of $\mathbb{P}_k$ in (5.2), they are converging sequences. □

Theorem 5.2 (Proof Omitted)**.** Suppose that, for each node k, Problem (5.2) satisfies Assumption 4 and the conditions for convergence of the original DASF algorithm (see Chapter 3). Then the convergence results of the DASF algorithm apply to each sequence $(X_k^i)_i$, $k \in \mathcal{K}$, obtained from the DANSF algorithm. In particular, if each problem (5.2) satisfies the conditions of the DASF algorithm for convergence to their optimum, the sequences also converge respectively to an optimal point $X_k^* \in \mathcal{X}_k^*$ of Problem (5.2).

Note that this also implies that each node has access to its optimal node-specific filter output $\mathbf{z}_k^*(t) = X_k^{*T}\mathbf{y}(t)$ for all samples collected after convergence of the algorithm. Sensor observations used *during* convergence of the algorithm are fused suboptimally (similar to how an adaptive filter initially produces suboptimal filter outputs). The algorithm can be used in an adaptive or tracking context if the dynamics of the statistics change slowly, i.e., slower than the convergence time of the DANSF algorithm.

5.4 Simulations

To demonstrate the generic nature of the DANSF algorithm, we consider a new "toy" problem that has not been investigated in the literature in the context of node-specific spatial filtering. For results on practical scenarios, we refer readers to [40, 88, 89, 114, 135, 136], which can be shown to be special cases of DANSF. The node-specific problem we consider at each node k is formulated as:

$$\begin{aligned} \underset{X_k \in \mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \operatorname{tr}(X_k^T B_k) \\ \text{subject to} \quad & \operatorname{tr}(X_k^T R_{\mathbf{yy}} X_k) \leq 1, \end{aligned} \tag{5.19}$$

where $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$. Taking $B_k \neq 0$, the unique solution of the problem is given by (see Appendix B.8)

$$X_k^* = -R_{\mathbf{yy}}^{-1} B_k \cdot \sqrt{\operatorname{tr}(B_k^T R_{\mathbf{yy}}^{-1} B_k)^{-1}}. \tag{5.20}$$

For each k, we take $B_k = B \cdot D_k$, where each element of B and D_k's are drawn from a Gaussian distribution with zero-mean and variance 1, i.e., $\mathcal{N}(0,1)$. Note that Problem (5.19) satisfies Assumption 4 as $X_k^* = X_l^* \cdot D_{k,l}$, where

$$D_{k,l} = D_l^{-1} D_k \cdot \sqrt{\frac{\operatorname{tr}(B_l^T R_{\mathbf{yy}}^{-1} B_l)}{\operatorname{tr}(B_k^T R_{\mathbf{yy}}^{-1} B_k)}}. \tag{5.21}$$

We will first look at the convergence properties of the DANSF algorithm under stationarity conditions and for different topologies. We consider the network-wide signal $\mathbf{y}$ to follow a mixture model given by

$$\mathbf{y}(t) = \Pi \cdot \mathbf{d}(t) + \mathbf{n}(t), \tag{5.22}$$

where each element of $\mathbf{d} \in \mathbb{R}^Q$ and $\mathbf{n} \in \mathbb{R}^M$ independently follows $\mathcal{N}(0, 0.5)$ and $\mathcal{N}(0, 0.1)$ respectively, at each time sample. On the other hand, every entry of the mixture matrix $\Pi \in \mathbb{R}^{M \times Q}$ is drawn from $\mathcal{N}(0, 0.2)$. In a second experimental setting, we will show that the DANSF algorithm is able to track changes in the statistical properties of $\mathbf{y}$, demonstrating its adaptive properties in a non-stationary setting. Throughout these experiments, we take $Q = 3$, $K = 10$ and $M_k = 7$ for every $k \in \mathcal{K}$. In every experiment described below, $\mathcal{T}^i(\cdot, q)$ is taken to be the shortest path pruning function.

5.4.1 Stationary setting

At each iteration i of the DANSF algorithm, each node k transmits $N = 10^4$ samples of $\widehat{\mathbf{y}}_k^i$ to the updating node q. The updating node q then solves its local problem (5.9) given by

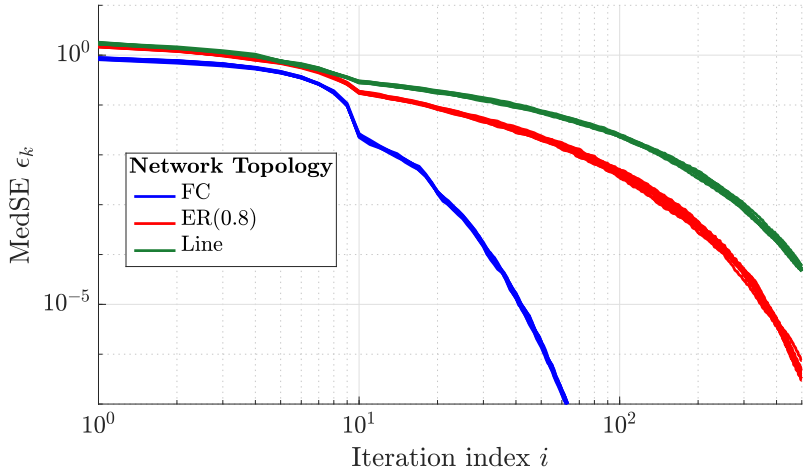
$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \operatorname{tr}(\widetilde{X}_q^T \widetilde{B}_q^i) \\ \text{subject to} \quad & \operatorname{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} \widetilde{X}_q) \leq 1, \end{aligned} \tag{5.23}$$

where $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$ are defined as in (5.8) and $R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} = \mathbb{E}[\widetilde{\mathbf{y}}_q^i(t) \widetilde{\mathbf{y}}_q^{iT}(t)]$.

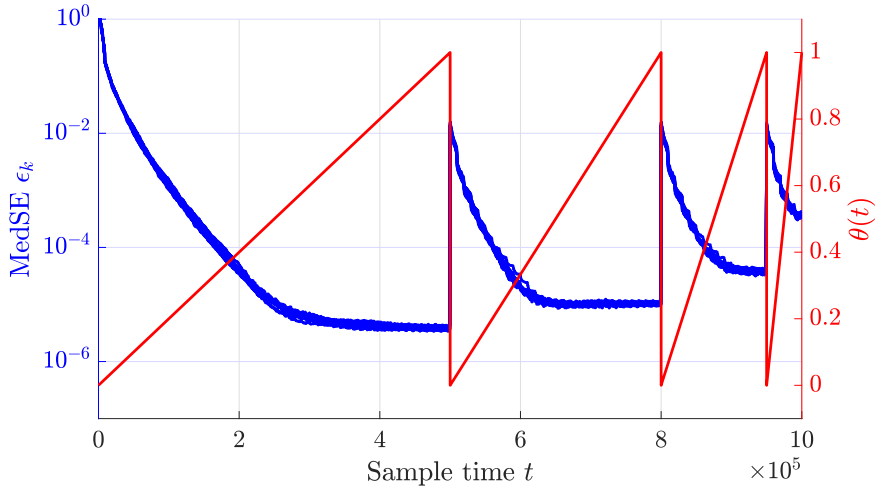
The performance of the DANSF algorithm is assessed by computing the relative MedSE ϵ_k:

$$\epsilon_k(i) = \operatorname{median}\left(\frac{\|X_k^i - X_k^*\|_F^2}{\|X_k^*\|_F^2} \right), \tag{5.24}$$

for each node k, where the median is taken over 100 Monte Carlo runs. X_k^* is the solution (5.20) of (5.19) for node k. Figure 5.1a shows the MedSE ϵ_k for every node k and for different network topologies namely fully-connected networks, networks with line topologies, and networks with randomly generated topologies. We observe that the DANSF algorithm converges to X_k^* for every node k of the network, as stated in Theorem 5.2, although at different convergence rates for different topologies. Fully-connected networks converge the fastest, while the slowest convergence rate is obtained for networks with a line topology. A similar result was observed for the DASF algorithm, where networks with more connected topologies lead to faster convergence rates (see Section 2.5.2).



(a) MedSE ϵ_k for all nodes k of the DANSF algorithm in a stationary setting for various network topologies. The descriptions of the different network topologies are provided in Table 2.3.



(b) MedSE ϵ_k for all nodes k in an adaptive setting in a randomly generated network using the Erdős-Rényi model with connection probability 0.8. The relationship between the time t and the iterations i is given by $i = \lfloor t/N \rfloor$.

Figure 5.1: Convergence results of the DANSF algorithm in stationary (top) and adaptive (bottom) settings.

Additionally, we see from Figure 5.1a that each node's estimate of its variable X_k converges to the respective optimal value X_k^* without large deviations in

convergence rate between different nodes.

5.4.2 Adaptive setting

In this second experimental setting, we demonstrate that the DANSF algorithm is able to track changes in the signal statistics, therefore making the method adaptive. We consider the same setting as previously, except $N = 10^3$ and the mixture matrix Π changes at each time sample t. In particular, we have $\Pi(t) = \Pi_0 \cdot (1 - \theta(t)) + (\Pi_0 + \Delta) \cdot \theta(t)$, where the elements of Π_0 and Δ are independently drawn from $\mathcal{N}(0, 0.2)$ and $\mathcal{N}(0, 0.01)$ respectively, and θ is given in Figure 5.1b. This implies that the stationarity condition does not hold and X_k^*'s are now time-dependent. At each iteration, they are estimated as

$$X_k^{*i} = -(\widehat{R}_{\mathbf{yy}}^i)^{-1} B_k \cdot \sqrt{\mathrm{tr}\left(B_k^T (\widehat{R}_{\mathbf{yy}}^i)^{-1} B_k\right)^{-1}}, \tag{5.25}$$

where

$$\widehat{R}_{\mathbf{yy}}^i = \frac{1}{N} \sum_{\tau=iN}^{iN+N-1} \mathbf{y}(\tau)\mathbf{y}^T(\tau). \tag{5.26}$$

Figure 5.1b shows the MedSE taken over 100 Monte Carlo runs:

$$\epsilon_k(i) = \mathrm{median}\left(\frac{\|X_k^i - X_k^{*i}\|_F^2}{\|X_k^{*i}\|_F^2}\right), \tag{5.27}$$

over time, with $i = \lfloor t/N \rfloor$, for each node k. At each run, the topology of the network is a randomly generated graph using the Erdős-Rényi model with connection probability 0.8. In Figure 5.1b, we see that an abrupt change in the signal statistics implies a sudden increase in the MedSE value, while the algorithm gradually adapts to slow rates of change, as can be seen by the decreasing MedSE. We also observe that the algorithm reaches MedSE floors instead of converging towards 0 as in the stationary case. This is due to the fact that the optimal solutions X_k^*, for each node k, change at each iteration, where the error floor is higher for faster rates of change in the signal statistics.

5.5 Conclusion

We have proposed the DANSF algorithm to solve node-specific signal fusion problems in a distributed fashion over a network. The DANSF algorithm builds upon the principles of the DASF framework and extends it to problems with different optimization problems at each node, yet with coupled solution sets,

which leads to analogous convergence results between both algorithms. We provided a proof for the convergence in cost, which showed that we obtain a monotonic decrease of the cost at each node. Simulations of the DANSF algorithm applied to a new problem validated our convergence claims.

6 | The trace ratio optimization problem and a new distributed algorithm

This chapter is largely based on **C. A. Musluoglu** and A. Bertrand, "Distributed Adaptive Trace Ratio Optimization in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3653-3670, 2021, ©2021 IEEE and **C. A. Musluoglu** and A. Bertrand, "Distributed trace ratio optimization in fully connected sensor networks," in Proceedings of the *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 1991-1995, 2021.

ABSTRACT | The trace ratio optimization (TRO) problem consists of finding an orthonormal basis for the discriminative subspace that maximizes the ratio of two trace operators on two covariance matrices corresponding to two distinctive classes or signal components. The TRO problem is encountered in various signal processing problems such as dimensionality reduction, signal enhancement, and discriminative analysis. This chapter is intended as a case study of a specific spatial filtering problem but will also be used to introduce a modification to the DASF algorithm to make it computationally more efficient for fractional programs. This new algorithm will be shown to converge to an optimal solution of the TRO problem, without suffering from loss in convergence rates compared the DASF approach. Simulation results are provided to validate and complement the theoretical results.

6.1 Introduction

The TRO problem is the maximization of the ratio between two quadratic forms, where the optimization variable X is required to have orthonormal columns. Mathematically, it can be expressed as

$$\begin{aligned} \underset{X}{\text{maximize}} \quad & \frac{\text{tr}(X^T A X)}{\text{tr}(X^T B X)} \\ \text{subject to} \quad & X^T X = I, \end{aligned} \tag{6.1}$$

where A and B are positive definite matrices. Note that Problem (6.1) is different from a standard generalized Rayleigh quotient optimization, due to the presence of additional orthogonality constraints on X, and therefore the solution is not provided by a generalized eigenvalue decomposition (GEVD). The TRO problem is commonly encountered in signal processing and machine learning tasks [93–98] when the goal is to find an optimal subspace projection for data points belonging to two different classes. The subspace is generated by the columns of X while A and B are chosen in a way that reflects the difference between the classes. For example, in motor imagery brain-computer interfaces based on EEG, one class could represent EEG activity during imaginary right-hand movement, while the other could represent EEG activity during right-foot movement. In that case, A would correspond to the covariance matrix of the signals resulting from one of these EEG activities while B would be the covariance matrix of the signals from the other one [113].

The TRO problem took its roots from Fisher's linear discriminant [140] and the Foley-Sammon transform (FST) [141, 142]. These are essentially "greedy" formulations of (6.1), in which X is replaced with a single-column vector. This single-column problem is then solved multiple times (for each column of X) while preserving orthogonality with respect to previously computed columns. However, the optimality does not hold for the space spanned by the whole set of vectors, because of the greedy computation method. This was pointed out in [19], while also providing a method to find a generalized optimal set. It was however mentioned in [93] that this latter technique suffers from separability issues on the projected set of vectors. Therefore, the generalized Foley-Sammon transform was defined in [93] as maximizing the ratio of two trace operators, which led to the TRO problem (6.1). Several methods to solve the general TRO problem have been described in the literature, using the Grassmann manifold [96], by semidefinite programming [98], and using an iterative method related to the Rayleigh quotient iteration [93–95].

In this chapter, we study the TRO problem in a distributed spatial filtering context. In the previous chapters, it was shown that the DASF algorithm

can be applied to the TRO problem, however, we will see that it requires solving multiple optimization problems at each node. Therefore, we propose a new approach based on the DASF framework that significantly reduces the computational cost at each updating node. The proposed distributed TRO (DTRO) algorithm is proven to converge to the solution of the centralized TRO problem with a convergence rate similar to the DASF algorithm.

The outline of the chapter is as follows. In Section 6.2, we review the centralized TRO problem and an algorithm to solve it. Then, in Section 6.3, we review the DASF algorithm applied to the TRO problem and propose another approach that reduces the computational cost. The communication and computational aspects of the proposed algorithms are then discussed in Section 6.4. Finally, we provide in Section 6.5 simulation results of the DTRO algorithms in various settings to validate its performance, along with discussions.

6.2 The trace ratio optimization problem

6.2.1 Problem definition and relationship to other problems

For convenience, and to introduce some new notation, we repeat the formal definition of the TRO problem. Given two positive definite matrices $A, B \in \mathbb{R}^{M\times M}$, the TRO problem aims at finding a matrix $X \in \mathbb{R}^{M\times Q}$ solving the following problem

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & r(X) \triangleq \frac{\operatorname{tr}(X^T A X)}{\operatorname{tr}(X^T B X)} \\ \text{subject to} \quad & X^T X = I_Q. \end{aligned} \tag{6.2}$$

The optimization variable X therefore contains Q orthonormal vectors in its columns spanning a subspace maximizing the objective r, typically with $Q \ll M$. A solution of (6.2) is given by

$$X^* = \mathrm{EVC}_Q(A - \rho^* B), \tag{6.3}$$

where ρ^* is the maximal value of r over the constraints $X^T X = I_Q$ [143]. It is noted that the solution of (6.2), is unique up to a unitary transformation [129], while it can be shown that all local maxima of the TRO problem are global maxima [144]. Additional details on the optimality conditions of the TRO problem are provided in Appendix B.4.

The matrix pair (A, B) of Problem (6.2) can have various interpretations depending on the application. For example, in linear discriminant analysis,

the aim is to tightly group points of a same class while separating each class from another in the best way possible [145], yielding A to correspond to the between-class scatter matrix of the data points, whereas B would be the within-class scatter matrix. In a multi-channel signal processing context, X can be interpreted as a collection of orthogonal filter banks applied to different states or components of a multi-channel signal, e.g., to discriminate between signal and noise or between two underlying states that alter the statistics of the signal. A and B would in that case represent the two covariance matrices corresponding to these two signal states or components (see also Section 6.3.1).

In various applications, the optimal discriminant vectors are often taken to be the generalized eigenvectors (GEVCs) of the ordered matrix pair (A, B) for easier computation of the optimal vectors, as in [19]. Mathematically, both problems are similarly formulated, the only difference being the constraint set. The optimization problem related to the GEVD problem can be written as

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{maximize}} \quad & r(X) = \frac{\text{tr}(X^T A X)}{\text{tr}(X^T B X)} \\ \text{subject to} \quad & X^T B X = I_Q. \end{aligned} \tag{6.4}$$

It can be shown [146] that the maximal value of r is the scaled sum of the largest generalized eigenvalues (GEVLs) of the matrix pair (A, B). A solution of (6.4) is then given by the GEVCs corresponding to those GEVLs. These GEVCs are defined as the columns of the $M \times Q$ matrix $X^\dagger$ satisfying

$$AX^\dagger = BX^\dagger \Lambda^\dagger, \tag{6.5}$$

where $\Lambda^\dagger \in \mathbb{R}^{Q \times Q}$ is a diagonal matrix with the GEVLs on its diagonal. It has been pointed out in previous studies ([93, 94, 129]) that although related, problems (6.2) and (6.4) differ in the general case, the only exception being the setting $Q = 1$. In that particular case, if ρ^* is the optimal value of (6.2) and $\rho^\dagger$ the optimal value of (6.4), then $\rho^* = \rho^\dagger$ and if $\mathbf{x}^\dagger \in \mathbb{R}^M$ is a solution of (6.4), then $\mathbf{x}^* = \|\mathbf{x}^\dagger\|^{-1}\mathbf{x}^\dagger \in \mathbb{R}^M$ is a solution of (6.2).

Despite the similarity of the solutions for $Q = 1$, the links between the solutions of both problems are less trivial for higher projection dimensions. It has been discussed in [93–95, 129] that both problems are not interchangeable, resulting in optimal projection matrices with different properties. In [95], it is pointed out that the GEVD problem can be described as a greedy way to solve the TRO problem. Interesting comparison arguments between both methods have been given in [94], while [96] explains that the GEVD problem will not necessarily give a larger maximal value for r. Even though the GEVD problem is easier to compute, it is noted in [147] that the solution of the TRO problem has

the advantage that it does not distort the metric structure of the signals, by enforcing orthogonality on the filters.

We also note that the TRO problem should not be confused with the problem referred to as the "ratio trace" [94], which can be considered as a relaxation of (6.4). This problem is obtained by replacing the trace operators in the objective with the determinant operator while removing the constraints. All possible GEVCs corresponding to the Q largest GEVLs of the matrix pair (A, B) are a subset of the solution set of the ratio trace problem, the latter also including matrices obtained by a scaling followed by an orthogonal transformation of the columns of $X^\dagger$ [94, 148].

6.2.2 The trace ratio optimization algorithm

Although there exist several ways to solve (6.2), we will only review the one presented in [94, 129], as it serves as the basis for the distributed algorithm presented in Section 6.3. The method shares similar steps with the Rayleigh quotient iteration method [149–152], used for computing eigenvalues.

The iterative algorithm to solve the TRO problem is constructed by transforming the optimization problem to a scalar one (i.e., in a single variable), by defining the auxiliary functions $f : \mathcal{S} \times \mathbb{R} \to \mathbb{R}$:

$$f(X, \rho) \triangleq \operatorname{tr}\left(X^T(A - \rho B)X\right), \tag{6.6}$$

where $\mathcal{S} \triangleq \{X \in \mathbb{R}^{M \times Q} : X^T X = I_Q\}$ and $g : \mathbb{R} \to \mathbb{R}$ as

$$g(\rho) \triangleq \max_{X \in \mathcal{S}} f(X, \rho). \tag{6.7}$$

As shown in [93], the following theorem relates g to the optimum ρ^* of (6.2).

Theorem 6.1. [93] We have the following relationships between g in (6.7) and the optimum ρ^* of (6.2):

- $g(\rho) = 0 \Leftrightarrow \rho = \rho^*$,
- $g(\rho) > 0 \Leftrightarrow \rho < \rho^*$,
- $g(\rho) < 0 \Leftrightarrow \rho > \rho^*$.

Algorithm 6: Trace Ratio Optimization (TRO) algorithm [94]

X^0 initialized randomly, $\rho^0 \leftarrow r(X^0)$, $i \leftarrow 0$

repeat

1) $X^{i+1} \leftarrow \mathrm{EVC}_Q(A - \rho^i B)$, where $\mathrm{EVC}_Q(\cdot)$ is defined as in Definition 1
2) $X^{i+1} \leftarrow X^{i+1} D^{i+1}$, where $D^{i+1} = \mathrm{argmin}_{D \in \mathcal{D}} \|X^{i+1} D - X^i\|_F$, with $\mathcal{D}$ the set of signature matrices, i.e., diagonal matrices containing either 1 or -1 on their diagonal
3) $\rho^{i+1} \leftarrow r(X^{i+1})$ where r is defined in (6.2)
$i \leftarrow i + 1$

Furthermore, it is shown in [93] that an X^* satisfying

$$X^* \in \underset{X \in \mathcal{S}}{\mathrm{argmax}}\ f(X, \rho^*), \tag{6.8}$$

solves the TRO problem (6.2), and the converse is also true, i.e., a solution of the TRO problem also maximizes $f(X, \rho^*)$ under the constraint $X \in \mathcal{S}$. The initial problem is therefore transformed into finding the root of the function g over the variable ρ. It is noted that the evaluation of $g(\rho)$ defined in (6.7) is equivalent to taking the sum of the Q largest eigenvalues (EVLs) of $(A - \rho B)$ [153]. Since the matrix $A - \rho B$ is symmetric, the variable X maximizing the function $f(\cdot, \rho)$ within the orthogonal constraint set $\mathcal{S}$ therefore corresponds to the Q eigenvectors (EVCs) corresponding to those EVLs, i.e., the X's that satisfy

$$(A - \rho B)X = X\Lambda, \tag{6.9}$$

where Λ is a diagonal matrix containing the Q largest EVLs. Additionally, if X contains the EVCs corresponding to the Q largest EVLs of $A - \rho B$ and $\mathrm{tr}(X^T(A - \rho B)X) = 0$, then $\rho = \rho^*$ and X is a solution of the TRO problem (6.2) [143].

The iterative step of the TRO algorithm in [94] is based on computing a new ρ using X from (6.9), substituting it in r defined in (6.2) and repeating this process. Algorithm 6 summarizes the iterative TRO solver from [94], which has a proven convergence to the optimal value ρ^* and an optimal argument X^*. In the general case, the convergence is quadratic [95].

Remark 6.1. Note that Algorithm 6 can be used to solve more general TRO problems of the form

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & \frac{\operatorname{tr}(X^TAX)}{\operatorname{tr}(X^TBX)} \\ \text{subject to} \quad & X^TCX = I_Q \end{aligned} \tag{6.10}$$

for symmetric matrices C by making the change of variable $W = \Sigma^{\frac{1}{2}}U^TX$, where U and Σ are obtained from the singular value decomposition of $C : C = U\Sigma U^T$. The problem solved by Algorithm 6 is then

$$\begin{aligned} \underset{W\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & \frac{\operatorname{tr}(W^T\Sigma^{-\frac{1}{2}}U^TAU\Sigma^{-\frac{1}{2}}W)}{\operatorname{tr}(W^T\Sigma^{-\frac{1}{2}}U^TBU\Sigma^{-\frac{1}{2}}W)} \\ \text{subject to} \quad & W^TW = I_Q, \end{aligned} \tag{6.11}$$

resulting in a solution W^*. The solution of (6.10) is finally given by $X^* = U\Sigma^{-\frac{1}{2}}W^*$.

Since the EVCs obtained in step 1 are unique up to a sign, the signs are chosen so as to minimize the norm of the difference between two iterations, which avoids "oscillations" in the sign of the columns when approaching convergence, as described in step 2. We note that, at convergence, equation (6.9) becomes

$$(A - \rho^*B)X^* = X^*\Lambda^*, \tag{6.12}$$

but the sign of the columns of X^* can be arbitrarily chosen, hence the solution of Algorithm 6 is only defined up to an arbitrary sign of the columns of X^*, and all outputs solve the TRO problem (6.2). Note that all solutions of Algorithm 6 are TRO solutions, but not all TRO solutions are a solution of Algorithm 6, i.e., the solution set of the latter is more restricted (with only a sign ambiguity instead of a unitary transformation ambiguity). This is an important fact that will be exploited later on in the convergence proof of the distributed TRO algorithm. In the remaining parts of this chapter, we denote by X^* a solution of (6.2) which is an output of Algorithm 6, i.e., X^* is a TRO solution that also satisfies (6.12).

6.3 Distributed methods for the TRO problem

6.3.1 Adaptive TRO in multi-channel signal processing

In a multi-channel signal processing context, we consider two M-channel time signals $\mathbf{y}(t)$ and $\mathbf{v}(t) \in \mathbb{R}^M$, where t is a sample time index, such that their covariance matrix $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}(t)^T]$ and $R_{\mathbf{vv}} = \mathbb{E}[\mathbf{v}(t)\mathbf{v}(t)^T]$ would replace B and A in the TRO objective:

$$\begin{aligned} \underset{X \in \mathbb{R}^{M\times Q}}{\text{maximize}} \quad & r(X) \triangleq \frac{\operatorname{tr}(X^T R_{\mathbf{vv}} X)}{\operatorname{tr}(X^T R_{\mathbf{yy}} X)} \\ \text{subject to} \quad & X \in \mathcal{S}. \end{aligned} \tag{6.13}$$

The solution of (6.13) could be interpreted as a discriminative spatial filter bank with M inputs and Q outputs, for which the total (summed) energy of the signals at the output can be used for discrimination between these two classes. Another example can be given in signal denoising [2], where $\mathbf{v}$ would represent "signal-plus-noise" segments and $\mathbf{y}$ would represent "noise-only" segments. In that case, the solution of (6.13) would act as an orthogonal spatial filter bank for joint dimensionality reduction and denoising purposes.

We assume that the signals are short-term stationary and ergodic, so that given a sufficiently large number of samples N of the signals $\mathbf{y}$ and $\mathbf{v}$, we can estimate the covariance matrices using an estimation well-suited for the application at hand, such as $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)] \approx \frac{1}{N}\sum_{\tau=t}^{t+N-1} \mathbf{y}(\tau)\mathbf{y}^T(\tau)$, and similarly for $\mathbf{v}$. The different iterations of Algorithm 6 can then be spread out over different time windows of N samples. The objective r can then be approximated as:

$$r(X) \approx \frac{\frac{1}{N}\sum_{\tau=t}^{t+N-1} \|X^T\mathbf{v}(\tau)\|^2}{\frac{1}{N}\sum_{\tau=t}^{t+N-1} \|X^T\mathbf{y}(\tau)\|^2}. \tag{6.14}$$

6.3.2 The DASF algorithm for the TRO problem

As in the previous parts of this text, let us consider a network represented by the graph $\mathcal{G}$, with K nodes belonging to a set $\mathcal{K} = \{1, \ldots, K\}$. Each node k measures two[1] M_k-dimensional signals $\mathbf{y}_k$ and $\mathbf{v}_k$. Stacking every node's signals together, we obtain the $M-$dimensional signals $\mathbf{y}$ and $\mathbf{v}$, with $M = \sum_{k\in\mathcal{K}} M_k$ and $\mathbf{y} = \left[\mathbf{y}_1^T, \ldots, \mathbf{y}_K^T\right]^T$, $\mathbf{v} = \left[\mathbf{v}_1^T, \ldots, \mathbf{v}_K^T\right]^T$. As in previous chapters, each node k has only access to its own observations $\mathbf{y}_k$ and $\mathbf{v}_k$, and the network-wide variable $X \in \mathbb{R}^{M\times Q}$ is partitioned as

$$X = \left[X_1^T, \ldots, X_K^T\right]^T, \tag{6.15}$$

[1] Note that $\mathbf{y}_k$ and $\mathbf{v}_k$ could also denote the same sensor signal recorded in two different conditions as explained in Section 6.2.1.

where $X_k \in \mathbb{R}^{M_k \times Q}$, corresponds to the part of X that is applied to the sensor signals of node k.

At iteration i, after selecting an updating node q (that changes at each iteration) and pruning the network so as to obtain a tree $\mathcal{T}^i(\mathcal{G}, q)$ (see Section 2.4.3), every node k compresses their data using their current estimate X_k^i of their local variable and transmits them towards node q. In the case of the TRO problem, we have two stochastic signals $\mathbf{y}$ and $\mathbf{v}$, therefore node q will receive N samples of

$$\begin{aligned}
\widehat{\mathbf{y}}_{n\to q}^i(t) &= X_n^{iT}\mathbf{y}_n(t) + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{\mathbf{y}}_{k\to n}^i(t) = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i(t), \\
\widehat{\mathbf{v}}_{n\to q}^i(t) &= X_n^{iT}\mathbf{v}_n(t) + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{\mathbf{v}}_{k\to n}^i(t) = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{v}}_k^i(t)
\end{aligned} \tag{6.16}$$

from all its neighbors $n \in \mathcal{N}_q$, where $\mathcal{B}_{nq}$ is the set of nodes containing node n obtained by cutting the link between n and q (see Figure 2.3). Note that the set $\mathcal{N}_k$ refers to the neighbors of node k in the pruned network, i.e., with respect to the graph $\mathcal{T}^i(\mathcal{G}, q)$. Additionally, the TRO problem has two deterministic matrices in its constraints: $(X^T I_M)\cdot(X^T I_M)^T$. However, as noted in Remark 2.5, it is sufficient for each node k to transmit towards node q the matrix $X_k^{iT}X_k^i$, instead of X_k^i. Therefore, node q also receives

$$\widehat{\Gamma}_{n\to q}^i \triangleq X_n^{iT}X_n^i + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{\Gamma}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} X_k^{iT}X_k^i \tag{6.17}$$

from its neighbors. The locally available data at node q is then given by

$$\begin{aligned}
\widetilde{\mathbf{y}}_q^i(t) &= [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_{n_1\to q}^{iT}(t), \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}(t)]^T, \\
\widetilde{\mathbf{v}}_q^i(t) &= [\mathbf{v}_q^T(t), \widehat{\mathbf{v}}_{n_1\to q}^{iT}(t), \ldots, \widehat{\mathbf{v}}_{n_{|\mathcal{N}_q|}\to q}^{iT}(t)]^T, \\
\widetilde{\Gamma}_q^i &= BlkDiag(I_{M_q}, \widehat{\Gamma}_{n_1\to q}^i, \ldots, \widehat{\Gamma}_{n_{|\mathcal{N}_q|}\to q}^i),
\end{aligned} \tag{6.18}$$

with $\{n_1, \ldots, n_{|\mathcal{N}_q|}\} = \mathcal{N}_q$. We have $\widetilde{\mathbf{y}}_q^i(t), \widetilde{\mathbf{v}}_q^i(t) \in \mathbb{R}^{\widetilde{M}_q}$ for each time sample t and $\widetilde{\Gamma}_q^i \in \mathbb{R}^{\widetilde{M}_q \times \widetilde{M}_q}$, where $\widetilde{M}_q = M_q + |\mathcal{N}_q|\cdot Q$. The updating node q can then create its local TRO problem to solve, given as

$$\begin{aligned}
\underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{maximize}} \quad & \frac{\operatorname{tr}(\widetilde{X}_q^T R_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}^i \widetilde{X}_q)}{\operatorname{tr}(\widetilde{X}_q^T R_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}^i \widetilde{X}_q)} \\
\text{subject to} \quad & \widetilde{X}_q^T \widetilde{\Gamma}_q^i \widetilde{X}_q = I_Q,
\end{aligned} \tag{6.19}$$

where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} = \mathbb{E}[\widetilde{\mathbf{y}}^i_q(t)\widetilde{\mathbf{y}}^{iT}_q(t)]$ and $R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} = \mathbb{E}[\widetilde{\mathbf{v}}^i_q(t)\widetilde{\mathbf{v}}^{iT}_q(t)]$.

At this point, the updating node q can continue to follow the steps of the DASF algorithm (see Section 2.5.2) in which case it has to solve its local TRO problem (6.19) using Algorithm 6. More specifically, the change of variable $\widetilde{W}_q = \Sigma^{\frac{1}{2}}U^T\widetilde{X}_q$ is made at node q, where U and Σ are obtained from the singular value decomposition of $\widetilde{\Gamma}^i_q : \widetilde{\Gamma}^i_q = U\Sigma U^T$. Node q then applies Algorithm 6 to obtain $\widetilde{W}^*_q$ after convergence and finally computes $\widetilde{X}^*_q = U\Sigma^{-\frac{1}{2}}\widetilde{W}^*_q$ (see Remark 6.1). However, since Algorithm 6 is itself iterative, using it to solve the local TRO problem at the updating node would be computationally expensive and would create nested iterations inside the outer-loop iterations of the DASF algorithm, generating an algorithm that operates at two different time scales. Note that a similar remark can be made if we were to alternatively compute the EVD in step 1 of Algorithm 6 using a distributed (G)EVD algorithm such as, e.g., [2,87]. In the next subsection, we propose a method that solves (6.13) without nested iterations, but instead *interleaves* the iterations of Algorithm 6 with those of the DASF algorithm and therefore runs at a single time scale.

6.3.3 The distributed TRO algorithm

Solving the local TRO problem (6.19) at each updating node q can become expensive computationally, as it requires applying Algorithm 6 which is an iterative procedure. Instead, we propose the distributed TRO (DTRO) algorithm which only requires applying a single iteration of Algorithm 6 at each updating node.

More specifically, after receiving the compressed data in (6.18) from its neighbors, the updating node q solves its local auxiliary problem at node q similar to (6.6) and (6.8)

$$\underset{\widetilde{X}_q\in\widetilde{\mathcal{S}}^i_q}{\operatorname{argmax}}\ \operatorname{tr}\left(\widetilde{X}^T_q(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}_q\right), \tag{6.20}$$

where $\widetilde{\mathcal{S}}^i_q \triangleq \{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q\times Q} : \widetilde{X}^T_q\widetilde{\Gamma}^i_q\widetilde{X}_q = I_Q\}$. The parameter ρ^i can be locally computed using node q's own observations and the ones received from other nodes

$$\rho^i = r(X^i) = \frac{\mathbb{E}\left[\|X^{iT}\mathbf{v}(t)\|^2\right]}{\mathbb{E}\left[\|X^{iT}\mathbf{y}(t)\|^2\right]} = \frac{\mathbb{E}\left[\|X^{iT}_q\mathbf{v}_q(t) + \sum_{n\in\mathcal{N}_q}\widehat{\mathbf{v}}^i_{n\to q}(t)\|^2\right]}{\mathbb{E}\left[\|X^{iT}_q\mathbf{y}_q(t) + \sum_{n\in\mathcal{N}_q}\widehat{\mathbf{y}}^i_{n\to q}(t)\|^2\right]}, \tag{6.21}$$

such that each node is able to evaluate the network-wide objective from the compressed data only.

Due to the constraint set $\widetilde{\mathcal{S}}_q^i$, (6.20) is a GEVD problem (as opposed to an EVD problem at the equivalent stage of Algorithm 6) for the matrix pair $(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, \widetilde{\Gamma}^i_q)$. At iteration i, node q is therefore solving for the variable $\widetilde{X}_q$ the following GEVD problem

$$(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}_q = \widetilde{\Gamma}^i_q \widetilde{X}_q \widetilde{\Lambda}_q, \tag{6.22}$$

where $\widetilde{\Lambda}_q$ is a $Q \times Q$ diagonal matrix containing the GEVLs of the matrix pair $(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, \widetilde{\Gamma}^i_q)$. The solution at iteration $\widetilde{X}^*_q$ obtained at the updating node q is then defined to be the solution of the GEVD problem (6.22)

$$\widetilde{X}^*_q \triangleq \mathrm{GEVC}_Q(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, \widetilde{\Gamma}^i_q), \tag{6.23}$$

where $V = \mathrm{GEVC}_Q(A, B)$ is defined in Definition 1.

Remark 6.2. Obtaining $\widetilde{X}^*_q$ as in (6.23) corresponds to applying a single iteration of Algorithm 6 at the updating node q. Indeed, referring to Remark 6.1, we note that the change of variable $\widetilde{W}_q = \Sigma^{\frac{1}{2}} U^T \widetilde{X}_q$, where U and Σ are obtained from the singular value decomposition of $\widetilde{\Gamma}^i_q : \widetilde{\Gamma}^i_q = U\Sigma U^T$, is not necessary here as it would lead to (6.20) becoming an EVD (instead of a GEVD) problem. Its solution would then be

$$\widetilde{W}^*_q = \mathrm{EVC}_Q\left(\Sigma^{-\frac{1}{2}} U^T (R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}) U \Sigma^{-\frac{1}{2}}\right), \tag{6.24}$$

satisfying

$$\Sigma^{-\frac{1}{2}} U^T (R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}) U \Sigma^{-\frac{1}{2}} \widetilde{W}^*_q = \widetilde{W}^*_q \widetilde{\Lambda}_q. \tag{6.25}$$

Rewriting the previous equation results in

$$(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}^*_q = \widetilde{\Gamma}^i_q \widetilde{X}^*_q \widetilde{\Lambda}_q \tag{6.26}$$

for $\widetilde{X}^*_q = U\Sigma^{-\frac{1}{2}}\widetilde{W}^*_q$ as in (6.23).

Remark 6.3. We assume in the remaining of this chapter that both matrices in the matrix pair $(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, \widetilde{\Gamma}^i_q)$ are full rank and the largest $Q+1$ GEVLs of the pair are all distinct, so that the solution $\widetilde{X}_q$ in (6.22) is well-defined (see Section 3.4.3 and Appendix B.3). In contrived cases where this assumption does not hold, some technical modifications to the algorithm

are necessary to make the problem well-defined. For the sake of an easy exposition, we make abstraction of this problem for the time being and refer the reader to Section 6.C for precisions.

Since the minimization (6.20) has multiple solutions, an ambiguity exists on the choice of the local variable, namely on the sign of each column of $\widetilde{X}_q^*$. We will then use the following rule to select a solution

$$\widetilde{X}_q^* \leftarrow \widetilde{X}_q^* D^{i+1},\ D^{i+1} = \underset{D\in\mathcal{D}}{\operatorname{argmin}} \|\widetilde{X}_q^* D - \widetilde{X}_q^i\|_F, \tag{6.27}$$

where $\mathcal{D}$ is the set of $Q \times Q$ signature matrix, i.e., diagonal matrices containing either 1 or -1 on their diagonal and

$$\widetilde{X}_q^i = [X_q^{iT}, I_Q, \ldots, I_Q]^T \in \mathbb{R}^{\widetilde{M}_q \times Q}. \tag{6.28}$$

This corresponds to minimizing the distance $\|\widetilde{X}_q^* - \widetilde{X}_q^i\|_F$ over the set of possible solutions of (6.20), as in the DASF algorithm.

The final step of the DTRO algorithm at iteration i is to communicate the updates to the other nodes. Let us partition $\widetilde{X}_q^*$ as

$$\widetilde{X}_q^* = [X_q^{(i+1)T}, G_{n_1}^{(i+1)T}, \ldots, G_{n_{|\mathcal{N}_k|}}^{(i+1)T}]^T, \tag{6.29}$$

with $n_k \in \mathcal{N}_q$ and where X_q is $M_q \times Q$ and G_n's are $Q \times Q$. Finally, G_n^{i+1}'s are disseminated in the full branch $\mathcal{B}_{nq}$ of the network, and each node updates its local variable as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q, \\ X_k^i G_n^{i+1} & \text{if } k \neq q, \text{ with } k \in \mathcal{B}_{nq}. \end{cases} \tag{6.30}$$

This process is then repeated by selecting a new updating node, as shown in Algorithm 7, which summarizes the steps of the DTRO algorithm.

6.3.4 Convergence of the DTRO algorithm

From Algorithm 7, we note that if we were to remove the updating of ρ (step 3) in the DTRO algorithm, we would obtain an instance of the distributed adaptive covariance matrix generalized eigenvector estimation (DACGEE) algorithm from [2] which is a special case of the DASF algorithm applied to the GEVD problem for the matrix pair $(R_{\mathbf{yy}} - \rho R_{\mathbf{vv}}, I_M)$ for an arbitrary (fixed) value ρ. This demonstrates that the DTRO algorithm interleaves iterations of Algorithm 6

Algorithm 7: Distributed Trace Ratio Optimization (DTRO) algorithm

X^0 initialized randomly, $i \leftarrow 0$

repeat

$q \leftarrow (i \mod K) + 1$

1) The network $\mathcal{G}$ is pruned to obtain a tree $\mathcal{T}^i(\mathcal{G}, q)$

2) Node q receives $\widehat{\Gamma}^i_{n\to q}$ as defined in (6.17) and $\widehat{\mathbf{y}}^i_{n\to q}(t)$, $\widehat{\mathbf{v}}^i_{n\to q}(t)$ for $t = iN, \ldots, iN + N - 1$ from all $n \in \mathcal{N}_q$ as in (6.16) to construct $\widetilde{\Gamma}^i_q$, $\widetilde{\mathbf{y}}^i_q(t)$ and $\widetilde{\mathbf{v}}^i_q(t)$ as defined in (6.18)

3) Node q estimates $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}$ from the samples of $\widetilde{\mathbf{y}}^i_q(t)$ and $\widetilde{\mathbf{v}}^i_q(t)$ defined in (6.18)

4) ρ^i is updated using (6.21)

5) $\widetilde{X}^*_q \leftarrow \mathrm{GEVC}_Q(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, \widetilde{\Gamma}^i_q)$, where $\widetilde{\Gamma}^i_q$ is defined as in (6.18)

6) $\widetilde{X}^*_q \leftarrow \widetilde{X}^*_q D^{i+1}$, where D^{i+1} is given in (6.27)

7) Partition $\widetilde{X}^*_q$ as in (6.29), disseminate G^{i+1}_n in $\mathcal{B}_{nq}, \forall n \in \mathcal{N}_q$

8) Every node updates X^{i+1}_k according to (6.30)

$i \leftarrow i + 1$

with the iterations of a distributed GEVD algorithm. However, we cannot rely on the convergence of Algorithm 6 to justify convergence of the DTRO algorithm because in each iteration, the latter solves partially, and not fully, the network-wide GEVD problem. On the other hand, the convergence of the DACGEE algorithm in [2] or the DASF algorithm does not imply convergence of the DTRO algorithm either, as ρ changes at each iteration, changing the eigenvalue problem to solve at each iteration.

Nevertheless, it can be shown that the DTRO algorithm converges. The following result is analogous to Theorem 3.1 where we obtain that the DTRO algorithm converges in objective values.

Theorem 6.2. Let $(X^i)_i$ be the sequence of iterates satisfying Algorithm 7's update rule. Then, all elements of $(X^i)_{i>0}$ belong to the constraint set $\mathcal{S}$ and $\big(r(X^i)\big)_{i>0} = (\rho^i)_{i>0}$ is a converging sequence that is monotonically increasing.

Proof. See Section 6.A. □

Additionally, it can be shown that the fixed points of the DTRO algorithm are stationary points of the DTRO problem (6.13), similar to Theorem 3.2.

Theorem 6.3. A fixed point $\overline{X}$ of Algorithm 7 contains eigenvalues of $R_{\mathbf{vv}} - r(\overline{X})R_{\mathbf{yy}}$ in its columns, and is a stationary point of the TRO problem (6.13).

Proof. See Section 6.B. □

Convergence of the sequence $(X^i)_i$ to a single point can then be established using a similar result to Theorem 3.6. Additionally, since the TRO problem does not have any local maxima [144], convergence of $(X^i)_i$ to an optimal X^* solving the TRO problem (2.60) can be shown using Corollary 3.3.

6.4 Communication and computational aspects

Although it is difficult to generalize communication and especially computational costs of the DASF framework as the number of parameters can vary significantly between problems, we provide a detailed discussion for the specific case of the TRO problem, comparing DTRO, DASF and a centralized solver implementing Algorithm 6.

6.4.1 Communication costs

Based on the description of the DTRO algorithm in Section 6.3, the data and parameters that are being communicated between two neighboring nodes at iteration i are the N samples of $Q-$dimensional compressed signals $\widehat{\mathbf{y}}_k^i$ and $\widehat{\mathbf{v}}_k^i$ and the $Q \times Q$ matrices $X_k^{iT} X_k^i$ and G_k^i, which implies the communication cost is in $\mathcal{O}(NQ + 2Q^2)$ per transmission link and per iteration, where $\mathcal{O}$ denotes the Landau notation. This result is the same for any given link in the network, whether the latter is fully-connected or not. In general $N \gg Q$, which makes the first term NQ the dominant one. In comparison, in a centralized setting the N samples of uncompressed signals of dimension M_k, are being sent from each node k towards the fusion center leading to a communication cost of $\mathcal{O}(NM_k)$ for node k where typically $M_k > Q$. The compression ratio at node k is therefore roughly M_k/Q. An updating scheme that reduces the communication overhead is also presented in [105] allowing to decrease the

number of parameters being sent from the updating node to the rest of the network. Note that the communication cost of solving the TRO problem using the DASF algorithm is the same as the DTRO algorithm.

6.4.2 Computational complexity

At each iteration i of Algorithm 7, the updating node q needs to compute the two sample covariance matrices $R^i_{\tilde{\mathbf{y}}_q\tilde{\mathbf{y}}_q}$ and $R^i_{\tilde{\mathbf{v}}_q\tilde{\mathbf{v}}_q}$ which requires around $\widetilde{M}_q^2 N$ operations each, while the GEVC computation has a complexity of $\mathcal{O}(\widetilde{M}_q^3)$. Resolving the sign ambiguity can be done by comparing the norm of two M_q-dimensional vectors, which is done Q times, once for each column of X^i_q, and therefore requires around $M_q Q$ operations, which, considering a large enough sample size N is negligible compared to the two previous computations. Node q's computations therefore have a complexity of $\mathcal{O}(\widetilde{M}_q^2(N+\widetilde{M}_q))$. On the other hand, at non-updating nodes $k \neq q$, $2M_kQN$ and M_kQ^2 operations are required to compress the local signals and to compute the expressions $X_k^{iT}X_k^i$, respectively. These nodes then update the estimation of their local filters, which is done in $\mathcal{O}(M_kQ^2)$ operations. The computations at nodes $k \neq q$ have therefore a complexity of $\mathcal{O}(M_kQN)$, again considering a large N. In comparison, the adaptive (sliding window) version of the centralized TRO algorithm (Algorithm 6) would have a complexity of $\mathcal{O}(M^2(N+M))$ per window. For example, in a fully-connected network with $K = 30$ nodes, $Q = 1$, $N = 1000$ and $M_k = 15\ \forall k$, the centralized adaptive algorithm requires $\sim 294 \times 10^6$ operations per window of N samples (assuming a single iteration per window), whereas the DTRO algorithm requires $\sim 2.02 \times 10^6$ operations per window. Solving the TRO problem in a distributed setting using the DASF algorithm would multiply this latter number by a constant factor, due to solving multiple local auxiliary problems at each updating node.

It is important to note that in the case of non-fully-connected networks, the $\widetilde{M}_k$'s depend on the number of neighbors a node has and not the total number of nodes in the network. Therefore, the non-fully-connected topologies scale better compared to fully-connected ones. To follow up on the previous example, if the average number of neighbors per node is equal to 3, the distributed algorithm would require only $\sim 0.33 \times 10^6$ operations per window of N samples. However, as will be demonstrated in the next section, the DTRO algorithm converges the fastest in fully-connected networks in general, as was the case for the DASF algorithm. Hence in practice, there is a tradeoff between computational complexity and convergence speed.

6.5 Simulations and discussions

In this section, we provide experimental results on the DTRO algorithm and demonstrate its convergence. We will first consider a stationary setting, where we will also compare the computational cost of the DTRO and the implementation of the DASF algorithm solving the TRO problem. We will then provide experimental results of the DTRO algorithm in an adaptive setting. Throughout these experiments, we consider randomly generated networks to apply the DTRO and DASF algorithms on, where the network is generated using the Erdős-Rényi model with connection probability 0.8, and where the pruning function $\mathcal{T}^i(\cdot, q)$ is the shortest path pruning. Additionally, the number of channels M_k of the signals $\mathbf{y}_k$ and $\mathbf{v}_k$ measured at each node k are equal and given by M/K, where we take $K = 10$ and $M = 50$, therefore $M_k = 5$ for every node k. The number of columns of X is fixed at 2, i.e., $Q = 2$.

6.5.1 Stationary setting

In this first experiment, we consider that the $\mathbf{y}$ and $\mathbf{v}$ are stationary, following a mixture model:

$$\mathbf{y}(t) = \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{6.31}$$

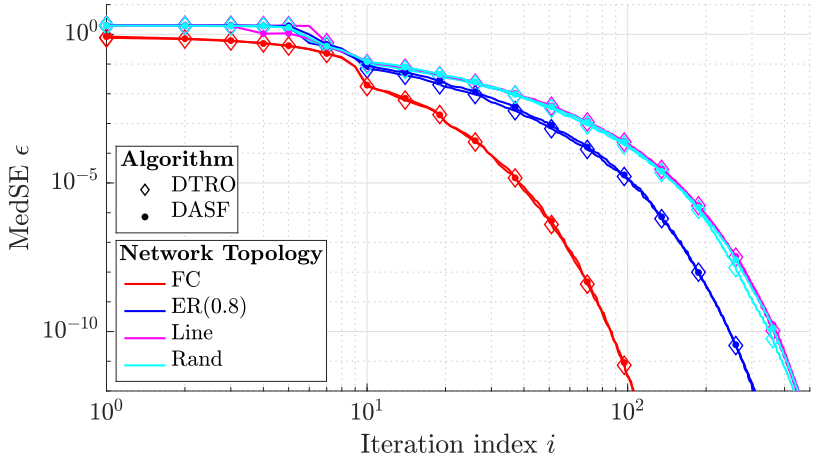
$$\begin{aligned}\mathbf{v}(t) &= \Pi_r \cdot \mathbf{r}(t) + \mathbf{y}(t) \\ &= \Pi_r \cdot \mathbf{r}(t) + \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t),\end{aligned} \tag{6.32}$$

with $\mathbf{r}(t)$, $\mathbf{s}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.5)$, $\mathbf{n}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.1)$ for every entry and time instant t. Π_s and Π_r are mixture matrices independent of time with each entry independently drawn from $\mathcal{N}(0, 0.1)$. At each iteration, every node measures $N = 10^4$ samples of its local signals to estimate the covariance matrices of $\widehat{\mathbf{y}}_k^i$ and $\widehat{\mathbf{v}}_k^i$. As in previous chapters, we will assess the convergence of the DTRO algorithm using the MedSE ϵ:

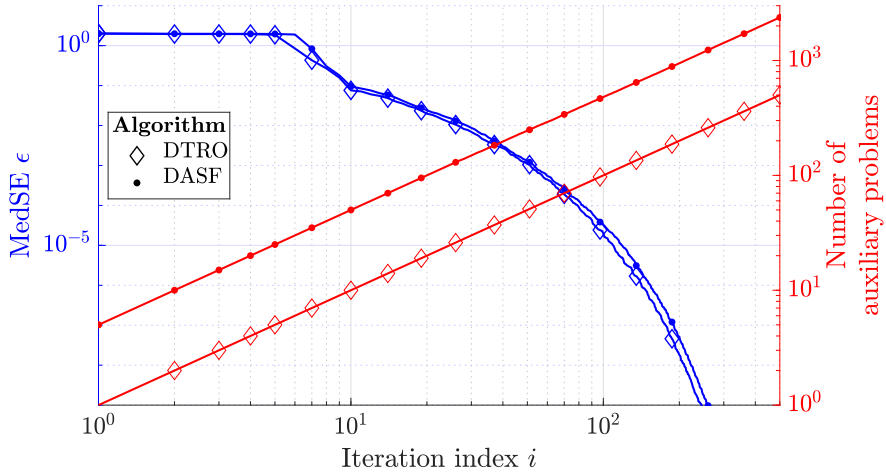
$$\epsilon(i) = \text{median}\left(\frac{\|X^i - X^*\|_F^2}{\|X^*\|_F^2}\right), \tag{6.33}$$

taken over 100 Monte Carlo runs. X^* is an optimal solution of the TRO problem (6.13), computed using a centralized implementation of Algorithm 6. Since the TRO problem has multiple solutions, we select X^* a posteriori as the one that is closest to X^i in the final iteration of the DTRO algorithm.

At each iteration i, the updating node q solves its local auxiliary problem (6.20) in the DTRO algorithm. We will compare the computational cost of the DTRO



(a) Convergence comparison between the DTRO algorithm and the DASF algorithm for various network topologies when solving Problem (6.13). The descriptions of the different network topologies are provided in Table 2.3.



(b) Convergence and cumulative computational cost comparison between the DTRO algorithm and the DASF algorithm when solving Problem (6.13) on a network generated using the Erdős-Rényi model with a connection probability of 0.8.

Figure 6.1: Convergence comparison between DTRO and DASF in a stationary setting.

algorithm to the one of the DASF algorithm solving the TRO problem. For the

DASF algorithm, the updating node q has to solve a local TRO problem as in (6.19) using Algorithm 6, i.e., it solves (6.20) multiple times at each updating node. In this experiment, the stopping criterion of Algorithm 6 has been set to be a threshold of 10^{-8} on the norm of the difference of two consecutive $\widetilde{X}_q$'s and a maximum number of iterations of 10. Comparisons of the DTRO and DASF algorithms are provided in Figures 6.1a and 6.1b, where we observe similar MedSE values per iteration for both methods, while the DASF algorithm requires to solve an average value (over iterations of median values) of 4.74 times more auxiliary problems than the DTRO method. Additionally, despite the reduced number of computations at each node, we see that the convergence rates of the DTRO and the DASF algorithm are equivalent.

6.5.2 Adaptive Setting

In this experiment, we consider the same settings as in Section 6.5.1, however the signal models are now given by

$$\mathbf{y}(t) = \Pi_s(t) \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{6.34}$$

$$\mathbf{v}(t) = \Pi_r(t) \cdot \mathbf{r}(t) + \Pi_s(t) \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{6.35}$$

i.e., the mixture matrices Π_s and Π_r are now time-dependent. In particular, we have $\Pi_s(t) = \Pi_{s,0} \cdot (1 - \theta(t)) + (\Pi_{s,0} + \Delta_s) \cdot \theta(t)$, where θ is given in Figure 6.2 and the elements of $\Pi_{s,0}$ and Δ are independently drawn from $\mathcal{N}(0, 0.1)$ and $\mathcal{N}(0, 10^{-4})$ respectively, while $\Pi_r(t)$ is defined similarly. Therefore, the signals $\mathbf{y}$ and $\mathbf{v}$ are not stationary anymore, which implies that X^* is time-dependent. In particular, at each iteration i, we estimate the solution X^{*i} at iteration i using $N = 10^3$ time samples of $\mathbf{y}$ and $\mathbf{v}$ by solving

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{maximize}} \quad & \frac{\operatorname{tr}(X^T \widehat{R}^i_{\mathbf{vv}} X)}{\operatorname{tr}(X^T \widehat{R}^i_{\mathbf{yy}} X)} \\ \text{subject to} \quad & X^T X = I_Q, \end{aligned} \tag{6.36}$$

using Algorithm 6, where

$$\widehat{R}^i_{\mathbf{vv}} = \frac{1}{N} \sum_{\tau=iN}^{iN+N-1} \mathbf{v}(\tau)\mathbf{v}^T(\tau), \tag{6.37}$$

$$\widehat{R}^i_{\mathbf{yy}} = \frac{1}{N} \sum_{\tau=iN}^{iN+N-1} \mathbf{y}(\tau)\mathbf{y}^T(\tau). \tag{6.38}$$

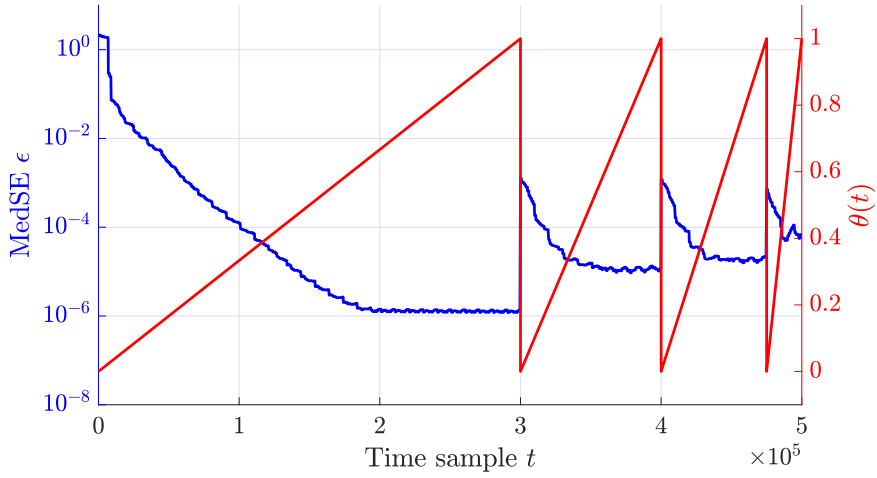


Figure 6.2: MedSE ϵ over time of the DTRO algorithm in an adaptive setting. The relationship between the time t and the iterations i is given by $i = \lfloor t/N \rfloor$.

The MedSE ϵ is then given by

$$\epsilon(i) = \text{median}\left(\frac{\|X^i - X^{*i}\|_F^2}{\|X^{*i}\|_F^2}\right), \tag{6.39}$$

with $i = \lfloor t/N \rfloor$, and where the median is taken over 100 Monte Carlo runs. Figure 6.2 shows the MedSE value over time, where we see that the DTRO algorithm is able to track slow changes in the signal statistics of $\mathbf{y}$ and $\mathbf{v}$ and can adapt to abrupt changes as well, which is characterized by an initial jump in the MedSE value that gradually decreases. Note that the MedSE values settle around a certain threshold due to the fact that X^* is time-dependent, where we have a higher MedSE settling value for faster rates of change (i.e., steeper slope of θ) in the signal statistics.

6.6 Conclusion

In this chapter, we have studied the TRO problem and presented a distributed adaptive algorithm for solving the trace ratio optimization problem in a distributed WSN setting, by solving local GEVD problems at the nodes. The proposed DTRO algorithm is a special case of the DASF algorithm (for the TRO problem) where the local problems are only partially solved at each

updating node. Nevertheless, we have shown that the DTRO algorithm still converges to an optimal solution of the TRO problem. We have also analyzed the communication and processing cost of the centralized, the DASF, and the DTRO algorithms, and concluded that the DTRO algorithm has a significantly smaller communication and processing burden. Simulation results allowed us to validate our convergence and computational efficiency claims. In the next chapter, we will generalize the technique used in the DTRO algorithm to other fractional programs.

Appendices

6.A Convergence of the objective

This section provides a proof of Theorem 6.2.

To prove the statement that any X^i obtained from Algorithm 7 belongs to $\mathcal{S}$, we can use the same steps as in Lemma 2.1 from Chapter 2.

For the second statement, let us re-introduce the matrix C_q^i defined in (2.50) such that $X^i = C_q^i \widetilde{X}_q^i$, $\widetilde{\mathbf{y}}_q^i(t) = C_q^{iT}\mathbf{y}(t)$, $\widetilde{\mathbf{v}}_q^i(t) = C_q^{iT}\mathbf{v}(t)$ and $\widetilde{\Gamma}_q^i = C_q^{iT}C_q^i$ (see Remark 2.5). Additionally, $\widetilde{X}_q^i = [X_q^{iT}, I_Q, \ldots, I_Q]^T$ is defined as in (6.28). Then, at the end of iteration $i > 0$, we have

$$f(X^i, \rho^i) \triangleq \operatorname{tr}\left(X^{iT}(R_{\mathbf{vv}} - \rho^i R_{\mathbf{yy}})X^i\right) \tag{6.40}$$

$$= \operatorname{tr}\left(\widetilde{X}_q^{iT}(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}_q^i\right) \tag{6.41}$$

$$= 0, \tag{6.42}$$

where the second equation follows from $X^i = C_q^i\widetilde{X}_q^i$. The last equation is obtained by observing that $\rho^i = r(X^i) = r(C_q^i\widetilde{X}_q^i)$, which after plugging it in (6.41) yields (6.42). On the other hand, we can also write

$$f(X^{i+1}, \rho^i) \triangleq \operatorname{tr}\left(X^{(i+1)T}(R_{\mathbf{vv}} - \rho^i R_{\mathbf{yy}})X^{i+1}\right) \tag{6.43}$$

$$= \operatorname{tr}\left(\widetilde{X}_q^{*T}(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}_q^*\right). \tag{6.44}$$

Since $\widetilde{X}_q^*$ maximizes $\operatorname{tr}\left(\widetilde{X}_q^T(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}_q\right)$ under the constraint $\widetilde{X}_q^T\widetilde{\Gamma}_q^i\widetilde{X}_q = I_Q$ (or equivalently $\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i$) as given in (6.20)-(6.22), we have

$$f(X^{i+1}, \rho^i) \geq \operatorname{tr}\left(\widetilde{X}_q^T(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}_q\right), \tag{6.45}$$

$$\forall \widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i.$$

In particular, taking $\widetilde{X}_q = [X_q^{iT}, I_Q, \ldots, I_Q]^T$, we have $C_q^i\widetilde{X}_q = X^i$. Plugging this choice in (6.45), we obtain

$$f(X^{i+1}, \rho^i) \geq \operatorname{tr}\left(X^{iT}(R_{\mathbf{vv}} - \rho^i R_{\mathbf{yy}})X^i\right) \tag{6.46}$$

$$= f(X^i, \rho^i) = 0, \tag{6.47}$$

where the last equation follows from (6.42). Combining (6.44) with (6.46)-(6.47) results in

$$f(X^{i+1},\rho^i)=\operatorname{tr}\left(\widetilde{X}_q^{*T}(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}-\rho^i R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q})\widetilde{X}_q^*\right)\geq 0, \tag{6.48}$$

which finally results in

$$\frac{\operatorname{tr}\left(\widetilde{X}_q^{*T}R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}\widetilde{X}_q^*\right)}{\operatorname{tr}\left(\widetilde{X}_q^{*T}R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q^*\right)}\geq\rho^i, \tag{6.49}$$

where it is noted that the denominator is strictly larger than zero due to the fact that $R_{\mathbf{yy}}$ (and consequently any $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$) is positive definite. Note that the left-hand side of (6.49) is equal to ρ^{i+1} by definition, which implies that $\rho^{i+1}\geq\rho^i$, which proves that the sequence $(\rho^i)_{i>0}$ is monotonic non-decreasing. By monotonicity, and since the sequence $(\rho^i)_{i>0}$ has an upper bound ρ^* it converges. □

6.B Stationarity of fixed points

This section provides a proof for Theorem 6.3.

In this proof, we reintroduce the matrix C_q^i defined in (2.50) such that $\widetilde{\mathbf{y}}_q^i(t)=C_q^{iT}\mathbf{y}(t)$, $\widetilde{\mathbf{v}}_q^i(t)=C_q^{iT}\mathbf{v}(t)$, $X^{i+1}=C_q^i\widetilde{X}_q^*$ and $\widetilde{\Gamma}_q^i=C_q^{iT}C_q^i$ (see Remark 2.5). We assume that at iteration i, $\overline{X}=X^i$ is a fixed point of the DTRO algorithm, i.e., $X^{i+1}=X^i=\overline{X}$ for any updating node is q, also implying $\rho^{i+1}=\rho^i=\bar{\rho}=r(\overline{X})$. Additionally, after reaching a fixed point, the GEVLs do not change over iterations, therefore $\widetilde{\Lambda}_q^{i+1}=\widetilde{\Lambda}_q^i=\overline{\Lambda}_q$. Replacing these in (6.22), we obtain

$$C_q^{iT}\left(R_{\mathbf{vv}}-\bar{\rho}R_{\mathbf{yy}}\right)\overline{X}=C_q^{iT}\overline{X}\overline{\Lambda}_q. \tag{6.50}$$

Note that, from the definition of C_q^i, the first M_q rows of $C_q^{iT}(R_{\mathbf{vv}}-\bar{\rho}R_{\mathbf{yy}})\overline{X}$ correspond to the first M_q rows of $(R_{\mathbf{vv}}-\bar{\rho}R_{\mathbf{yy}})\overline{X}$ such that

$$\left[(R_{\mathbf{vv}}-\bar{\rho}R_{\mathbf{yy}})\overline{X}\right]_{1:M_q,\cdot}=\left[\overline{X}\overline{\Lambda}_q\right]_{1:M_q,\cdot} \tag{6.51}$$

$$=\overline{X}_q\overline{\Lambda}_q, \tag{6.52}$$

where $\left[\,\cdot\,\right]_{1:M_q,\cdot}$ extracts the first M_q rows of a matrix and $\overline{X}_q$ corresponds to the $q-$th block of $\overline{X}$ based on the partitioning (6.15). Since we consider a fixed point assumption, we can apply the same reasoning to every node such that

(6.52) will hold for any node q. After stacking all the corresponding equations, we can write

$$(R_{\mathbf{vv}} - \bar{\rho} R_{\mathbf{yy}})\, \overline{X} = \begin{bmatrix} \overline{X}_1 \overline{\Lambda}_1 \\ \vdots \\ \overline{X}_K \overline{\Lambda}_K \end{bmatrix}. \tag{6.53}$$

Left multiplying (6.50) from the left by $\widetilde{X}_q^{iT}$ defined in (6.28) and using the fact that $C_q^i \widetilde{X}_q^i = X^i = \overline{X}$, we obtain

$$\overline{X}^T (R_{\mathbf{vv}} - \bar{\rho} R_{\mathbf{yy}})\, \overline{X} = \overline{X}^T \overline{X} \overline{\Lambda}_q = \overline{\Lambda}_q, \tag{6.54}$$

where the last equation follows from the orthogonality constraint $\overline{X}^T \overline{X} = I_Q$ (as obtained in Theorem 6.2). We observe that the left-hand side of (6.54) does not depend on the node q, therefore $\overline{\Lambda}_q$ is equal for all nodes and so we can set $\overline{\Lambda}_k = \overline{\Lambda}$, $\forall k \in \mathcal{K}$. We can then rewrite equation (6.53) as

$$(R_{\mathbf{vv}} - \bar{\rho} R_{\mathbf{yy}})\, \overline{X} = \overline{X} \overline{\Lambda}. \tag{6.55}$$

Therefore, the columns of $\overline{X}$ can only contain EVCs of $(R_{\mathbf{vv}} - \bar{\rho} R_{\mathbf{yy}})$ when $\overline{X}$ is a fixed point. Since (6.55) defines the stationarity condition of the TRO problem (see Appendix B.4), $\overline{X}$ is a stationary point of (6.13). Note that this result is analogous to the one we have provided for Theorem 3.2 in Section 3.A, where Condition 1a is satisfied for any $X : X^T X = I_Q$ (see Example 3.1).

6.C Modifications for rank-deficiency

Although from a statistical point of view, this would be a rare event, we discuss in this section some possible modifications that can be applied to the methods presented previously for special cases where the assumptions of Remark 6.3 do not hold. Similar discussions and solutions have been presented in [154].

Suppose that there exist iterations at which the matrix $(R^i_{\widetilde{\mathbf{v}}_q \widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q})$ or $\widetilde{\Gamma}^i_q$ is not full rank. This implies that C^i_q is not full rank which in turn means that there is at least a node k for which X^i_k has linearly dependent columns. A way to fix this problem would be to make node k replace all but one of the linearly dependent columns by random vectors.

On the other hand, suppose that the largest $Q+1$ GEVLs of the matrix pair $(R^i_{\widetilde{\mathbf{v}}_q \widetilde{\mathbf{v}}_q} - \rho^i R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q}, \widetilde{\Gamma}^i_q)$ are not all distinct, making the solution of (6.22) ill-defined. Then, one solution would be to skip node q at iteration i, while assuming the problem will not occur again. Otherwise, suppose there are $d < Q+1$ different

GEVLs in the first $Q+1$ GEVLs (when ordered in decreasing order). We are looking for an $\widetilde{X}_q$ such that its range space is in $\mathcal{V}_q^i$, where $\mathcal{V}_q^i$ denotes the space spanned by all GEVCs corresponding to the d largest GEVLs, while making a minimal change from the current X^i to ensure local convergence at node q. If the basis vectors of $\mathcal{V}_q^i$ are put in the columns of the matrix $V_q^i \in \mathbb{R}^{M\times Q}$, then $\widetilde{X}_q^* = V_q^i P_q^*$, where $P_q^* \in \mathbb{R}^{Q\times Q}$ is obtained through solving

$$\begin{aligned} P_q^* &= \underset{P_q}{\operatorname{argmin}} \, \|V_q^i P_q - [X_q^{iT}, I_Q, \ldots, I_Q]^T\|_F \\ &\text{subject to} \quad P_q^T P_q = I_Q, \end{aligned} \tag{6.56}$$

which is known as the Orthogonal Procrustes Problem [155].

7 | Distributed adaptive signal fusion for fractional programs

This chapter is largely based on **C. A. Musluoglu** and A. Bertrand, "Distributed Adaptive Signal Fusion for Fractional Programs," Submitted for review, pp. 1-13, 2023, and **C. A. Musluoglu** and A. Bertrand, "A computationally efficient algorithm for distributed adaptive signal fusion based on fractional programs," in Proceedings of the *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1-5, 2023.

ABSTRACT | The DASF algorithm requires each node to sequentially build a compressed version of the original network-wide problem and solve it locally. However, these local problems can still result in a high computational load at the nodes, especially when the required solver is iterative. In Chapter 6, we saw a specific problem — the TRO problem — where this is the case and proposed the DTRO algorithm as a computationally more efficient alternative to the DASF method. In this chapter, we study the case of general fractional programs, i.e., problems for which the objective function is a ratio of two continuous functions, which indeed require such iterative solvers. By exploiting the structure of a commonly used method for solving fractional programs and interleaving it with the iterations of the standard DASF algorithm, we obtain a distributed algorithm with a significantly reduced computational cost compared to the straightforward application of DASF as a meta-algorithm. We prove convergence and optimality of this "fractional DASF" (F-DASF) algorithm and demonstrate its performance via numerical simulations.

7.1 Introduction

In this chapter, we focus in particular on so-called fractional programs, i.e., problems where the function ϱ can be written as a fraction of two continuous functions, i.e., $\varrho(X^T\mathbf{y}(t)) = \varphi_1(X^T\mathbf{y}(t))/\varphi_2(X^T\mathbf{y}(t))$. Several spatial filtering problems have such an objective, including signal-to-noise ratio (SNR) maximization filters [9], trace ratio optimization [94], regularized total least squares [99, 100], among others. These problems fit the DSFO problem family and can therefore be solved using the DASF framework. As seen in Chapter 2, the main idea behind the DASF method is to iteratively create and solve a compressed version of the global problem at each node by transmitting only compressed data within the network. We observed that a convenient property of the DASF "meta" algorithm is that the local problems to be solved at each node are compressed instances of the original (centralized) problem. The DASF algorithm therefore merely requires to "copy-paste" this solver at each individual node [110].

However, solving the compressed problem can be computationally expensive when an expensive iterative solver is required. This is indeed also the case for fractional programs, which are commonly solved using an iterative algorithm referred to as Dinkelbach's procedure. In this chapter, we propose the fractional DASF (F-DASF) algorithm which implements a single iteration of the Dinkelbach procedure within each iteration of the DASF algorithm, therefore interleaving the iterations of both algorithms and avoiding nested iterative loops. This however implies that F-DASF cannot rely on the convergence guarantees and proofs of the original DASF algorithm, as these assume that the nodes fully execute Dinkelbach's procedure until convergence within each iteration of DASF. Therefore, the main result of this chapter will be to show that the simplified F-DASF algorithm still converges to the solution of the global problem, under similar conditions as the original DASF algorithm. In our results, we will see that on top of significantly reducing the computational cost and simplifying the overall method, the F-DASF algorithm has comparable convergence rates to the DASF algorithm.

The resulting F-DASF algorithm can in fact be viewed as a generalization of the DTRO algorithm introduced in Chapter 6, which is a particular fractional program where both φ_1 and φ_2 consist of only a single quadratic term. Our proposed F-DASF algorithm can be applied to the entire class of fractional programs. Furthermore, it extends the "vanilla" DASF framework towards a more efficient class of algorithms for the particular case of fractional programs.

The outline of this chapter is as follows. In Section 7.2, we review the basics of fractional programs and Dinkelbach's procedure to solve them. In Section 7.3

we formally define the distributed problem setting and the assumptions used throughout this chapter. The proposed F-DASF algorithm is provided in Section 7.4 while its convergence proof is given in Section 7.5. Finally, Section 7.6 shows the performance of the algorithm and its comparison with the state-of-the-art.

7.2 Fractional programming review

A fractional program is an optimization problem with an objective function r represented by a ratio of two continuous and real-valued functions f_1 and f_2:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & r(X) \triangleq \frac{f_1(X)}{f_2(X)} \\ \text{subject to} \quad & X \in \mathcal{S}, \end{aligned} \tag{7.1}$$

where $\mathcal{S} \subset \mathbb{R}^{M \times Q}$ is a non-empty constraint set and $f_2(X) > 0$ for $X \in \mathcal{S}$. We define the minimal value of r over $\mathcal{S}$ as $\rho^* \triangleq \min_{X \in \mathcal{S}} r(X)$ and the set of arguments achieving this value as $\mathcal{X}^* \triangleq \{X \in \mathcal{S} \mid r(X) = \rho^*\}$. We denote by $X^* \in \mathcal{X}^*$ one particular solution of (7.1). To solve fractional programs, there exist two prominent methods based on solving auxiliary problems instead of the original problem (7.1), for which an overview can be found in [156]. The first method consists of creating an equivalent problem by defining new optimization variables using what is referred to as the Charnes-Cooper transform. It initially was presented in [157] for linear f_1 and f_2 with $\mathcal{S}$ a convex polyhedron, and was then extended to cases where f_1 and f_2 are convex [156, 158].

The second method is a parametric approach and is the one we will focus on in this chapter. Initially presented in [159], it is often referred to as Dinkelbach's procedure. Let us define the auxiliary functions $f : \mathcal{S} \times \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$:

$$f(X, \rho) \triangleq f_1(X) - \rho f_2(X), \tag{7.2}$$

$$g(\rho) \triangleq \min_{X \in \mathcal{S}} f(X, \rho), \tag{7.3}$$

where ρ is a real-valued scalar. It can be shown that g is continuous, strictly decreasing with ρ, and has a unique root. We also observe that for any fixed X, the function f in (7.2) is affine, and therefore g is concave. Note that these results do not require convexity assumptions on f_1 or f_2. It has been shown [159–163] that there is a direct relationship between the roots of g and the solution of (7.1), given in the following lemma.

Theorem 7.1 (see [162]). Suppose $\mathcal{S}$ is compact. Then, if for a scalar ρ we have $g(\rho) = 0$ then $\rho = \rho^*$, where ρ^* is the global minimum of r.

It is noted that it is possible to obtain weaker relationships between the roots of g and (7.1) if $\mathcal{S}$ is not compact [162,163], but this is beyond the scope of this review, i.e., we will assume that $\mathcal{S}$ is compact in the remaining of this chapter, unless mentioned otherwise.

Dinkelbach's procedure is an iterative method aiming to find the unique root ρ^* of g. We start by initializing X^0 randomly and set $\rho^0 = r(X^0)$. Then, at each iteration i, we find the solution set of the auxiliary problem minimizing f given ρ:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X, \rho^i) = f_1(X) - \rho^i f_2(X) \\ \text{subject to} \quad & X \in \mathcal{S}. \end{aligned} \tag{7.4}$$

The solution of (7.4) might not be unique but given by a set $\mathcal{X}^{i+1}$, in which case one $X^{i+1} \in \mathcal{X}^{i+1}$ is selected. Finally, the function values are updated as

$$\rho^{i+1} = r(X^{i+1}). \tag{7.5}$$

The steps of Dinkelbach's procedure are summarized in Algorithm 8, while convergence results are summarized in the following theorem, which combines results from [162–164].

Theorem 7.2. Assuming $\mathcal{S}$ is compact, the sequence $(\rho^i)_i$ converges to the finite minimal value ρ^* of (7.1). Additionally, convergent subsequences of $(X^i)_i$ converge to an optimal solution of (7.1). If Problem (7.1) has a unique solution X^*, then $(X^i)_i$ converges to X^*.

If the constraint set $\mathcal{S}$ is not compact, we can still obtain convergence of the sequence $(\rho^i)_i$ to ρ^* if Problem (7.1) has a solution, the solution sets of the auxiliary problems (7.4) are not empty and $\sup_i f_2(X^i)$ is finite [162], with sup denoting the supremum.

It is noted that Dinkelbach's procedure can be viewed as an instance of Newton's root-finding method, as we can show that [163]

$$\rho^{i+1} = \rho^i - \frac{g(\rho^i)}{-f_2(X^{i+1})}, \tag{7.6}$$

Algorithm 8: Dinkelbach's procedure [159]

X^0 initialized randomly in $\mathcal{S}$, $\rho^0 \leftarrow r(X^0)$, $i \leftarrow 0$
repeat
1) $X^{i+1} \leftarrow \underset{X \in \mathcal{S}}{\operatorname{argmin}}\ f(X, \rho^i)$
2) $\rho^{i+1} \leftarrow r(X^{i+1})$ where r is defined in (7.1)
$i \leftarrow i + 1$

which corresponds to Newton's root finding method applied to the function g, since $-f_2(X^{i+1})$ is a subgradient of g at ρ^i. Other algorithms using a different root finding method have also been described to solve fractional programs, for example the bisection method [100,156]. It is however indicated in [165] that the Newton root finding procedure has faster convergence compared to those in the context of fractional programming. Additionally, when the fractional (7.1) corresponds to the TRO problem, the Dinkelbach procedure corresponds to Algorithm 6 presented in Chapter 6.

7.3 Problem setting and preliminaries

In this section, we provide the details of the problem setting, which in various parts is similar to the settings of previous chapters. However, to give an unambiguous description of the novel method presented in this chapter, some definitions and clarifications are repeated. We consider a WSN represented by a graph $\mathcal{G}$ consisting of K nodes where the set of nodes is denoted as $\mathcal{K} = \{1, \ldots, K\}$. Each node k measures a stochastic M_k-channel signal $\mathbf{y}_k$ considered to be (short-term) stationary and ergodic. The observation of $\mathbf{y}_k$ at time t is denoted as $\mathbf{y}_k(t) \in \mathbb{R}^{M_k}$. We define the network-wide signal to be

$$\mathbf{y} = [\mathbf{y}_1^T, \ldots, \mathbf{y}_K^T]^T, \tag{7.7}$$

such that the time sample $\mathbf{y}(t)$ at time t is the $M-$dimensional vector obtained by stacking every $\mathbf{y}_k(t)$, where $M = \sum_{k \in \mathcal{K}} M_k$. As was the case in previous chapters, we do not assume the statistics of $\mathbf{y}$ to be known, which implies that the algorithm has to estimate or learn these on the fly based on incoming sensor data at the different nodes.

Table 7.1: Examples of problems with fractional objectives as in (7.8).

In RTLS, d is a target signal that is assumed to be known, and A is a deterministic matrix used for Tikhonov regularization.

Problem	Cost function to minimize	Constraints
TRO [94]	$-\dfrac{\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]}{\mathbb{E}[\|\|X^T\mathbf{v}(t)\|\|^2]}$	$X^TX = I_Q$ $\Leftrightarrow (X^TB)(X^TB)^T = I_Q$ with $B = I_M$
RTLS [99, 100]	$\dfrac{\mathbb{E}[\|\|\mathbf{x}^T\mathbf{y}(t) - d(t)\|\|^2]}{1 + \|\|\mathbf{x}\|\|^2}$	$\|\|\mathbf{x}^TA\|\|^2 \leq \delta^2$

7.3.1 Fractional problems for signal fusion in WSNs

In this chapter, we are interested in spatial filters $X \in \mathbb{R}^{M\times Q}$ that are the solution of some fractional program. Two well-known examples, namely TRO and regularized total least squares (RTLS), are shown in Table 7.1 for illustration purposes (see also Chapter 6 for a detailed review of the TRO problem).

A generic instance of such fractional programs for spatial filtering and signal fusion can be written as

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \varrho(X^T\mathbf{y}(t), X^TB) \triangleq \frac{\varphi_1\left(X^T\mathbf{y}(t), X^TB\right)}{\varphi_2\left(X^T\mathbf{y}(t), X^TB\right)} \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) \leq 0, \quad \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) = 0, \quad \forall j \in \mathcal{J}_E. \end{aligned} \tag{7.8}$$

As in the case of the DASF framework, the functions η_j represent inequality constraints for $j \in \mathcal{J}_I$ and equality constraints for $j \in \mathcal{J}_E$, while the full index set of constraints is denoted as $\mathcal{J} = \mathcal{J}_I \cup \mathcal{J}_E$, such that we have $J = |\mathcal{J}|$ constraints in total. Throughout this chapter, we assume that the constraint set of (7.8), denoted as $\mathcal{S}$, is compact, and that the denominator is always strictly positive over $\mathcal{S}$, as in (7.1). The matrix B is again an $M \times L$ deterministic matrix independent of time. For example, in the TRO problem of Table 7.1, we have $B = I_M$ in the constraints. In a distributed context, we partition B

similar to (7.7):

$$B = [B_1^T, \ldots, B_K^T]^T, \tag{7.9}$$

where B_k is assumed to be known by node k. Note that the optimization variable X in (7.8) only appears in the fusion form $X^T\mathbf{y}$ or X^TB, which is a requirement for the DASF framework. Furthermore, every other parameter of the problem that is not fused with X is assumed to be known by every node (such as the signal d in the RTLS example in Table 7.1).

The optimization problems written in the form (7.8) represent a subclass of the DSFO problems (2.3) considered in the DASF framework, namely problems for which the objective is a fractional function. Our goal is to derive a new algorithm for these cases, which is computationally more attractive than straightforwardly applying the generic DASF "meta" algorithm to (7.8), as was the case in Chapter 6 for the TRO problem.

As in Section 2.3.3, we also allow Problem (7.8) to contain multiple signals $\mathbf{y}$ or matrices B, which have been omitted from (7.8) for notational conciseness. For example, the RTLS problem in Table 7.1 requires two different $B-$matrices. Indeed, we observe that we can take $B^{(1)} = I_M$ in the denominator of the objective function since $\|\mathbf{x}\|^2 = (\mathbf{x}^T \cdot I_M) \cdot (\mathbf{x}^T \cdot I_M)^T = (\mathbf{x}^T \cdot B^{(1)}) \cdot (\mathbf{x}^T \cdot B^{(1)})^T$. Similarly, we have $B^{(2)} = A$ in the constraint function. For the sake of intelligibility, we will continue the derivation and algorithm analysis for a single set of $\mathbf{y}$, and B, yet we emphasize that this is not a hard restriction as the algorithm can be easily generalized to multiple signal variables and multiple deterministic matrices (by following a similar approach as in Section 2.3.3).

Note that in a centralized context, Problem (7.8) can be solved using the Dinkelbach procedure, where each auxiliary problem can be written as

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \varphi_1\left(X^T\mathbf{y}(t), X^TB\right) - \rho^i\, \varphi_2\left(X^T\mathbf{y}(t), X^TB\right) \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) \leq 0, \quad \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) = 0, \quad \forall j \in \mathcal{J}_E, \end{aligned} \tag{7.10}$$

with

$$\rho^i = \frac{\varphi_1\left(X^{iT}\mathbf{y}(t), X^{iT}B\right)}{\varphi_2\left(X^{iT}\mathbf{y}(t), X^{iT}B\right)}. \tag{7.11}$$

Since the minimization of Problem (7.8) is done over X only, we define the following functions in order to compactify some of the equations throughout

this chapter:

$$\begin{aligned} r(X) &\triangleq \varrho(X^T\mathbf{y}(t), X^TB), \\ f_j(X) &\triangleq \varphi_j(X^T\mathbf{y}(t), X^TB),\ j \in \{1,2\}, \\ h_j(X) &\triangleq \eta_j(X^T\mathbf{y}(t), X^TB),\ \forall j \in \mathcal{J}. \end{aligned} \tag{7.12}$$

These definitions allow us to write Problem (7.8) as in (7.1) and the auxiliary problems (7.10) as in (7.4), where $\mathcal{S}$ denotes the constraint set defined by the constraint functions h_j. In the remaining parts of this chapter, we will mention interchangeably (7.1) and (7.8) to refer to the problem we aim to solve while interchangeably using (7.4) and (7.10) for its auxiliary problems, depending on whether we want to emphasize the specific $X^T\mathbf{y}(t)$, or X^TB, structure or not. As in Section 7.2, we define X^* to be a solution of Problem (7.8) and $\mathcal{X}^*$ to be the full solution set.

7.3.2 Adaptivity and approximations of statistics

As in the case of the DASF framework, the signal $\mathbf{y}$ is stochastic, therefore the functions $\varphi_1, \varphi_2, \eta_j$ implicitly contain an operator to transform probabilistic quantities into deterministic values, such that the problem solved in practice is deterministic. The most common operator encountered in signal processing applications is the expectation, i.e., there exist deterministic functions Φ_1, Φ_2, H_j such that

$$\begin{aligned} \varphi_j(X^T\mathbf{y}(t), X^TB) &= \mathbb{E}[\Phi_j(X^T\mathbf{y}(t), X^TB)],\ j \in \{1,2\}, \\ \eta_j(X^T\mathbf{y}(t), X^TB) &= \mathbb{E}[H_j(X^T\mathbf{y}(t), X^TB)],\ \forall j \in \mathcal{J}. \end{aligned} \tag{7.13}$$

A common way to approximate these functions in a practical implementation is to take N time samples of the signal $\mathbf{y}$, say $\{\mathbf{y}(\tau)\}_{\tau=t}^{t+N-1}$, and approximate the expectation through a sample average, for example:

$$\mathbb{E}[\Phi_j(X^T\mathbf{y}(t), X^TB)] \approx \frac{1}{N}\sum_{\tau=t}^{t+N-1} \Phi_j(X^T\mathbf{y}(\tau), X^TB). \tag{7.14}$$

Under the ergodicity assumption on the signal $\mathbf{y}$, the expression (7.14) gives an accurate approximation for sufficiently large N. In practical realizations, the objective functions and constraints in (7.8) will be evaluated on such sample batches of size N.

Furthermore, the stationarity assumption imposed on $\mathbf{y}$ is merely added for mathematical tractability, as it allows us to remove the time-dependence, as it

is often done in the adaptive signal processing literature to analyze asymptotic convergence, i.e., when $N \to +\infty$. However, in practice, we do not require the signals to be stationary, but rather short-term stationary and/or with slowly varying statistics. In Section 7.6, we will demonstrate that the proposed method is indeed able to track changes in the statistical properties of the signal, as is the case for the DASF algorithm.

7.3.3 General assumptions

To be able to state convergence guarantees of the proposed algorithm, we require some assumptions on Problem (7.8). These assumptions are similar to those for the DASF algorithm given in Chapters 2 and 3, the main difference being that some assumptions are required to hold for the auxiliary problems as well, in order for our proposed method to work. In practice, if Problem (7.8) satisfies these assumptions, the corresponding auxiliary problems typically do so as well.

Assumption 5. The targeted instance of Problem (7.8) and its corresponding auxiliary problems (7.10) are well-posed, in the sense that the solution set is not empty and varies continuously with a change in the parameters of the problem.

Assumption 6. The solutions of Problem (7.8) and its corresponding auxiliary problems (7.10) satisfy the linear independence constraint qualifications (LICQ), i.e., X^* satisfies the KKT conditions.

An additional assumption requiring the compactness of the sublevel sets of the objective of (7.10) was required in the original DASF algorithm, namely Assumption 3 in Chapters 2 and 3, its purpose being to ensure that the points generated by the algorithm lie in a compact set. As we will show in Section 7.5, this assumption can be omitted in the case of the proposed F-DASF algorithm due to the fact that we only consider fractional programs with compact constraint sets. Note that the latter assumption should not be viewed as a limiting condition, as we will empirically demonstrate in Section 7.6.2 through a simulation that the F-DASF algorithm introduced in Section 7.4 can still converge to the correct solution for non-compact constraint sets in problems for which the centralized Dinkelbach procedure also converges, yet without a theoretical guarantee.

7.4 Cost-efficient DASF for fractional programs

In this section, we propose a new algorithm based on the DASF framework for solving (7.8) in a distributed and adaptive fashion, where the required statistics of $\mathbf{y}$ are estimated and learned on the fly based on the incoming sensor observations. As mentioned earlier, the DASF algorithm can be straightforwardly applied to (7.8) to obtain such a distributed algorithm, where the nodes would use a solver for the original (centralized) problem to solve locally compressed instances of the network-wide problem. However, this would require a node to fully execute Dinkelbach's procedure (Algorithm 8) within each iteration of the DASF algorithm. Since Algorithm 8 is itself iterative, we would obtain nested iterations of Algorithm 8 within the DASF algorithm's iterations, which would result in a large computational cost at each node. Instead, we propose to interleave the steps of the DASF algorithm with the iterations of Dinkelbach's procedure, where each DASF iteration only requires performing one iteration of Algorithm 8, thereby greatly reducing the computational burden. This generalizes the DTRO algorithm presented in Chapter 6 for the TRO problem to other fractional programs. We will refer to this algorithm as the fractional DASF (F-DASF) algorithm, for which a convergence analysis will be presented in Section 7.5.

7.4.1 Data flow of fractional DASF (F-DASF)

This subsection describes the data flow within the F-DASF algorithm. It is noted that a significant part of this description is equivalent to the DASF framework's, yet it is included here for self-containedness.

At each iteration i, we select an arbitrary node $q \in \mathcal{K}$ to be the updating node. As in the previous chapters, the node that is selected to be the updating node changes at each iteration. Based on the selected node, the network is pruned so as to obtain a tree $\mathcal{T}^i(\mathcal{G}, q)$, such that there is a unique path connecting each pair of nodes. The pruning $\mathcal{T}^i$ can again be chosen freely, however, it should avoid cutting any link between the updating node q and its neighbors $n \in \mathcal{N}_q$, where we denote as $\mathcal{N}_q$ the set of nodes neighboring node q with respect to the graph $\mathcal{T}^i(\mathcal{G}, q)$, i.e., the pruned network.

Let us define X^i to be the estimation of the global filter X at iteration i and partitioned as

$$X^i = [X_1^{iT}, \ldots, X_K^{iT}]^T, \tag{7.15}$$

where X_k is a $M_k \times Q$ matrix such that $X^{iT}\mathbf{y} = \sum_k X_k^{iT}\mathbf{y}_k$ and $X^{iT}B = \sum_k X_k^{iT}B_k$. At the beginning of each iteration i, every node $k \in \mathcal{K}\backslash\{q\}$ uses

its current estimate X_k^i to compress its M_k-channel sensor signal $\mathbf{y}_k$ into a $Q-$channel signal, assuming $Q \leq M_k$. A similar operation is done on B_k to obtain

$$\widehat{\mathbf{y}}_k^i(t) \triangleq X_k^{iT}\mathbf{y}_k(t),\ \widehat{B}_k^i \triangleq X_k^{iT}B_k. \tag{7.16}$$

Given X^i, the objective and constraints of Problems (7.8) and (7.10) can then be evaluated at $X = X^i$ at a certain node if that node has access to the sums

$$X^{iT}\mathbf{y}(t) = \sum_{k\in\mathcal{K}} \widehat{\mathbf{y}}_k^i(t),\ X^{iT}B = \sum_{k\in\mathcal{K}} \widehat{B}_k^i. \tag{7.17}$$

As in the DASF framework, the idea is to reconstruct these sums at the updating node q by fusing and forwarding the signals $\widehat{\mathbf{y}}_k^i$ and matrices $\widehat{B}_k^i$ towards node q. For the signals $\widehat{\mathbf{y}}_k^i$, this means that each node k will need to transmit N samples, where N is large enough to be able to accurately estimate the statistics required to evaluate the objective and constraints of (7.8) and (7.10).

The data is then fused and forwarded towards node q in a similar fashion as to the DASF algorithm. Each node $k \neq q$ waits until it receives data from all its neighboring nodes l except one, say node n. Upon receiving this data, node k sums all the data received from its neighbors to its own compressed data (7.16) and transmits this to node n. The data transmitted to node n from node k is therefore N samples of

$$\widehat{\mathbf{y}}_{k\to n}^i(t) = X_k^{iT}\mathbf{y}_k(t) + \sum_{l\in\mathcal{N}_k\setminus\{n\}} \widehat{\mathbf{y}}_{l\to k}^i(t). \tag{7.18}$$

Note that the second term of (7.18) is recursive, and vanishes for leaf nodes, i.e., nodes with a single neighbor. Therefore, the data transmission is initiated at leaf nodes of the pruned network $\mathcal{T}^i(\mathcal{G}, q)$. The data eventually arrives at the updating node q, which receives N samples of

$$\widehat{\mathbf{y}}_{n\to q}^i(t) = X_n^{iT}\mathbf{y}_n(t) + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{\mathbf{y}}_{k\to n}^i(t) = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i(t) \tag{7.19}$$

from all its neighbors $n \in \mathcal{N}_q$. The set $\mathcal{B}_{nq}$ in (7.19) contains the nodes in the subgraph that includes node n after cutting the edge between nodes n and q in $\mathcal{T}^i(\mathcal{G}, q)$ (see Figure 2.3 for an example). A similar procedure is applied for the deterministic matrix B, such that node q receives

$$\widehat{B}_{n\to q}^i = X_n^{iT}B_n + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{B}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} \widehat{B}_k^i, \tag{7.20}$$

from $n \in \mathcal{N}_q$.

7.4.2 Updating scheme of F-DASF

In the previous subsection, we concluded that the data transmitted to the updating node q is given by $\widehat{\mathbf{y}}^i_{n\to q}$ and $\widehat{B}^i_{n\to q}$, sent from all neighbors $n \in \mathcal{N}_q$ of q. In this subsection, we discuss the updating scheme of the F-DASF algorithm, i.e., how X^{i+1} is obtained. This updating scheme of the F-DASF algorithm is significantly different from the original DASF algorithm as we will interleave the steps of the Dinkelbach procedure with the DASF iterations.

Let us label the neighboring nodes of node q as $\mathcal{N}_q \triangleq \{n_1, \ldots, n_{|\mathcal{N}_q|}\}$. Node q's own observation $\mathbf{y}_q$ and the compressed signals obtained from its neighbors can then be concatenated for each sample at time t as

$$\widetilde{\mathbf{y}}^i_q(t) \triangleq [\mathbf{y}^T_q(t), \widehat{\mathbf{y}}^{iT}_{n_1\to q}(t), \ldots, \widehat{\mathbf{y}}^{iT}_{n_{|\mathcal{N}_q|}\to q}(t)]^T \in \mathbb{R}^{\widetilde{M}_q}, \tag{7.21}$$

where $\widetilde{M}_q = |\mathcal{N}_q| \cdot Q + M_q$, and similarly for the deterministic term B:

$$\widetilde{B}^i_q \triangleq [B^T_q, \widehat{B}^{iT}_{n_1\to q}, \ldots, \widehat{B}^{iT}_{n_{|\mathcal{N}_q|}\to q}]^T \in \mathbb{R}^{\widetilde{M}_q \times L}. \tag{7.22}$$

Defining $\widetilde{X}^i_q$ as

$$\widetilde{X}^i_q \triangleq [X^{iT}_q, I_Q, \ldots, I_Q]^T, \tag{7.23}$$

and using (7.17), we can write

$$X^{iT}\mathbf{y}(t) = \widetilde{X}^{iT}_q \widetilde{\mathbf{y}}^i_q(t), \; X^{iT}B = \widetilde{X}^{iT}_q \widetilde{B}^i_q. \tag{7.24}$$

The above equation implies that the output of the network-wide spatial filtering operations $X^{iT}\mathbf{y}$ and $X^{iT}B$ can be computed from the compressed data at node q. This allows node q to evaluate the objective of (7.8) at X^i:

$$\rho^i = \varrho\left(\widetilde{X}^{iT}_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^{iT}_q \widetilde{B}^i_q\right) = \frac{\varphi_1\left(\widetilde{X}^{iT}_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^{iT}_q \widetilde{B}^i_q\right)}{\varphi_2\left(\widetilde{X}^{iT}_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^{iT}_q \widetilde{B}^i_q\right)}. \tag{7.25}$$

Moreover, node q will then be able to construct a compressed version of the auxiliary problem (7.10) locally

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \varphi_1\left(\widetilde{X}^T_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^T_q \widetilde{B}^i_q\right) - \rho^i \varphi_2\left(\widetilde{X}^T_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^T_q \widetilde{B}^i_q\right) \\ \text{subject to} \quad & \eta_j\left(\widetilde{X}^T_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^T_q \widetilde{B}^i_q\right) \leq 0 \;\; \forall j \in \mathcal{J}_I, \\ & \eta_j\left(\widetilde{X}^T_q \widetilde{\mathbf{y}}^i_q(t), \widetilde{X}^T_q \widetilde{B}^i_q\right) = 0 \;\; \forall j \in \mathcal{J}_E. \end{aligned} \tag{7.26}$$

At this point, we should note that solving (7.26) corresponds to executing a single iteration of Dinkelbach's procedure in Algorithm 8. If we would have straightforwardly applied the standard DASF algorithm, the objective function in (7.26) would be replaced with a new fractional program given by

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \frac{\varphi_1\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right)}{\varphi_2\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right)} \\ \text{subject to} \quad & \eta_j\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_j\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{7.27}$$

This problem is more difficult to solve and would actually require solving multiple instances of (7.26) in order to execute all iterations in Algorithm 8.

We define $\widetilde{X}_q^*$ as the solution of (7.26), i.e.,

$$\widetilde{X}_q^* \triangleq \underset{\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}} \ \varphi_1\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right) - \rho^i \varphi_2\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\right), \tag{7.28}$$

where $\widetilde{\mathcal{S}}_q^i$ is the constraint set of (7.26). If (7.26) does not have a unique solution, $\widetilde{X}_q^*$ is selected as the solution closest to $\widetilde{X}_q^i$ in (7.23), i.e.,

$$\widetilde{X}_q^* = \underset{\widetilde{X}_q \in \widetilde{\mathcal{X}}_q^i}{\operatorname{argmin}} \ \|\widetilde{X}_q - \widetilde{X}_q^i\|_F \tag{7.29}$$

where $\widetilde{\mathcal{X}}_q^i$ is the set of possible solutions $\widetilde{X}_q$ of (7.26) and $\widetilde{X}_q^i$ is given in (7.23). Note that as in the DASF algorithm, the Frobenius norm to select $\widetilde{X}_q^*$ in (7.29) is an arbitrary choice, and another distance function can be selected as long as it is continuous.

Finally, let us partition $\widetilde{X}_q^*$ as

$$\widetilde{X}_q^* = [X_q^{(i+1)T}, G_{n_1}^{(i+1)T}, \ldots, G_{n_{|\mathcal{N}_q|}}^{(i+1)T}]^T, \tag{7.30}$$

such that G_n is $Q \times Q$, $\forall n \in \mathcal{N}_q$, where the $G-$matrices consist of the part of $\widetilde{X}_q^*$ that is multiplied with the compressed signals of the neighbors of node q in the inner product $\widetilde{X}_q^{*T} \widetilde{\mathbf{y}}_q^i$. Every matrix G_n^{i+1} is then disseminated in the pruned network to the corresponding subgraph $\mathcal{B}_{nq}$ through node $n \in \mathcal{N}_q$, and every node k in the network can update its local variable X_k as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q, \\ X_k^i G_n^{i+1} & \text{if } k \in \mathcal{B}_{nq},\ n \in \mathcal{N}_q. \end{cases} \tag{7.31}$$

Algorithm 9: Fractional DASF (F-DASF) algorithm

X^0 initialized randomly, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) The network $\mathcal{G}$ is pruned into a tree $\mathcal{T}^i(\mathcal{G}, q)$.
  2) Every node k collects N samples of $\mathbf{y}_k$. All nodes compress these to N samples of $\widehat{\mathbf{y}}_k^i$ and also compute $\widehat{B}_k^i$ as in (7.16).
  3) The nodes sum-and-forward their compressed data towards node q via the recursive rule (7.18) (and a similar rule for the $\widehat{B}_k^i$'s). Node q eventually receives N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ given in (7.19), and the matrix $\widehat{B}_{n\to q}^i$ defined in (7.20), from all its neighbors $n \in \mathcal{N}_q$.
  at Node q **do**
    4a) Compute ρ^i as in (7.25).
    4b) Compute (7.28), i.e., execute a single Dinkelbach iteration by solving (7.26), resulting in $\widetilde{X}_q^*$. If the solution is not unique, select a solution via (7.29).
    4c) Partition $\widetilde{X}_q^*$ as in (7.30).
    4d) Disseminate every G_n^{i+1} in the corresponding subgraph $\mathcal{B}_{nq}$.
  end
  5) Every node updates X_k^{i+1} according to (7.31).
  $i \leftarrow i + 1$

This process is then repeated by selecting different updating nodes at different iterations and using new batches of N samples of signals $\mathbf{y}_k$. Algorithm 9 summarizes the steps of the proposed F-DASF algorithm presented in this section.

7.5 Technical analysis and convergence

In this section, we analyze the convergence properties of the F-DASF algorithm. Similar to the proof in Chapter 3, for mathematical tractability, we assume that all signals are stationary and that each node is able to perfectly estimate the statistics of its locally available signals, i.e., as if $N \to +\infty$. In practice, estimation errors on these statistics (due to finite N) will result in the algorithm only converging to a neighborhood of the true optimal point, where the solution will "hover" around this optimal point. In that sense, the proof should be viewed as an asymptotic convergence proof.

7.5.1 Preliminaries

We start by describing the relationship between the local problems and the global problem, which, although similar to the case of the DASF algorithm, is repeated here as it will be useful for the technical analyses that will be presented in the later parts of this section. From equations (7.19)-(7.22), we have

$$\widetilde{\mathbf{y}}_q^i(t) = C_q^{iT}\mathbf{y}(t),\ \widetilde{B}_q^i = C_q^{iT}B, \tag{7.32}$$

i.e., there exists a linear (compressive) relationship between the local data at the updating node q and the global data $\mathbf{y}$ and B. The matrix $C_q^i \in \mathbb{R}^{M\times\widetilde{M}_q}$ is the same matrix as in the DASF algorithm, given in (2.50) and repeated here:

$$C_q^i = \left[\begin{array}{c|c} 0 & \\ I_{M_q} & \Theta_{-q}^i \\ 0 & \end{array}\right]. \tag{7.33}$$

In (7.33) I_{M_q} is placed in the $q-$th block-row, and Θ_{-q}^i is a block matrix with K block-rows and $|\mathcal{N}_q|$ block-columns, where the block at the $k-$th block-row and $m-$th block-column is represented by $\Theta_{-q}^i(k,m) \in \mathbb{R}^{M_k\times Q}$. Each block-column corresponds to one of the neighbors $n \in \mathcal{N}_q$ of q, which we re-index as $m_n \in \{1,\ldots,|\mathcal{N}_q|\}$, such that

$$\Theta_{-q}^i(k,m_n) = \begin{cases} X_k^i & \text{if } k \in \mathcal{B}_{nq}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.34}$$

Similar to (7.24), when node q applies a generic spatial filter $\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q\times Q}$ to $\widetilde{\mathbf{y}}_q^i$, it is equivalent to applying a generic spatial filter $X \in \mathbb{R}^{M\times Q}$ to the network-wide sensor signal $\mathbf{y}$, i.e., $\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i = X^T\mathbf{y}$. With (7.32), we then have $\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i = \widetilde{X}_q^T(C_q^{iT}\mathbf{y}) = (C_q^i\widetilde{X}_q)^T\mathbf{y}$, while following similar steps for the deterministic matrix B gives $\widetilde{X}_q^T\widetilde{B}_q^i = \widetilde{X}_q^T(C_q^{iT}B) = (C_q^i\widetilde{X}_q)^TB$, such that

$$X = C_q^i\widetilde{X}_q. \tag{7.35}$$

When comparing (7.10) with (7.26), we observe that their objective and constraint functions are indeed the same if $\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i = X^T\mathbf{y}$. Therefore, Lemma 2.1 also applies here such that if $\widetilde{X}_q$ is a feasible point of the local auxiliary problem (7.26), the point X parameterized by (7.35) is a feasible point of the global auxiliary problem (7.10), and vice versa, as formalized in the following lemma (the proof is equivalent to the proof provided in Lemma 2.1 for the original DASF algorithm and is therefore omitted here).

Lemma 7.1. For any iteration $i > 0$ of Algorithm 9,

$$\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i \Leftrightarrow C_q^i \widetilde{X}_q \in \mathcal{S}. \tag{7.36}$$

In particular, $X^i \in \mathcal{S}$ and $\widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i$ for all $i > 0$, where $\widetilde{X}_q^i$ is defined in (7.23).

We note that this lemma always holds, independent of whether the initialization point of F-DASF is in $\mathcal{S}$ or not. In particular, the last sentence of the lemma is important as it implies that every output X^i of the F-DASF algorithm is a feasible point of the global auxiliary problem (7.10) and hence also of the global problem (7.8), since they have the same constraint set.

The relationship (7.35) also allows us to compactly write Problem (7.26) using the functions f_j and h_j defined in (7.12):

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & f(C_q^i \widetilde{X}_q, \rho^i) = f_1\left(C_q^i \widetilde{X}_q\right) - \rho^i f_2\left(C_q^i \widetilde{X}_q\right) \\ \text{subject to} \quad & h_j\left(C_q^i \widetilde{X}_q\right) \leq 0, \quad \forall j \in \mathcal{J}_I, \\ & h_j\left(C_q^i \widetilde{X}_q\right) = 0, \quad \forall j \in \mathcal{J}_E, \end{aligned} \tag{7.37}$$

where

$$\rho^i = \frac{f_1\left(C_q^i \widetilde{X}_q^i\right)}{f_2\left(C_q^i \widetilde{X}_q^i\right)}. \tag{7.38}$$

In the remaining parts of this section, we will often refer to the compact notation in equations (7.1), (7.4) and (7.37) to refer to the global problem, its auxiliary problems, and the compressed auxiliary problem, respectively (instead of (7.8), (7.10) and (7.26)) for notational convenience.

7.5.2 Convergence of the objective

The following theorem establishes the convergence of the auxiliary parameter ρ^i, which then also implies that there is convergence in the objective of (7.1) / (7.8) via the relationship in (7.25).

Theorem 7.3. The sequence $(\rho^i)_i$ generated by Algorithm 9 is a non-increasing, converging sequence.

Proof. From (7.38), we have $f\big(C_q^i \widetilde{X}_q, \rho^i\big) = f_1\big(C_q^i \widetilde{X}_q^i\big) - \rho^i f_2\big(C_q^i \widetilde{X}_q^i\big) = 0$. Since $\widetilde{X}_q^*$ solves (7.37), we have that $f\big(C_q^i \widetilde{X}_q^*, \rho^i\big) \leq f\big(C_q^i \widetilde{X}_q, \rho^i\big)$ for any $\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i$. We note that $\widetilde{X}_q^i$ as defined in (7.23) is a feasible point of (7.26), i.e., belongs to $\widetilde{\mathcal{S}}_q^i$, as shown in Lemma 7.1. We then write $f\big(C_q^i \widetilde{X}_q^*, \rho^i\big) \leq f\big(C_q^i \widetilde{X}_q^i, \rho^i\big) = 0$. After reordering the terms of $f\big(C_q^i \widetilde{X}_q^*, \rho^i\big)$, we obtain $\frac{f_1\left(C_q^i \widetilde{X}_q^*\right)}{f_2\left(C_q^i \widetilde{X}_q^*\right)} = \rho^{i+1} \leq \rho^i$. Therefore, the sequence $(\rho^i)_i$ is monotonic non-increasing and since it is lower bounded by ρ^*, it must converge. □

As in the DASF algorithm, convergence of the sequence $(\rho^i)_i$ does not necessarily imply convergence of the underlying sequence $(X^i)_i$, nor the optimality of the resulting accumulation point, which is the topic of the next three subsections.

7.5.3 Stationarity of fixed points

A fixed point of the F-DASF algorithm is defined as a point $\overline{X} \in \mathcal{S}$ that is invariant under any F-DASF update step (for any updating node q), i.e., $X^i = \overline{X}$, $\forall i > 0$ when initializing Algorithm 9 with $X^0 = \overline{X}$. We will now present results that guarantee that such fixed points of the F-DASF algorithm are stationary points of the global problem (7.8) under mild technical conditions that are akin to the standard LICQ conditions in the optimization literature [118]. Since these conditions are very similar to those in the original DASF algorithm, we only discuss them briefly here, and refer to Chapter 3 for more details on their practical implications.

Condition 4a. For a fixed point $\overline{X}$ of Algorithm 9, the elements of the set $\{\overline{X}^T \nabla_X h_j(\overline{X})\}_{j \in \mathcal{J}}$ are linearly independent.

We restate the remark from Section 3.3.2 that Condition 4a requires the linear independence of a set of J matrices of size $Q \times Q$, which can only be satisfied if

$$J \leq Q^2. \tag{7.39}$$

When Condition 4a is satisfied, we obtain a first result on the stationarity of the fixed points of the F-DASF algorithm.

Theorem 7.4. If Condition 4a is satisfied for a fixed point $\overline{X}$ of Algorithm 9, then $\overline{X}$ is a stationary point of both (i) the auxiliary problem (7.4) (or (7.10)) for $\rho = r(\overline{X})$ and (ii) the global problem (7.1) (or (7.8)), satisfying their KKT conditions.

Proof. See Section 7.A. □

For the multi-input single-output (MISO) case, the filter X is in fact a vector such that $Q = 1$. As mentioned in Section 3.3.2, (7.39) implies in this case that Condition 4a can only be satisfied if we have at most one constraint in Problem (7.8). An alternative condition for such cases in order to relax the upper bound on the number of constraints is then as follows.

Condition 4b. For a fixed point $\overline{X}$ of Algorithm 9, the elements of the set $\{D_{j,q}(\overline{X})\}_{j\in\mathcal{J}}$ are linearly independent for any q, where

$$D_{j,q}(\overline{X}) \triangleq \begin{bmatrix} \overline{X}_q^T \nabla_{X_q} h_j(\overline{X}) \\ \sum_{k\in\mathcal{B}_{n_1 q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \\ \vdots \\ \sum_{k\in\mathcal{B}_{n_{|\mathcal{N}_q|} q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \end{bmatrix}, \tag{7.40}$$

which is a block-matrix containing $(1 + |\mathcal{N}_q|)$ blocks of $Q \times Q$ matrices.

The size of the matrices $D_{j,q}$ depends on the number of neighbors of node q, therefore Condition 4b depends on the topology of the network. In order to satisfy the linear independence of the matrices (7.40), a similar upper bound was provided in Section 3.3.2 for Condition 1b:

$$J \leq (1 + \min_{k\in\mathcal{K}} |\mathcal{N}_k|) \cdot Q^2, \tag{7.41}$$

where it is assumed that the pruning function $\mathcal{T}^i(\mathcal{G}, q)$ preserves all the links of the updating node q. Additionally, we had seen in Section 3.C that we require the number of constraints J to be upper bounded as

$$J \leq \frac{Q^2}{K-1} \sum_{k\in\mathcal{K}} |\mathcal{N}_k| \tag{7.42}$$

for Condition 4b to be satisfied. Combining both (7.41) and (7.42), we have

$$J \leq \min\left(\frac{Q^2}{K-1}\sum_{k\in\mathcal{K}}|\mathcal{N}_k|,\ (1+\min_{k\in\mathcal{K}}|\mathcal{N}_k|)\cdot Q^2\right). \tag{7.43}$$

This is a more relaxed bound than the one in (7.39), yet it requires knowing the full topology of the network. As in the case of the previous condition (and the equivalent conditions for the DASF algorithm in Chapter 3), Condition 4b is merely a technical condition, as it is highly likely to be satisfied in practice when (7.43) holds. Condition 4b leads to a result analogous to Theorem 7.4, given below.

Theorem 7.5. If Condition 4b is satisfied for a fixed point $\overline{X}$ of Algorithm 9, then $\overline{X}$ is a stationary point of both (i) the auxiliary problem (7.4) (or (7.10)) for $\rho = r(\overline{X})$ and (ii) the global problem (7.1) (or (7.8)), satisfying their KKT conditions.

The proof of this theorem differs from the one of Theorem 7.4 only in an aspect that is the same for both the F-DASF and DASF algorithms, which is why we omit it here. For more details, we refer to Sections 3.B and 3.C of Chapter 3, in combination with the F-DASF specific proof of 7.4 provided in Section 7.A.

Theorems 7.4 and 7.5 only guarantee that a fixed point of Algorithm 9 is a stationary point of Problem (7.1). A stronger result can be obtained if the fixed point minimizes the auxiliary problem (7.4).

Theorem 7.6. (see [160]) If a point $\overline{X}$ minimizes the auxiliary problem (7.4) for $\rho = r(\overline{X})$, then it is a global minimizer of Problem (7.1).

Proof. We define $\bar{\rho}$ as $r(\overline{X}) = f_1(\overline{X})/f_2(\overline{X})$, which, after rearranging the terms, gives

$$f_1(\overline{X}) = \bar{\rho}\cdot f_2(\overline{X}) \Rightarrow f(\overline{X},\bar{\rho}) = 0. \tag{7.44}$$

Since it is assumed that $\overline{X}$ minimizes f over the constraint set $\mathcal{S}$, we can write

$$g(\bar{\rho}) = \min_{X\in\mathcal{S}} f(X,\bar{\rho}) = f(\overline{X},\bar{\rho}). \tag{7.45}$$

By combining (7.44) and (7.45), we therefore conclude that $g(\bar{\rho}) = 0$. From Theorem 7.1, $g(\bar{\rho}) = 0$ implies that $\bar{\rho} = \rho^*$ is the global minimum of (7.1). Therefore, $\overline{X}$ is a global minimizer of (7.1). □

Combining Theorems 7.4 to 7.6 we can obtain the following result.

Corollary 7.1. Let $\overline{X}$ be a fixed point of Algorithm 9 satisfying either Condition 4a or Condition 4b. Then, for $\rho = r(\overline{X})$, if the auxiliary problem (7.4) has a unique minimum and no other KKT points, we have $\overline{X} = X^*$, where X^* is a solution of (7.1).

Proof. From Theorems 7.4 and 7.5, we saw that $\overline{X}$ is a KKT point of both (7.1) and its auxiliary problem (7.4) for $\rho = r(\overline{X})$. Since the only KKT point of (7.4) is its unique minimum, $\overline{X}$ solves (7.4) for $\rho = r(\overline{X})$. From Theorem 7.6, we conclude that $\overline{X}$ also solves the global problem (7.1). □

7.5.4 Convergence of arguments

Conditions 4a and 4b allowed us to show that fixed points of the F-DASF algorithm are stationary points of (7.1). In this subsection, we give the conditions and results to guarantee that all accumulation points of the F-DASF algorithm are fixed points, and therefore also global minimizers of (7.1).

Condition 5. The local auxiliary problems (7.37) satisfy Assumptions 5 and 6.

This condition translates to requiring the local auxiliary problems to inherit the same properties of the centralized auxiliary problems (7.4). It is usually satisfied in practice since we already assumed that the centralized auxiliary problems (7.4) satisfy the assumptions presented in Section 7.3.3, while the local auxiliary problems are compressed versions of these.

Condition 6. The number of solutions of each local auxiliary problem (7.26) is finite or the solver of the local auxiliary problems (7.26) can only obtain a finite subset of the solutions of (7.26).

Note that in various fractional problems, the number of stationary points is not finite, for example in the TRO problem of Chapter 6. We have seen that a solution of the auxiliary problems of the TRO problem is given by an X

containing in its columns the Q eigenvectors of $R_{\mathbf{vv}} - \rho^i \cdot R_{\mathbf{yy}}$ corresponding to its Q largest eigenvalues. However, any matrix XU, where U is an orthogonal matrix, is also a solution of the auxiliary problem of the TRO. For these cases, Condition 6 can be relaxed so as to require that the solver used for solving the auxiliary problems (7.4) can only obtain a finite set of the solutions of (7.4). For the TRO example, a solver that outputs the Q principal unit norm eigenvectors of $R_{\mathbf{vv}} - \rho^i \cdot R_{\mathbf{yy}}$ is therefore sufficient, as the possible outputs are only different up to a sign change in each column, making the set of possible solutions finite (except for the contrived degenerate case where these Q eigenvalues are not all distinct).

These two conditions allow us to state the following result:

Theorem 7.7. Suppose that Condition 5 is satisfied for Problem (7.1) under the updates of Algorithm 9. Then:

1. Any accumulation point $\overline{X}$ of the sequence $(X^i)_i$ is a fixed point of Algorithm 9 for any q.
2. $\lim_{i\to+\infty} \|X^{i+1} - X^i\| = 0$,

If additionally, Condition 6 holds, then $(X^i)_i$ converges to a single fixed point $\overline{X}$.

The proof of Theorem 7.7 is a slight reformulation of the proofs of Theorems 3.4 and 3.5 provided in Sections 3.D and 3.E for the DASF algorithm, and is omitted here to avoid overlap. However, for the proofs in these sections to work in the case of F-DASF as well, we have to establish that all the points of the sequence $(X^i)_i$ lie in a compact set. This can be easily shown to be true for the F-DASF algorithm. Indeed, since the set $\mathcal{S}$ of Problem (7.1) is compact, and from the fact that all points of the sequence $(X^i)_i$ generated by the F-DASF algorithm lie in $\mathcal{S}$ (Lemma 7.1), the sequence $(X^i)_i$ lies in a compact set, therefore satisfying all conditions for proving the results presented in Theorem 7.7 via the proofs in Sections 3.D and 3.E.

7.5.5 Convergence to stationary points and global minima

The final step is to combine the previous results to be able to state the convergence guarantees of the F-DASF algorithm to a stationary point of (7.1).

Theorem 7.8. Suppose that Condition 4 (either the form a or b), 5 and 6 are satisfied for Problem (7.1) under the updates of Algorithm 9. Then the sequence $(X^i)_i$ converges and $\lim_{i\to+\infty} X^i = \overline{X}$, where $\overline{X}$ is a stationary point of (7.1) satisfying its KKT conditions.

Proof. From Theorem 7.7, $(X^i)_i$ converges to a single point $\overline{X}$. Therefore, $\overline{X}$ is a fixed point of Algorithm 9. From Theorems 7.4 and 7.5, fixed points of the F-DASF algorithm are stationary points of (7.1) satisfying its KKT conditions. □

Theorem 7.8 leads to stronger results if certain conditions are met on the uniqueness of the minimum of Problem (7.1), or the auxiliary problems solved at each node, as described in the two following results.

Corollary 7.2. Suppose that all conditions of Theorem 7.8 are satisfied such that the sequence $(X^i)_i$ converges to $\overline{X}$. If the auxiliary problem (7.4) has a unique minimum for $\rho = r(\overline{X})$ and no other KKT points, we then have $\overline{X} = X^*$ with X^* the global minimum of (7.1).

Proof. The proof comes directly from combining the results of Theorem 7.7 — implying the convergence of the sequence $(X^i)_i$ to a fixed point $\overline{X}$ — and Corollary 7.1. □

Corollary 7.3. If the global problem (7.1) has a unique minimum X^* and no other stationary points, and all conditions of Theorem 7.8 are satisfied, the sequence $(X^i)_i$ converges to X^*.

The previous result comes from the fact that under uniqueness of the minimum, we have a guarantee that the only possible KKT point is the global minimum X^*. Even if this uniqueness assumption is not met, we can still expect the F-DASF algorithm to converge to a global minimum if all minima are global minima. This is because of the monotonic decrease of $(r(X^i))_i = (\rho^i)_i$ (see Theorem 7.3), which implies that the sequence $(X^i)_i$ is kicked out of a potential equilibrium point which is not a minimum and cannot return to it, making it unstable.

Corollary 7.4. Under the same settings of Theorem 7.8, if all minima of Problem (7.1) are global minima, the only stable fixed points of Algorithm 9 are in $\mathcal{X}^*$.

In general, we see that the F-DASF algorithm converges under similar technical conditions as the DASF algorithm, except for the fact that some of these conditions are required to hold for the auxiliary problems (instead of the original problem) in the F-DASF case.

7.6 Simulations

In this section, we demonstrate the performance of the F-DASF algorithm in various experimental settings and compare it to the DASF algorithm for two new fractional programs. Throughout these experiments, we consider network topologies generated randomly using the Erdős-Rényi model with connection probability 0.8, and where the pruning function $\mathcal{T}^i(\cdot, q)$ is chosen to be the shortest path. The signal $\mathbf{y}$ follows a mixture model:

$$\mathbf{y}(t) = \Pi \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{7.46}$$

with $\mathbf{s}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$, $\mathbf{n}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_n^2)$ for every entry and time instant t. The entries of the mixture matrix Π are independent of time and are independently drawn from $\mathcal{N}(0, \sigma_\Pi^2)$. For each experiment, the number of channels M_k of the signals measured at node k are equal for each node, and given by M/K. Table 7.2 gives an overview of the different parameters selected for the simulations presented below. As in previous chapters, the performance metric we use to assess convergence of the F-DASF algorithm is the MedSE ϵ:

$$\epsilon(i) = \text{median}\left(\frac{\|X^i - X^*\|_F^2}{\|X^*\|_F^2}\right), \tag{7.47}$$

taken over multiple Monte Carlo runs, where X^* is an optimal solution of the respective problem, obtained from a centralized solver implementing Dinkelbach's procedure. In the results we present next a stopping criterion for Dinkelbach's procedure used in DASF has been set to be a threshold of 10^{-8} on the norm of the difference of two consecutive $\widetilde{X}_q$'s and a maximum number of iterations of 10.

In Section 7.6.1, we will consider both a stationary setting and also time-varying mixture matrices to simulate non-stationarity in an adaptive context, while Section 7.6.2 demonstrates convergence for a problem with a non-compact constraint set in a stationary setting.

Table 7.2: Summary of parameters used in the simulations of the F-DASF algorithm.

Experiment	Section 7.6.1	Section 7.6.2
Q	1	2
K	15	10
M	75	100
Signal Statistics	$\sigma^2 = 0.5$, $\sigma_n^2 = 0.1$, $\sigma_\Pi^2 = 0.2$	$\sigma^2 = 0.5$, $\sigma_n^2 = 0.2$, $\sigma_\Pi^2 = 0.2$
N	10000	
Monte Carlo Runs	100	

7.6.1 Regularized total least squares

The regularized total least squares (RTLS) problem [99, 100] is written as

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^M}{\text{minimize}} \quad & \frac{\mathbb{E}[|\mathbf{x}^T\mathbf{y}(t) - d(t)|^2]}{1 + \mathbf{x}^T\mathbf{x}} = \frac{\mathbf{x}^T R_{\mathbf{yy}}\mathbf{x} - 2\mathbf{x}^T\mathbf{r}_{\mathbf{y}d} + r_{dd}}{1 + \mathbf{x}^T\mathbf{x}} \\ \text{subject to} \quad & \|\mathbf{x}^T L\|^2 \leq 1, \end{aligned} \tag{7.48}$$

where we have $X = \mathbf{x} \in \mathbb{R}^M$, i.e., $Q = 1$, $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$, $\mathbf{r}_{\mathbf{y}d} = \mathbb{E}[d(t)\mathbf{y}(t)]$ and $r_{dd} = \mathbb{E}[d^2(t)]$. The RTLS problem has applications in signal estimation tasks when both the observation and source signals are noisy [166–168]. Note that in (7.48), we have two deterministic matrices B, $B_1 = I_M$ and $B_2 = L$, where the former appears in the denominator of the objective: $\mathbf{x}^T\mathbf{x} = (\mathbf{x}^T I_M) \cdot (\mathbf{x}^T I_M)^T$.

We first consider a stationary setting where (7.46) is of the form $\mathbf{y}(t) = \mathbf{p} \cdot s(t) + \mathbf{n}(t)$, i.e., $\Pi = \mathbf{p}$ is a vector and s is a scalar. Moreover, d represents a noisy version of s, given by $d(t) = s(t) + w(t)$, where the time samples of w follow $\mathcal{N}(0, 0.01)$. Finally, L is a diagonal matrix where each element of the diagonal is drawn from $\mathcal{N}(1, 0.1)$. At each iteration i, a batch of $N = 10^4$ samples of $\mathbf{y}$ and d is used to solve the RTLS problem, such that the relationship between i and t is $i = \lfloor t/N \rfloor$.

For a fixed ρ, the auxiliary problem of (7.48) is given by

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^M}{\text{minimize}} \quad & \mathbf{x}^T R_{\mathbf{yy}}\mathbf{x} - 2\mathbf{x}^T\mathbf{r}_{\mathbf{y}d} + r_{dd} - \rho \cdot (1 + \mathbf{x}^T\mathbf{x}) \\ \text{subject to} \quad & \|\mathbf{x}^T L\|^2 \leq 1. \end{aligned} \tag{7.49}$$

At iteration i of Algorithm 9, the problem solved at node q is the compressed auxiliary problem (7.26):

$$\begin{aligned} \underset{\widetilde{\mathbf{x}}_q \in \mathbb{R}^{\widetilde{M}_q}}{\text{minimize}} \quad & \widetilde{\mathbf{x}}_q^T R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} \widetilde{\mathbf{x}}_q - 2\widetilde{\mathbf{x}}_q^T \mathbf{r}^i_{\widetilde{\mathbf{y}}_q d} + r_{dd} - \rho^i \cdot (1 + \widetilde{\mathbf{x}}_q^T \widetilde{\Gamma}^i_q \widetilde{\mathbf{x}}_q) \\ \text{subject to} \quad & \|\widetilde{\mathbf{x}}_q^T \widetilde{L}^i_q\|^2 \leq 1, \end{aligned} \tag{7.50}$$

where $R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} = \mathbb{E}[\widetilde{\mathbf{y}}^i_q(t) \widetilde{\mathbf{y}}^{iT}_q(t)]$ and $\mathbf{r}^i_{\widetilde{\mathbf{y}}_q d} = \mathbb{E}[\widetilde{\mathbf{y}}^i_q(t) d(t)]$, with $\widetilde{\mathbf{y}}^i_q$ the locally available signal defined in (7.21). The matrix $\widetilde{L}^i_q$ is the locally available versions of L obtained by taking $B = L$ and computing $\widetilde{B}^i_q$ as in (7.22). The matrix $\widetilde{\Gamma}^i_q$ is given by $\widetilde{\Gamma}^i_q = C^{iT}_q C^i_q$ (see Remark 2.5). Problem (7.50) is a quadratic problem with quadratic constraints and can be solved by a solver implementing, for example, an interior-point method.

In comparison, the DASF algorithm will require node q to solve a compressed version of (7.48) given by

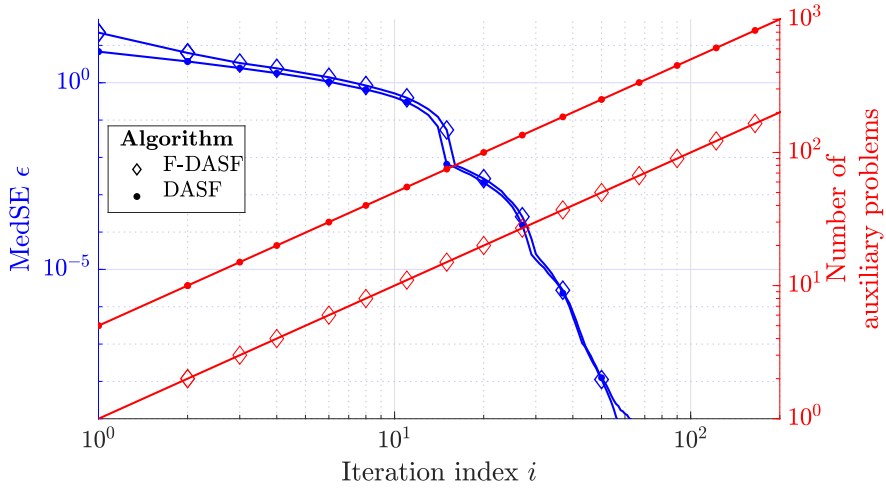
$$\begin{aligned} \underset{\widetilde{\mathbf{x}}_q \in \mathbb{R}^{\widetilde{M}_q}}{\text{minimize}} \quad & \frac{\widetilde{\mathbf{x}}_q^T R^i_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q} \widetilde{\mathbf{x}}_q - 2\widetilde{\mathbf{x}}_q^T \mathbf{r}^i_{\widetilde{\mathbf{y}}_q d} + r_{dd}}{1 + \widetilde{\mathbf{x}}_q^T \widetilde{\Gamma}^i_q \widetilde{\mathbf{x}}_q} \\ \text{subject to} \quad & \|\widetilde{\mathbf{x}}_q^T \widetilde{L}^i_q\|^2 \leq 1, \end{aligned} \tag{7.51}$$

solved at each iteration by applying the Dinkelbach algorithm, i.e., by solving the auxiliary problem (7.50) multiple times.

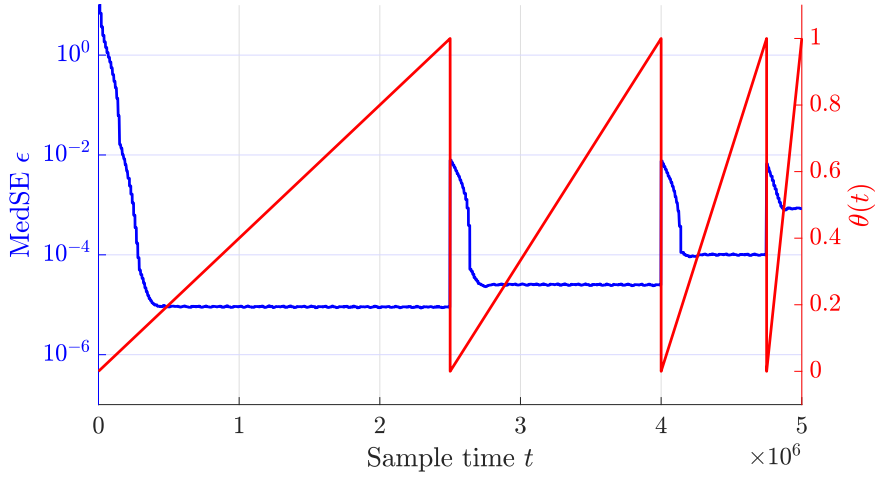
The stopping criterion we use for the Dinkelbach algorithm in each iteration of DASF is a minimum threshold of 10^{-10} on the norm of two consecutive $\widetilde{\mathbf{x}}_q$'s. Figure 7.1a shows the comparison of these performance metrics for a network with $K = 15$ nodes, each with $M_k = 5$ channels. We see that using the DASF algorithm to solve the RTLS problem (7.48) in a distributed fashion requires a significantly higher number of computations compared to solving (7.48) using the F-DASF algorithm. In particular, the DASF algorithm requires solving an average value (over iterations of median values) of 5 times more auxiliary problems per signal batch compared to the F-DASF, while still achieving an equivalent convergence speed.

Let us now consider the case where $\mathbf{p}$ changes at each time instant t, implying that the stationarity assumption on $\mathbf{y}$ does not hold anymore. We have $\mathbf{p}(t) = \mathbf{p}_0 \cdot (1 - \theta(t)) + (\mathbf{p}_0 + \Delta) \cdot \theta(t)$, where θ is represented in Figure 7.1b and the entries of $\mathbf{p}_0$ and Δ are drawn from $\mathcal{N}(0, 1)$ and $\mathcal{N}(0, 0.01)$ respectively. The MedSE ϵ is then given by

$$\epsilon(i) = \text{median}\left(\frac{\|\mathbf{x}^i - \mathbf{x}^{*i}\|^2}{\|\mathbf{x}^{*i}\|^2} \right), \tag{7.52}$$



(a) MedSE ϵ and cumulative computational cost across iterations of the DASF and the proposed F-DASF algorithm in a stationary setting.



(b) MedSE ϵ and parameter θ versus sample time t in an adaptive setting for the F-DASF algorithm. The relationship between the time t and the iterations i is given by $i = \lfloor t/N \rfloor$.

Figure 7.1: Convergence comparison between the F-DASF and DASF algorithms for the RTLS problem (7.48).

with $i = \lfloor t/N \rfloor$, and where the median is taken over 100 Monte Carlo runs. The solution $\mathbf{x}^{*i}$ of (7.48) now depends on the iteration i, where the covariance

quantities of the problem are estimated as

$$\begin{aligned}\widehat{R}^i_{\mathbf{yy}} &= \frac{1}{N}\sum_{\tau=iN}^{iN+N-1}\mathbf{y}(\tau)\mathbf{y}^T(\tau),\\ \widehat{\mathbf{r}}^i_{\mathbf{y}d} &= \frac{1}{N}\sum_{\tau=iN}^{iN+N-1}\mathbf{y}(\tau)d(\tau)\end{aligned} \tag{7.53}$$

(note that r_{dd} is constant in time as in the stationary setting since the signal d is not dependent on $\mathbf{p}$). Figure 7.1b shows the MedSE as a function of sample time t, where we see that the F-DASF algorithm is able to track slow changes in the signal statistics and correct abrupt ones, shown by sudden increase followed by a gradual decrease in MedSE values, highlighting its adaptive properties. Note that the algorithm reaches a MedSE floor, rather than converging to 0 due to the fact that the target $\mathbf{x}^*$ changes at each iteration. The faster the rate of change in statistics, the higher the value of the MedSE floor.

7.6.2 A problem with a non-compact constraint set

To demonstrate the generic nature of the F-DASF algorithm, let us consider the following arbitrary toy problem

$$\begin{aligned}\underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \frac{\mathbb{E}[\|X^T\mathbf{y}(t)\|^2]+\operatorname{tr}(X^TA)}{\operatorname{tr}(X^TB)+c} = \frac{\operatorname{tr}(X^TR_{\mathbf{yy}}X)+\operatorname{tr}(X^TA)}{\operatorname{tr}(X^TB)+c}\\ \text{subject to} \quad & \operatorname{tr}(X^TB)+c>0,\end{aligned} \tag{7.54}$$

where $R_{\mathbf{yy}} = E[\mathbf{y}(t)\mathbf{y}^T(t)]$ is the covariance matrix of $\mathbf{y}$, assumed to be positive definite. In this example, every entry of the matrices $A \in \mathbb{R}^{M\times Q}$ and $B \in \mathbb{R}^{M\times Q}$ have been independently drawn from $\mathcal{N}(0,1)$, while c is taken such that the problem is feasible (see Appendix B.9 for further details). It can be shown that the solution of Problem (7.54) has the form (see Appendix B.9)

$$X^* = -\frac{1}{2}R^{-1}_{\mathbf{yy}}(A+\mu\cdot B), \tag{7.55}$$

where μ is a scalar depending on $R_{\mathbf{yy}}$, A, B, and c. Note that the constraint $\operatorname{tr}(X^TB)+c>0$ enforces the requirement $f_2(X)>0$, but does not constitute a compact set since it is open and unbounded, implying that neither the Dinkelbach procedure nor the F-DASF algorithm have a guarantee of convergence to an optimal point. Therefore, the convergence of the F-DASF (and DASF) algorithm will be assessed by comparing X^i's to X^* given in (7.55).

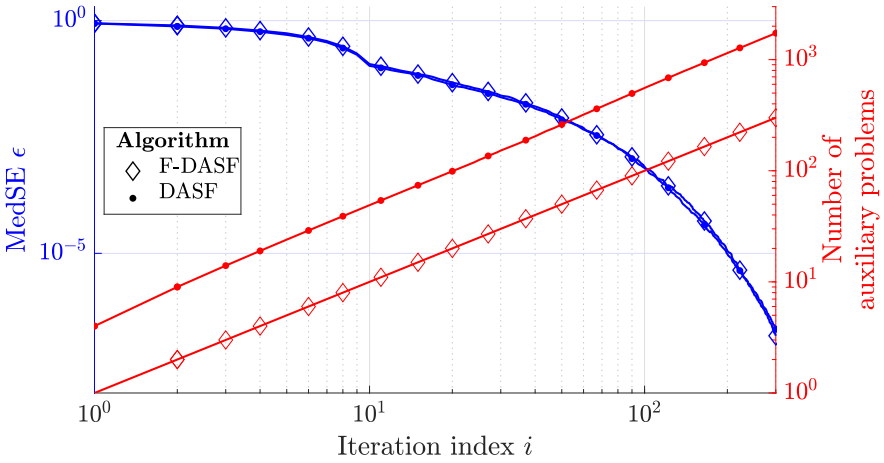


Figure 7.2: Convergence and cumulative computational cost comparison between the proposed F-DASF algorithm and the DASF algorithm when solving Problem (7.54).

For a given ρ, the auxiliary problem of (7.54) can be written as

$$\begin{aligned} &\underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad \operatorname{tr}(X^T R_{\mathbf{yy}} X) + \operatorname{tr}(X^T A) - \rho\cdot(\operatorname{tr}(X^T B) + c) \\ &\text{subject to} \quad \operatorname{tr}(X^T B) + c > 0. \end{aligned} \tag{7.56}$$

If $R_{\mathbf{yy}}$ is positive definite, problem (7.56) is convex and has a unique solution given by

$$X_\rho = \frac{1}{2} R_{\mathbf{yy}}^{-1}(\rho\cdot B - A). \tag{7.57}$$

At each iteration i of the F-DASF algorithm, the updating node q solves its local auxiliary problem (7.26) given by

$$\begin{aligned} &\underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \quad \operatorname{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q) + \operatorname{tr}(\widetilde{X}_q^T \widetilde{A}_q^i) - \rho^i\cdot(\operatorname{tr}(\widetilde{X}_q^T \widetilde{B}_q^i) + c) \\ &\text{subject to} \quad \operatorname{tr}(\widetilde{X}_q^T \widetilde{B}_q^i) + c > 0, \end{aligned} \tag{7.58}$$

where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} = E[\widetilde{\mathbf{y}}_q^i(t)\widetilde{\mathbf{y}}_q^{iT}(t)]$ is the covariance matrix of the locally available stochastic signal $\widetilde{\mathbf{y}}_q^i$ at node q defined in (7.21), while $\widetilde{A}_q^i$ and $\widetilde{B}_q^i$ are the locally available deterministic matrices, as defined in (7.22). The solution $\widetilde{X}_q^*$ of (7.58) is obtained by replacing $(\rho, R_{\mathbf{yy}}, A, B)$ by $(\rho^i, R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, \widetilde{A}_q^i, \widetilde{B}_q^i)$ in (7.57), since

both (7.56) and (7.58) have the same form. In contrast, the DASF algorithm will solve multiple problems (7.58) at each iteration i and node q as it will apply Dinkelbach's procedure on a compressed version of (7.54). Figure 7.2 shows a comparison of both the MedSE values and the cumulative computational cost between the proposed F-DASF algorithm and the existing DASF method. Despite the fact that Problem (7.54) does not have a compact set, we see that convergence can still be achieved to the optimal solution X^* given in (7.55), showing that the F-DASF algorithm can still be used to solve problems with non-compact constraint sets. The F-DASF algorithm solves an average value (over iterations of median values) of 5.77 times fewer auxiliary problems than DASF for the case of Problem (7.54), while again obtaining similar convergence results.

7.7 Conclusion

In this chapter, we have proposed the F-DASF algorithm which exploits the structure and properties of a fractional programming method to significantly reduce the computational cost of the DASF method applied to problems with fractional objectives. In particular, the DASF method requires solving a fractional program at each iteration while the F-DASF method only requires a partial solution, implying significantly fewer computations. Despite this reduced number of computations, the proposed method is shown to converge under similar assumptions as the DASF algorithm, and at similar convergence rates. The results obtained in this chapter also show that partial solutions can be sufficient for the DASF method to converge and open the way for further studies on other families of problems fitting the DASF framework where computational cost reductions can be obtained by solving optimization problems only partially at each iteration.

Appendices

7.A Fixed points are KKT points under Condition 4a

This section provides a proof for Theorem 7.4.

We will first show that a fixed point of the F-DASF algorithm is a KKT point of the auxiliary problem (7.4). Using this result, we will derive the steps to show that such a point is also a KKT point of the global problem (7.1).

The KKT conditions of the auxiliary problem (7.4) can be written as:

$$\nabla_X \mathcal{L}_\rho(X, \rho, \boldsymbol{\lambda}) = 0, \tag{7.59}$$

$$h_j(X) \leq 0 \ \ \forall j \in \mathcal{J}_I,\ h_j(X) = 0 \ \ \forall j \in \mathcal{J}_E, \tag{7.60}$$

$$\lambda_j \geq 0 \ \ \forall j \in \mathcal{J}_I, \tag{7.61}$$

$$\lambda_j h_j(X) = 0 \ \ \forall j \in \mathcal{J}_I, \tag{7.62}$$

where

$$\mathcal{L}_\rho(X, \rho, \boldsymbol{\lambda}) \triangleq f(X, \rho) + \sum_{j \in \mathcal{J}} \lambda_j h_j(X) \tag{7.63}$$

is the Lagrangian of (7.4). We use $\boldsymbol{\lambda}$ in bold as a shorthand notation for the set of all Lagrange multipliers $\lambda_j \in \mathbb{R}$ corresponding to the constraint h_j.

At iteration i, the updating node q solves the local auxiliary problem (7.37) which has the following Lagrangian

$$\widetilde{\mathcal{L}}_\rho(\widetilde{X}_q, \rho^i, \widetilde{\boldsymbol{\lambda}}(q)) \triangleq f(C_q^i \widetilde{X}_q, \rho^i) + \sum_{j \in \mathcal{J}} \lambda_j(q) h_j(C_q^i \widetilde{X}_q), \tag{7.64}$$

where the $\lambda_j(q)$'s are the Lagrange multipliers at updating node q and iteration i corresponding to the local problem (7.37) and $\widetilde{\boldsymbol{\lambda}}(q)$ denotes their collection. Since $\widetilde{X}_q^*$ obtained through the iterations of the F-DASF algorithm solves the local problem (7.37) at node q and iteration i, it must satisfy the KKT conditions of the local problem. In particular, the stationarity condition is given as

$$\nabla_{\widetilde{X}_q} \widetilde{\mathcal{L}}_\rho(\widetilde{X}_q^*, \rho^i, \widetilde{\boldsymbol{\lambda}}(q)) = 0. \tag{7.65}$$

From the parameterization $X = C_q^i \widetilde{X}_q$ in (7.35), we have

$$\widetilde{\mathcal{L}}_\rho(\widetilde{X}_q, \rho^i, \widetilde{\boldsymbol{\lambda}}(q)) = \mathcal{L}_\rho(C_q^i \widetilde{X}_q, \rho^i, \widetilde{\boldsymbol{\lambda}}(q)), \tag{7.66}$$

allowing us to apply the chain rule on (7.65) to obtain

$$C_q^{iT}\nabla_X \mathcal{L}_\rho(C_q^i\widetilde{X}_q^*, \rho^i, \widetilde{\boldsymbol{\lambda}}(q)) = 0. \tag{7.67}$$

Using (7.63), and the parameterized notation in (7.37), we find the KKT conditions of the local problem:

$$C_q^{iT}\nabla_X \left[f\left(C_q^i\widetilde{X}_q^*, \rho^i\right) + \sum_{j\in\mathcal{J}} \lambda_j(q) h_j\left(C_q^i\widetilde{X}_q^*\right)\right] = 0, \tag{7.68}$$

$$h_j\left(C_q^i\widetilde{X}_q^*\right) \leq 0 \;\; \forall j \in \mathcal{J}_I,\; h_j\left(C_q^i\widetilde{X}_q^*\right) = 0 \;\; \forall j \in \mathcal{J}_E, \tag{7.69}$$

$$\lambda_j(q) \geq 0 \;\; \forall j \in \mathcal{J}_I, \tag{7.70}$$

$$\lambda_j(q) h_j\left(C_q^i\widetilde{X}_q^*\right) = 0 \;\; \forall j \in \mathcal{J}_I, \tag{7.71}$$

satisfied by $X^{i+1} = C_q^i\widetilde{X}_q^*$ and its corresponding set of Lagrange multipliers $\widetilde{\boldsymbol{\lambda}}(q)$. Our aim is now to show that at a fixed point of the F-DASF algorithm, i.e., a point such that $X^{i+1} = X^i = \overline{X}$, the KKT conditions (7.68)-(7.71) of the local problem are equivalent to the KKT conditions (7.59)-(7.62) of the global problem.

Let us first look at the local stationarity condition under the fixed point assumption $X^{i+1} = X^i = \overline{X}$, which allows us to replace $C_q^i\widetilde{X}_q^* = X^{i+1}$ with $C_q^i\widetilde{X}_q^i = X^i = \overline{X}$, also implying $\rho^{i+1} = \rho^i = \bar{\rho} = r(\overline{X})$. Making these changes in (7.68) result in

$$C_q^{iT}\nabla_X \left[f(\overline{X}, \bar{\rho}) + \sum_{j\in\mathcal{J}} \lambda_j(q) h_j(\overline{X})\right] = 0. \tag{7.72}$$

From the structure of C_q^i as displayed in (7.33), we observe that the first M_q rows of (7.72) will be equal to

$$\nabla_{X_q} f(\overline{X}, \bar{\rho}) + \sum_{j\in\mathcal{J}} \lambda_j(q)\nabla_{X_q} h_j(\overline{X}) = 0. \tag{7.73}$$

Additionally, the fixed point assumption is independent of the updating node q, therefore, (7.73) holds for any node q. Stacking the variations of (7.73) for each node q then leads to

$$\begin{bmatrix} \nabla_{X_1} f(\overline{X}, \bar{\rho}) \\ \vdots \\ \nabla_{X_K} f(\overline{X}, \bar{\rho}) \end{bmatrix} = \nabla_X f(\overline{X}, \bar{\rho}) = - \begin{bmatrix} \sum_{j\in\mathcal{J}} \lambda_j(1)\nabla_{X_1} h_j(\overline{X}) \\ \vdots \\ \sum_{j\in\mathcal{J}} \lambda_j(K)\nabla_{X_K} h_j(\overline{X}) \end{bmatrix}. \tag{7.74}$$

Let us now return to (7.72) and multiply it by $\widetilde{X}_q^{iT}$ from the left, where $\widetilde{X}_q^i$ is defined in (7.23). From the fact that $C_q^i \widetilde{X}_q^i = X^i = \overline{X}$, we can write

$$\overline{X}^T \nabla_X f(\overline{X}, \bar{\rho}) = -\sum_{j \in \mathcal{J}} \lambda_j(q) \overline{X}^T \nabla_X h_j(\overline{X}). \tag{7.75}$$

From the linear independence of the set $\{\overline{X}^T \nabla_X h_j(\overline{X})\}_j$ implied by Condition 4a, we obtain the result that the Lagrange multipliers $\{\lambda_j(q)\}_j$ satisfying (7.75) are unique. Note that the left-hand side of (7.75) does not depend on the node q, therefore the multipliers are the same for every node, i.e., $\lambda_j(q) = \lambda_j$ for any node q. This result can be used to re-write (7.74) as

$$\nabla_X f(\overline{X}, \bar{\rho}) = -\sum_{j \in \mathcal{J}} \lambda_j \nabla_X h_j(\overline{X}). \tag{7.76}$$

Therefore, a fixed point $\overline{X}$ satisfies the global stationarity condition (7.59) of the auxiliary problem (7.4) with corresponding Lagrange multipliers $\{\lambda_j\}_j$.

We now look at the three other KKT conditions. Note that the local primal feasibility condition (7.69) is satisfied globally, since (7.69) and (7.60) are the same, as was already shown in (7.36) for any point, so it must also hold for a fixed point. Additionally, $(\overline{X}, \{\lambda_j\}_j)$ satisfies the local versions of both the dual feasibility (7.70) and the complementary slackness condition (7.71), where we replace $C_q^i \widetilde{X}_q^* = X^{i+1}$ by $\overline{X}$ from the fixed point assumption, and $\lambda_j(q)$'s by λ_j's for all $j \in \mathcal{J}_I$. Therefore, $(\overline{X}, \{\lambda_j\}_j)$ also satisfies their global counterparts (7.61) and (7.62). The pair $(\overline{X}, \{\lambda_j\}_j)$ hence satisfies all the KKT conditions of the auxiliary problem (7.4).

We now proceed with proving that $\overline{X}$ also satisfies the KKT conditions of the original fractional program (7.1), of which the KKT conditions are given by

$$\nabla_X \mathcal{L}(X, \boldsymbol{\mu}) = 0, \tag{7.77}$$

$$h_j(X) \leq 0 \ \ \forall j \in \mathcal{J}_I,\ h_j(X) = 0 \ \ \forall j \in \mathcal{J}_E, \tag{7.78}$$

$$\mu_j \geq 0 \ \ \forall j \in \mathcal{J}_I, \tag{7.79}$$

$$\mu_j h_j(X) = 0 \ \ \forall j \in \mathcal{J}_I, \tag{7.80}$$

where

$$\mathcal{L}(X, \boldsymbol{\mu}) = r(X) + \sum_{j \in \mathcal{J}} \mu_j h_j(X) \tag{7.81}$$

is the Lagrangian of Problem (7.1). The gradient of r with respect to X can be shown to be equal to (see Example A.5 in Appendix A)

$$\nabla_X r(X) = \frac{1}{f_2(X)} \left(\nabla_X f_1(X) - r(X) \nabla_X f_2(X) \right). \tag{7.82}$$

By plugging in $\overline{X}$ and using the fact that $r(\overline{X}) = \bar{\rho}$, we obtain

$$\nabla_X r(\overline{X}) = \frac{1}{f_2(\overline{X})} \nabla_X f(\overline{X}, \bar{\rho}), \tag{7.83}$$

and by substituting (7.76) in this equation, we eventually find

$$\nabla_X r(\overline{X}) = -\sum_{j \in \mathcal{J}} \frac{\lambda_j}{f_2(\overline{X})} \nabla_X h_j(\overline{X}). \tag{7.84}$$

Then, taking $\mu_j \triangleq \lambda_j / f_2(\overline{X})$ for every $j \in \mathcal{J}$, we observe that we satisfy the stationarity condition (7.77). Additionally, the primal feasibility condition (7.78) is automatically satisfied for $\overline{X}$ as it is identical to (7.60). Finally, since $f_2(X) > 0$ for any $X \in \mathcal{S}$, the multipliers $\mu_j = \lambda_j / f_2(\overline{X})$ satisfy (7.79) and (7.80) from the fact that λ_j's satisfy (7.61) and (7.62). The pair $(\overline{X}, \{\mu_j\}_j)$ therefore satisfies the KKT conditions of the global problem (7.1). □

Part III

Conclusion

8 Conclusion

This chapter provides a summary of the main contributions of this work and discusses various ideas for future research.

8.1 Main contributions

8.1.1 Part I

The first part of this thesis introduces a unified framework named DASF for solving spatial filtering and signal fusion problems in a distributed and adaptive fashion over networks such as WSNs. The DASF framework reduces the bandwidth and energy costs required from the sensor nodes deployed in WSN settings, which typically have limited battery life, processing power, and communication capabilities. It has the purpose of unifying the approach used in certain existing distributed algorithms for specific spatial filtering problems to create a pipeline of steps to follow for generalizing this approach to new problems with theoretical guarantees of convergence to an optimal solution.

Chapter 2 defined the family of problems that fit the framework and derived the steps of the DASF algorithm. We have seen that a large portion of commonly encountered problems in spatial filtering fit the framework, such as MMSE, PCA, GEVD, CCA, TRO and LCMV to name a few. The DASF algorithm has the interesting property that it requires the nodes in the network to create and solve a compressed version of the network-wide problem, therefore implying that both problems have the same structure, such that the same solver can be used. We have demonstrated initial convergence results in experiments where the DASF algorithm was used to solve various spatial filtering problems. The parameters affecting the convergence speed were also discussed, such as the

network topology, the number of nodes in the network and the compression factor applied to the data transmitted by the nodes. Since the DASF algorithm uses new batches of signal samples measured at the network nodes at each iteration, it can track changes in the statistical properties of these signals. These adaptive properties were presented in simulations, where we observed that the changes in the statistical properties of the signals in the network can be tracked by the DASF algorithm as long as they are not too fast.

A rigorous convergence analysis was then provided in Chapter 3, where technical conditions for convergence were defined, and convergence theorems were stated along with their proof. An important result we have found is that the number of constraints of the problem cannot exceed a certain threshold, which can otherwise lead to convergence problems. As long as this property holds, we have seen that the remaining technical conditions were mild and often satisfied in practical scenarios, as verified in various examples provided. Fixes for cases where some of these conditions are not satisfied were also discussed.

8.1.2 Part II

In the second part of this work, we have provided extensions of the DASF algorithm and various improvements to make it more efficient in certain aspects.

In Chapter 4, we have proposed a communication-efficient scheme that improves the tracking properties of the DASF algorithm in settings where the statistical properties of the signals measured over the network change rapidly. Although we have only considered fully-connected network scenarios, extensions are possible for other network topologies (see [132, Chapter 3.5.2]). Instead of using a new batch of signal samples at each iteration as would be so in the case of the original DASF algorithm, we re-use the same batch a certain number of times. We showed that when this number is equal to the number of nodes in the network K, we obtain a tracking speed K times larger than the original DASF algorithm, while only doubling the bandwidth required. Simulation results validated our claims.

An extension of the DASF algorithm to node-specific problems was presented in Chapter 5, i.e., to settings where each node in the network has a different optimization problem to solve that depends on the network-wide data. Under a certain assumption on the relationship between the problems at each node, we were able to follow similar steps as to the DASF framework to propose the DANSF algorithm for node-specific problems. We have provided certain

convergence guarantees of the DANSF algorithm, which are similar to the ones of the original DASF algorithm, and validated these in experimental settings.

In Chapter 6, we have studied in detail the TRO problem, typically used in dimensionality reduction problems as an alternative to the GEVD. However, solving the TRO problem in a WSN setting using the DASF algorithm requires each node to solve an iterative procedure, making it computationally costly. We have therefore proposed an approach that interleaves the steps of this iterative procedure with the ones of the DASF algorithm. The resulting DTRO algorithm applies a single iteration of this iterative procedure at each node, which was shown to be sufficient for convergence. Comparisons of the DTRO algorithm, with the DASF algorithm showed that the convergence speed is not affected, while the latter requires a significantly larger amount of computations.

In Chapter 7, we have extended the results of Chapter 6 for general fractional programs, i.e., optimization problems with an objective function written as the quotient of two continuous functions. We have interleaved the steps of the DASF algorithm with the ones of a generic iterative method named Dinkelbach's procedure, used for solving fractional programs, to obtain the F-DASF algorithm. We have noted that the DTRO algorithm is a special case of the F-DASF algorithm. Convergence results and proofs were provided, along with simulations that validated these results. In particular, we have observed that the F-DASF algorithm significantly lowers the computational burden at each node and does so while keeping similar convergence rates compared to the DASF algorithm.

8.2 Future research directions

8.2.1 Further potential extensions

The development of the DASF framework, along with its various extensions, was initiated by identifying recurring patterns between a few problems of interest (and the algorithms to solve them), and asking ourselves what core ingredients these problems (and their algorithms) have in common. In a similar way, it is possible to find other distributed spatial filtering application settings that have their own problem families that can be linked. For example, recent studies have extended the DASF framework to non-smooth problems, e.g., to allow non-smooth regularization terms [169]. On the other hand, taking the DANSF algorithm a step further, a generic method for multi-task WSNs could be an interesting direction to take. As in the case of node-specific problems discussed in Chapter 5, each node has also a different problem to solve in the multi-task scenario, however, the problems are allowed to be of different nature as

well [139]. For example, a distributed algorithm using similar steps as to the DASF framework has been proposed for WSNs where each node solves either an MMSE signal enhancement, an LCMV beamforming, or a direction of arrival estimation problem in a collaborative fashion [170]. We can then ask ourselves if other problems can be included too, and what assumptions they would require.

An important conclusion from Chapter 7 was that, at least for fractional programs, the DASF framework does not need to solve the local problems at each node fully. Only partially solving them by applying a single step of an iterative solver was shown to be sufficient, while significantly lowering the computational costs at the sensor nodes. An important future work would be to investigate if this can be generalized to other types of problems/solvers, as it raises the question of whether computing a solution of the local problems at each node is necessary, or would a partial solution suffice. This would be particularly interesting for the gradient descent algorithm, as it is a well-studied and very commonly used iterative solver, also having the advantage of being computationally efficient. This latter property makes it attractive even for problems that have closed-form solutions, such as the MMSE problem. Indeed, applying a single gradient descent step at each node would be computationally very efficient compared to computing a matrix inverse, which is required for this problem when using the closed-form solution in the DASF framework. Studying gradient descent methods within this context would also allow to expand the fields where the DASF framework could be used, especially in settings where the objective functions have a structure more complicated than traditional spatial filtering tasks, such as in federated learning.

8.2.2 Realistic scenario studies and improvements

In this work, we have mostly focused on the algorithm development aspect of distributed spatial filtering problems while making abstraction of the properties of the signals measured across the network, apart from assumptions such as (short-term) stationarity and ergodicity. Although we have shown in various experimental results that the DASF algorithm and its extensions are able to track changes in the statistics of the signals measured in the network, a formal mathematical analysis quantifying the tracking accuracy has not been proposed. However, being able to quantify how well the DASF framework performs in an adaptive context is important as it would provide insights and theoretical limitations on the accuracy of the estimated filters in realistic scenarios where signals are not stationary. This can be done by defining non-stationary signal models that reflect properties of realistic signals, as in [33,171].

Additionally, we have assumed throughout this text that the sensor nodes are

able to measure a number of signal samples as large as needed to accurately estimate the covariance matrices involved in the problems, as our focus was on studying convergence properties of the DASF framework. However, being able to reduce the number of samples is important for adaptivity, as the changes in signal statistics would be represented better in smaller batches of signals. Furthermore, the amount of time it takes to gather large amounts of signal samples can create significant delays in the network, which would therefore be reduced if the nodes were to collect fewer samples per batch. Note that estimating statistical quantities of signals using fewer samples also implies a decrease in the accuracy of their estimators, therefore creating a tradeoff between adaptivity and accuracy. Results on computing accurate estimators of covariance matrices using small amounts of samples can be found in the field of random matrix theory, which has found applications in spatial filtering and signal processing [172–174] and can therefore be an interesting direction to study for the DASF framework. Another potential direction would be to study stochastic gradient descent (SGD) methods in the context of the DASF framework, i.e., when each node implements the SGD algorithm to solve its local problem. Originally, SGD only requires a single signal sample, while faster convergence results can be obtained from so-called Batch-SGD methods, which use multiple samples. SGD-based approaches have been studied in distributed spatial filtering contexts for PCA [175] and beamforming [90] but have not yet been generalized to other problems or Batch-SGD methods.

As a final note, throughout this work, theoretical convergence guarantees of the DASF algorithm have been provided and validated in multiple simulations and on a large number of different examples and settings, demonstrating the versatility of the proposed framework. Yet the data that was used in these experiments was artificially generated using models that imitate realistic signals. It should be expected that certain modifications will have to be done for the DASF framework to perform well when using real data for the application of interest. The modifications improving the tracking and adaptivity properties of the DASF framework discussed in Chapter 4 are first steps towards this goal, where the (non-)stationarity assumptions were made stricter to simulate realistic scenarios. Implementing the DASF algorithm on real testing systems is therefore an important next step, allowing to highlight potential limitations the framework may still have in practical settings. Tackling these would in turn result in a more robust algorithm, ready to be deployed in practical applications it was intended to be used in.

Appendices

A | Optimization problems with matrix variables

This chapter is intended to give insights and technical tools to compute gradients of functions taking matrix arguments, set up the Lagrangian of an optimization problem with matrix variables, and derive its stationarity condition.

Given an optimization problem, deriving the equation for its stationarity condition is an important step for describing, and in various cases computing, its solution(s). Optimization problems with matrix variables are commonly encountered in various engineering applications, however, they are often not treated specifically in technical texts, where it is in general common to consider vector-valued arguments or more abstract objects. Although it is possible to extend techniques used in the vector case to the matrix one, particular care should be given as this can lead to erroneous definitions and therefore results [176]. We therefore aim to present the steps to derive the stationarity condition of an optimization problem with matrix variables efficiently and concisely as a reference for readers.

A.1 The gradient of a function with matrix variables

We consider differentiable functions $f : \mathcal{S} \to \mathbb{R}$, taking matrix-valued arguments $X \in \mathcal{S} \subseteq \mathbb{R}^{M\times Q}$. An example of such functions would be

$$f(X) = \operatorname{tr}(X^T AX) \tag{A.1}$$

over $\mathcal{S} = \mathbb{R}^{M\times Q}$, where $A \in \mathbb{R}^{M\times M}$ is a constant matrix. We define the gradient of f with respect to X as the $M \times Q$ matrix with entries given by

$$[\nabla_X f(X)]_{kl} = \frac{\partial}{\partial X_{kl}} f(X), \tag{A.2}$$

where $k \in \{1, \dots, M\}$ and $l \in \{1, \dots, Q\}$. Although obtaining an expression for the gradient of f is possible by computing each component separately using (A.2), it is often not practical to do so. Indeed, this technique can often lead to cumbersome notation and indexing and is error-prone when done by hand, as shown in the following example.

Example A.1. Let us consider the function f given in (A.1). We will compute its gradient based on (A.2). For this, let us write f as the following sum

$$f(X) = \operatorname{tr}(X^T AX) = \sum_{i=1}^{M}\sum_{j=1}^{Q}\sum_{m=1}^{M} X_{ij}A_{im}X_{mj}. \tag{A.3}$$

The partial derivatives of the product $X_{ij}X_{mj}$ can be written as

$$\frac{\partial}{\partial X_{kl}} X_{ij}X_{mj} = \begin{cases} 0 & \text{if } k \notin \{i,m\} \text{ or } l \neq j, \\ X_{il} & \text{if } k = m \neq i \text{ and } l = j, \\ X_{ml} & \text{if } k = i \neq m \text{ and } l = j, \\ 2X_{kl} & \text{if } k = i = m \text{ and } l = j, \end{cases} \tag{A.4}$$

which leads to the following expression:

$$\begin{aligned} \frac{\partial}{\partial X_{kl}} f(X) = \sum_{i=1}^{M}\sum_{j=1}^{Q}\sum_{m=1}^{M} A_{im}\big(&X_{il}\mathbb{1}\{k = m \neq i, l = j\} \\ &+X_{ml}\mathbb{1}\{k = i \neq m, l = j\} \\ &+2X_{kl}\mathbb{1}\{k = m = i, l = j\}\big), \end{aligned} \tag{A.5}$$

where $\mathbb{1}$ denotes the indicator function. We can simplify the previous equation to obtain:

$$\frac{\partial}{\partial X_{kl}} f(X) = \sum_{m=1}^{M} A_{km}X_{ml} + \sum_{i=1}^{M} A_{ik}X_{il}, \tag{A.6}$$

which, by inspection, results in

$$\nabla_X f(X) = (A + A^T)X. \tag{A.7}$$

From this example, we see how cumbersome the expressions can become when using (A.2) to compute the gradients index-by-index. We will therefore provide the description of an efficient way to compute the gradient of f by using results from calculus on manifolds. We omit various technical details on (smooth) manifolds as this is out of the scope of this chapter and refer readers to [177,178] for an in-depth review. To define the gradient we will use the notion of a differential operator, given as

$$Df(X)[V] = \frac{d}{dt} f(X + tV)|_{t=0}, \tag{A.8}$$

for any "direction" $V \in \mathcal{S}$. When $\mathcal{S}$ is a *Riemannian manifold*[1], the gradient of f and its differential operator have the following relationship:

$$Df(X)[V] = \frac{d}{dt} f(X + tV)|_{t=0} = \langle V, \nabla_X f(X)\rangle_{\mathcal{S}}, \tag{A.9}$$

[1]The spaces we consider in this work are all Riemannian manifolds.

where $\langle \cdot, \cdot \rangle_{\mathcal{S}}$ is an inner product. For the case $\mathcal{S} \subseteq \mathbb{R}^{M \times Q}$, we can take $\langle \cdot, \cdot \rangle_{\mathcal{S}}$ to be the standard inner product defined as:

$$\langle V, \nabla_X f(X) \rangle_{\mathcal{S}} = \operatorname{tr}(V^T \nabla_X f(X)). \tag{A.10}$$

Note the similarity of (A.10) with the definition of a directional derivative for the case where X is a vector.

Combining (A.9) with (A.10), we can extract an expression for the gradient of f by only computing a scalar derivative:

$$\operatorname{tr}(V^T \nabla_X f(X)) = \frac{d}{dt} f(X + tV)|_{t=0}. \tag{A.11}$$

We reiterate that some technical conditions are required for (A.11) to hold, which we have omitted for conciseness. However, the gradient of all functions we consider in this text can be computed using this formula.

Example A.2. Consider the function f such that $f(X) = \operatorname{tr}(A^T X)$ with $A, X \in \mathbb{R}^{M \times Q}$. Then

$$f(X + tV) = \operatorname{tr}(A^T X) + t \cdot \operatorname{tr}(A^T V), \tag{A.12}$$

leading to

$$\frac{d}{dt} f(X + tV)|_{t=0} = \operatorname{tr}(A^T V) = \operatorname{tr}(V^T A). \tag{A.13}$$

From (A.11), we have $\nabla_X f(X) = A$.

Example A.3. For the function $f : f(X) = \operatorname{tr}(X^T A X)$ with $X \in \mathbb{R}^{M \times Q}$ and $A \in \mathbb{R}^{M \times M}$, we have:

$$f(X + tV) = \operatorname{tr}(X^T A X) + t \cdot \operatorname{tr}(V^T A X) + t \cdot \operatorname{tr}(X^T A V) + t^2 \cdot \operatorname{tr}(V^T A V), \tag{A.14}$$

which results in

$$\frac{d}{dt} f(X + tV) = \operatorname{tr}(V^T A X) + \operatorname{tr}(X^T A V) + 2t \cdot \operatorname{tr}(V^T A V) \tag{A.15}$$

and finally

$$\frac{d}{dt} f(X + tV)|_{t=0} = \operatorname{tr}(V^T A X) + \operatorname{tr}(X^T A V) = \operatorname{tr}(V^T (A + A^T) X). \tag{A.16}$$

Using (A.11), we find that $\nabla_X f(X) = (A + A^T)X$. Compared to Example A.1, the result is obtained in a much more concise fashion.

Example A.4. Let us consider the function $f : \mathbb{R}^{M\times Q} \to \mathbb{R}$ taking $X \in \mathbb{R}^{M\times Q}$ as an argument and the parameterization $X = C\widetilde{X}$ for a certain $\widetilde{X} \in \mathbb{R}^{\widetilde{M}\times Q}$ and $C \in \mathbb{R}^{M\times\widetilde{M}}$. Additionally, we define the function $\widetilde{f} : \mathbb{R}^{\widetilde{M}\times Q} \to \mathbb{R}$ such that $\widetilde{f}(\widetilde{X}) = f(C\widetilde{X})$. In various parts of this text, we are interested in expressing $\nabla_{\widetilde{X}}\widetilde{f}(\widetilde{X})$ in terms of $\nabla_X f(X)$, which can be obtained using the chain rule. Here, we derive this result using (A.9).

For any direction $\widetilde{V} \in \mathbb{R}^{\widetilde{M}\times Q}$, we have $\widetilde{f}(\widetilde{X} + t\widetilde{V}) = f(C\widetilde{X} + tC\widetilde{V})$ and

$$\begin{aligned}\langle \widetilde{V}, \nabla_{\widetilde{X}}\widetilde{f}(\widetilde{X})\rangle &= \frac{d}{dt}\widetilde{f}(\widetilde{X} + t\widetilde{V})|_{t=0} \\ &= \frac{d}{dt} f(C\widetilde{X} + tC\widetilde{V})|_{t=0} = \langle C\widetilde{V}, \nabla_X f(C\widetilde{X})\rangle.\end{aligned} \tag{A.17}$$

The last expression is an inner product in $\mathbb{R}^{M\times Q}$, therefore we can write

$$\langle C\widetilde{V}, \nabla_X f(C\widetilde{X})\rangle = \mathrm{tr}(\widetilde{V}^T C^T \nabla_X f(C\widetilde{X})) = \langle \widetilde{V}, C^T \nabla_X f(C\widetilde{X})\rangle, \tag{A.18}$$

resulting in

$$\langle \widetilde{V}, \nabla_{\widetilde{X}}\widetilde{f}(\widetilde{X})\rangle = \langle \widetilde{V}, C^T \nabla_X f(C\widetilde{X})\rangle. \tag{A.19}$$

Since (A.19) holds for any $\widetilde{V} \in \mathbb{R}^{\widetilde{M}\times Q}$, we have

$$\nabla_{\widetilde{X}}\widetilde{f}(\widetilde{X}) = C^T \nabla_X f(C\widetilde{X}). \tag{A.20}$$

Example A.5. Consider the function $r : \mathcal{S} \to \mathbb{R}$ such that $r(X) = \frac{f_1(X)}{f_2(X)}$, with f_1, f_2 differentiable on $\mathcal{S}$ and $f_2(X) \neq 0$ for $X \in \mathcal{S}$. We will show that $\nabla_X r(X) = \frac{1}{f_2(X)}\left(\nabla_X f_1(X) - r(X)\nabla_X f_2(X)\right)$ using (A.9). Let us define the functions g_1 and g_2 such that $g_i(t) = f_i(X + tV)$ and $g_i'(t) = \frac{d}{dt}g_i(t)$ for

$i \in \{1, 2\}$, then:

$$\begin{aligned}\langle V, \nabla_X r(X)\rangle &= \frac{d}{dt} r(X + tV)|_{t=0} \\ &= \frac{d}{dt}\frac{f_1(X+tV)}{f_2(X+tV)}\bigg|_{t=0} = \frac{g_1'(0)g_2(0) - g_2'(0)g_1(0)}{g_2(0)^2}.\end{aligned} \tag{A.21}$$

From the fact that $g_i(0) = f_i(X)$ and

$$g_i'(0) = \frac{d}{dt} g_i(t)|_{t=0} = \frac{d}{dt} f_i(X + tV)|_{t=0} = \langle V, \nabla_X f_i(X)\rangle, \tag{A.22}$$

we find

$$\langle V, \nabla_X r(X)\rangle = \langle V, \nabla_X f_1(X)\rangle \frac{1}{f_2(X)} - \langle V, \nabla_X f_2(X)\rangle \frac{r(X)}{f_2(X)}. \tag{A.23}$$

Therefore, we finally have

$$\nabla_X r(X) = \frac{1}{f_2(X)} \left(\nabla_X f_1(X) - r(X)\nabla_X f_2(X)\right). \tag{A.24}$$

A.2 Optimization over matrix variables

Consider the following optimization problem

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X) \\ \text{subject to} \quad & h_j(X) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & h_j(X) = 0 \ \ \forall j \in \mathcal{J}_E, \end{aligned} \tag{A.25}$$

where f and all h_j's are real, scalar-valued functions, while $\mathcal{J}_I$ and $\mathcal{J}_E$ represent the index set of inequality and equality constraints, respectively. The Lagrangian of (A.25) is then given by

$$\mathcal{L}(X, \boldsymbol{\lambda}) = f(X) + \sum_{j \in \mathcal{J}} \lambda_j h_j(X), \tag{A.26}$$

where $\mathcal{J} = \mathcal{J}_I \cup \mathcal{J}_E$ represents the index sets of all constraints, λ_j's denote the Lagrange multipliers and $\boldsymbol{\lambda}$ is a shorthand notation for the set of Lagrange multipliers. Note that there is a scalar Lagrange multiplier assigned to each

scalar-valued constraint. The Karush-Kuhn-Tucker (KKT) conditions for optimality are then given by

$$\nabla_X \mathcal{L}(X, \boldsymbol{\lambda}) = 0, \tag{A.27}$$

$$h_j(X) \leq 0 \; \forall j \in \mathcal{J}_I, h_j(X) = 0 \; \forall j \in \mathcal{J}_E, \tag{A.28}$$

$$\lambda_j \geq 0 \; \forall j \in \mathcal{J}_I, \tag{A.29}$$

$$\lambda_j h_j(X) = 0 \; \forall j \in \mathcal{J}_I. \tag{A.30}$$

Equations (A.27)-(A.30) are called the stationarity, primal feasibility, dual feasibility, and complementary slackness equations, respectively. To derive the stationarity condition, the gradient of the Lagrangian with respect to X can be extracted by computing (A.11):

$$\begin{aligned} \frac{d}{dt}\mathcal{L}(X + tV, \boldsymbol{\lambda})|_{t=0} &= \mathrm{tr}(V^T \nabla_X \mathcal{L}(X, \boldsymbol{\lambda})) \\ &= \mathrm{tr}(V^T \nabla_X f(X)) + \sum_{j \in \mathcal{J}} \lambda_j \mathrm{tr}(V^T \nabla_X h_j(X)), \end{aligned} \tag{A.31}$$

i.e., by computing the gradient of f and each one of h_j's separately.

In many practical scenarios however, it is common to group multiple scalar constraints into a single equality or inequality constraint using vector or matrix structures, for example having an orthogonality constraint such as $X^T X = I_Q$. Although it is possible to decompose these vector/matrix structured constraints into distinct scalar constraints as in (A.25), it is often not practical to do so (see Example A.1 in Appendix A.1). We therefore present next a method to set up the Lagrangian in such cases in an efficient way, without needing to separate every constraint into a scalar-valued equality/inequality. For this purpose, consider the example problem

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X) \\ \text{subject to} \quad & \mathrm{tr}(X^T A X) = 1, \\ & \mathbf{b}^T X \mathbf{c} = 1, \\ & X\mathbf{d} = \mathbf{1}_M, \\ & X^T X = I_Q, \end{aligned} \tag{A.32}$$

with $A \in \mathbb{R}^{M \times M}$, $\mathbf{b} \in \mathbb{R}^M$, $\mathbf{c}, \mathbf{d} \in \mathbb{R}^Q$ and $\mathbf{1}_M$ denotes the all-one vector of size M. The first and second constraints are scalar-valued, hence we assign

a scalar Lagrange multiplier for each, λ_1 and λ_2, respectively. The third constraint is a vector equality of size M, therefore its corresponding Lagrange multiplier is $\boldsymbol{\lambda}_3 \in \mathbb{R}^M$. Finally, the fourth constraint is a $Q \times Q$ matrix equality, however, there is redundancy between the upper and lower triangular parts, implying $\frac{Q(Q+1)}{2}$ distinct constraints instead of Q^2. Therefore, the corresponding Lagrange multiplier Λ_4 will be a symmetric, $Q \times Q$ matrix. In the general case, if a constraint has the form $H(X) = 0$, where $H : \mathcal{S} \to \mathcal{D}$, its corresponding Lagrange multiplier λ is an element of $\mathcal{D}$ and added to the Lagrangian in the form of an inner product in $\mathcal{D}$, i.e., $\langle \lambda, H(X) \rangle_{\mathcal{D}}$.

The Lagrangian of (A.32) can then be written as

$$\begin{aligned} \mathcal{L}(X, \boldsymbol{\lambda}) = f(X) + \langle \lambda_1, \mathrm{tr}(X^T A X) - 1 \rangle_{\mathbb{R}} + \langle \lambda_2, \mathbf{b}^T X \mathbf{c} - 1 \rangle_{\mathbb{R}} \\ + \langle \boldsymbol{\lambda}_3, X\mathbf{d} - \mathbf{1}_M \rangle_{\mathbb{R}^M} + \langle \Lambda_4, X^T X - I_Q \rangle_{\mathbb{R}^{Q \times Q}} \end{aligned} \tag{A.33}$$

where $\langle u, v \rangle_{\mathbb{R}}$ is the product $u \cdot v$, $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^M}$ is the dot product $\mathbf{u}^T \mathbf{v}$ and $\langle U, V \rangle_{\mathbb{R}^{Q \times Q}}$ corresponds to $\mathrm{tr}(U^T V)$, leading to

$$\begin{aligned} \mathcal{L}(X, \boldsymbol{\lambda}) = f(X) + \lambda_1(\mathrm{tr}(X^T A X) - 1) + \lambda_2(\mathbf{b}^T X \mathbf{c} - 1) \\ + \boldsymbol{\lambda}_3^T (X\mathbf{d} - \mathbf{1}_M) + \mathrm{tr}(\Lambda_4^T (X^T X - I_Q)). \end{aligned} \tag{A.34}$$

The gradient of $\mathcal{L}$ with respect to X can then be extracted by computing (A.11):

$$\begin{aligned} \frac{d}{dt} \mathcal{L}(X + tV, \boldsymbol{\lambda})|_{t=0} = \mathrm{tr}(V^T \nabla_X f(X)) + \lambda_1 \mathrm{tr}(V^T (A + A^T) X) + \lambda_2 \mathrm{tr}(V^T \mathbf{b}\mathbf{c}^T) \\ + \mathrm{tr}(V^T \boldsymbol{\lambda}_3 \mathbf{d}^T) + 2 \cdot \mathrm{tr}(V^T X \Lambda_4), \end{aligned} \tag{A.35}$$

where we have used the facts that $\mathrm{tr}(DEF) = \mathrm{tr}(FDE)$, $\mathrm{tr}(D^T) = \mathrm{tr}(D)$, $a = \mathrm{tr}(a)$ for a scalar a and $\Lambda_4^T = \Lambda_4$ to obtain this equation.

Finally, assuming $\nabla_X f(X)$ is known, the stationarity condition of (A.32) is given as

$$\nabla_X \mathcal{L}(X, \boldsymbol{\lambda}) = \nabla_X f(X) + \lambda_1 (A + A^T) X + \lambda_2 \mathbf{b}\mathbf{c}^T + \boldsymbol{\lambda}_3 \mathbf{d}^T + 2X\Lambda_4 = 0. \tag{A.36}$$

B | Examples of optimization problems with matrix variables

This chapter summarizes various optimization problems encountered throughout this text and derives their optimality conditions and solutions.

B.1 Least squares

Given two matrices A and B, the least squares (LS) problem aims at finding an X such that AX is closest to B in the sense that the distance between the two is minimal. It is often written as

$$\underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \ \|AX-B\|_F^2 = \text{tr}(X^TA^TAX) - 2\cdot\text{tr}(B^TAX) + \text{tr}(B^TB). \quad \text{(B.1)}$$

Note that A^TA is always positive semidefinite, therefore the LS problem is convex. Since (B.1) is unconstrained, its solutions satisfy $\nabla_X\|AX-B\|_F^2 = 0$. From Examples A.2 and A.3 of Appendix A, we have

$$\nabla_X\|AX-B\|_F^2 = 2A^TAX - 2A^TB = 0, \quad \text{(B.2)}$$

or equivalently

$$A^TAX = A^TB, \quad \text{(B.3)}$$

where the last equality is commonly referred to as the normal equations of the problem. If A^TA is invertible, or in other words if A is full column rank, the solution of the LS problem is unique and given by

$$X^* = (A^TA)^{-1}A^TB. \quad \text{(B.4)}$$

In the particular case where A is a square matrix, A being full column rank implies A being invertible, therefore the solution is given by $X^* = A^{-1}B$. If A^TA is not invertible, then the normal equations have infinitely many solutions since $\mathcal{R}(A^TA) = \mathcal{R}(A^T)$, where $\mathcal{R}$ denotes the range space. In that case, selecting an optimal X is generally problem dependent, however, a common approach is to select the minimum-norm X, solution of the problem

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \|X\|_F^2 \\ \text{subject to} \quad & AX = B. \end{aligned} \quad \text{(B.5)}$$

When AA^T is invertible, i.e., A is full row rank, the solution of (B.5) is given by

$$X^* = A^T(AA^T)^{-1}B \quad \text{(B.6)}$$

(the steps to follow to derive this solution is provided in Appendix B.6).

Another common method to solve an LS problem when the solution is not unique, i.e., when A^TA is not invertible, is to add an ℓ_2 regularization term, such that Problem (B.1) becomes

$$\underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \ \|AX-B\|_F^2+\|\Gamma X\|_F^2, \tag{B.7}$$

where Γ is a regularization matrix, chosen such that $A^TA+\Gamma^T\Gamma$ becomes invertible. This problem is often referred to as ridge regression, and its solution is given by

$$X^*=(A^TA+\Gamma^T\Gamma)^{-1}A^TB, \tag{B.8}$$

obtained in a similar fashion as the derivation of the solution of the LS problem.

B.2 Linearly constrained minimum variance

The linearly constrained minimum variance (LCMV) optimization problem is commonly encountered in beamforming and is represented as minimizing a quadratic cost subject to linear constraints:

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \frac{1}{2}\text{tr}(X^TAX) \\ \text{subject to} \quad & X^TB=C, \end{aligned} \tag{B.9}$$

where A is assumed to be symmetric positive semidefinite. The Lagrangian of Problem (B.9) is given by

$$\mathcal{L}(X,\Lambda)=\frac{1}{2}\text{tr}(X^TAX)+\text{tr}(\Lambda^T(X^TB-C)), \tag{B.10}$$

with Λ the matrix of Lagrange multipliers having the same dimensions as C. Following similar steps as in Appendix A.2 of Appendix A, we can show that

$$\nabla_X\text{tr}(\Lambda^T(X^TB-C))=B\Lambda^T, \tag{B.11}$$

such that the stationarity condition of the LCMV problem is

$$\nabla_X\mathcal{L}(X,\Lambda)=0\Rightarrow AX=-B\Lambda^T. \tag{B.12}$$

If A is invertible, we obtain $X=-A^{-1}B\Lambda^T$ and multiplying both sides by B^T from the left results in $B^TX=-B^TA^{-1}B\Lambda^T$. From the constraint $X^TB=C$, we have $C=-\Lambda B^TA^{-1}B$. If $B^TA^{-1}B$ is also invertible, which is guaranteed if B is full column rank we can write

$$\Lambda=-C(B^TA^{-1}B)^{-1}. \tag{B.13}$$

Under these conditions, since $X = -A^{-1}B\Lambda^T$, the solution of (B.9) is unique and given by

$$X^* = A^{-1}B(B^TA^{-1}B)^{-1}C^T. \tag{B.14}$$

B.3 Generalized eigenvalue decomposition

Given a pair of matrices (A, B), $A, B \in \mathbb{R}^{M\times M}$, the generalized eigenvectors (GEVCs) X and eigenvalues (GEVLs) Λ are matrices such that $AX = BX\Lambda$. The matrix X contains the generalized eigenvectors of (A, B) in its columns, while Λ is a diagonal matrix containing the generalized eigenvalues of (A, B) in its diagonal. We will show that the Q GEVCs corresponding to the Q largest GEVLs are solutions of the following optimization problem

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & \operatorname{tr}(X^TAX) \\ \text{subject to} \quad & X^TBX = I_Q, \end{aligned} \tag{B.15}$$

where we assume that A and B are symmetric. Note that (B.15) corresponds to the Principal Component Analysis problem when $B = I_M$, i.e., an eigenvalue decomposition problem. The Lagrangian of (B.15) is given by

$$\mathcal{L}(X, \Lambda) = -\operatorname{tr}(X^TAX) + \operatorname{tr}(\Lambda^T(X^TBX - I_Q)), \tag{B.16}$$

where Λ is the $Q \times Q$ matrix of Lagrange multipliers. Following similar steps as in Example A.3, we can find that

$$\nabla_X\mathcal{L}(X, \Lambda) = -2AX + BX(\Lambda + \Lambda^T). \tag{B.17}$$

Since the constraint $X^TBX = I_Q$ has the same information in its upper and lower triangular part, Λ must be symmetric. The stationarity condition can then be written as

$$AX = BX\Lambda. \tag{B.18}$$

We see that taking X as the matrix containing Q GEVCs of the pair (A, B) in its columns, and selecting Λ to be a diagonal matrix with the corresponding GEVLs gives us a pair (X, Λ) satisfying the stationarity condition of (B.15). Note that (B.18) does not automatically imply that Λ should be diagonal, however, the spectral theorem states that it is diagonalizable since it is symmetric. Then, there exists a diagonal matrix Φ and an invertible matrix P such that $\Lambda = P\Phi P^{-1}$ for which XP and Φ correspond to the GEVCs and GEVLs of the pair (A, B). We can therefore assume without loss of generality that Λ is diagonal (we refer to [179] for further details).

From (B.18), we have $\text{tr}(X^T AX) = \text{tr}(X^T BX\Lambda)$, while the constraint $X^T BX = I_Q$ results in $\text{tr}(X^T AX) = \text{tr}(\Lambda) = \sum_{k=1}^{Q} \lambda_k$, where λ_k's are the GEVLs contained in the diagonal of Λ. Selecting λ_k's to be the Q largest GEVLs of the pair (A, B) would hence maximize the objective.

Therefore, an optimal solution X^* of Problem (B.15) is given by

$$X^* = \text{GEVD}_Q(A, B), \tag{B.19}$$

where $\text{GEVD}_Q(A, B)$ denotes the matrix containing the Q GEVCs corresponding to the Q largest GEVLs of (A, B). Note that X^* is not unique; any matrix obtained by changing the sign of the columns of X^* is still a solution of (B.15). Moreover, if the $Q + 1$ largest GEVLs of the pair (A, B) are not all distinct, additional solutions exist. For example, changing the order of the GEVCs corresponding to GEVLs with algebraic multiplicity greater than 1 in $\text{GEVD}_Q(A, B)$ would then still be a solution of Problem (B.15).

B.4 Trace ratio optimization

The trace ratio optimization (TRO) problem is given by

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{maximize}} \quad & \frac{\text{tr}(X^T AX)}{\text{tr}(X^T BX)} \\ \text{subject to} \quad & X^T X = I_Q, \end{aligned} \tag{B.20}$$

where we consider A to be symmetric and B to be symmetric positive definite. The TRO problem can be interpreted as the extension of the generalized Rayleigh quotient to matrix variables, therefore providing an alternative to the generalized eigenvalue decomposition problem. Its Lagrangian is written as

$$\mathcal{L}(X, \Lambda) = -\frac{\text{tr}(X^T AX)}{\text{tr}(X^T BX)} + \text{tr}(\Lambda^T (X^T X - I_Q)), \tag{B.21}$$

where Λ is a $Q \times Q$ symmetric matrix (see Appendix B.3), and we obtain the stationarity condition

$$\nabla_X \mathcal{L}(X, \Lambda) = -\frac{2}{\text{tr}(X^T BX)} \left(AX - \frac{\text{tr}(X^T AX)}{\text{tr}(X^T BX)} BX \right) + 2X\Lambda = 0, \tag{B.22}$$

by following similar steps as in Examples A.3 and A.5 of Appendix A. From the positive-definiteness of B, we have $\text{tr}(X^T BX) > 0$, and the stationarity

condition of Problem (B.20) can be rewritten as

$$\left(A - \frac{\mathrm{tr}(X^T AX)}{\mathrm{tr}(X^T BX)} B\right) X = X\Lambda. \tag{B.23}$$

Multiplying from the left the previous expression by X^T, using $X^T X = I_Q$, we have

$$X^T \left(A - \frac{\mathrm{tr}(X^T AX)}{\mathrm{tr}(X^T BX)} B\right) X = \Lambda \tag{B.24}$$

such that the stationarity condition can also be written as

$$\nabla_X \mathcal{L}(X, \Lambda) = -\frac{2(I_M - XX^T)}{\mathrm{tr}(X^T BX)} \left(AX - \frac{\mathrm{tr}(X^T AX)}{\mathrm{tr}(X^T BX)} BX\right) = 0. \tag{B.25}$$

On the other hand, note that the stationarity condition (B.23) is akin to an eigenvalue decomposition problem applied to the matrix $A - \frac{\mathrm{tr}(X^T AX)}{\mathrm{tr}(X^T BX)} B$, however, this matrix also depends on X, therefore requiring further analysis. As shown in [143, 144], a solution X^* is given by

$$X^* = \mathrm{EVD}_Q(A - \rho^* B), \tag{B.26}$$

where $\mathrm{EVD}_Q(A - \rho^* B)$ denotes the matrix containing the Q orthonormal eigenvectors corresponding to the Q largest eigenvalues of $A - \rho^* B$, and ρ^* is the minimal value of $\frac{\mathrm{tr}(X^T AX)}{\mathrm{tr}(X^T BX)}$ over the constraint set $X^T X = I_Q$.

As explained in Chapter 6, the TRO problem can be solved by successively solving eigenvalue decompositions of matrices of the form $A - \rho B$.

B.5 Canonical correlation analysis

Canonical correlation analysis (CCA) aims at finding two linear filters that maximize the correlation between two datasets after applying them on the respective dataset. Mathematically, this can be expressed as the following optimization problem:

$$\begin{aligned} \underset{X,W \in \mathbb{R}^{M \times Q}}{\text{maximize}} \quad & \mathrm{tr}(X^T AW) \\ \text{subject to} \quad & X^T BX = I_Q, \\ & W^T CW = I_Q, \end{aligned} \tag{B.27}$$

where the maximization is taken over the pair of variables (X, W). In practice, B and C correspond to the covariance matrix of the first and second datasets, while A is the cross-covariance matrix between both datasets. The Lagrangian of the CCA problem is given by

$$\begin{aligned}\mathcal{L}(X, W, \Lambda, \Phi) = -\mathrm{tr}(X^T A W) + \frac{1}{2}\mathrm{tr}(\Lambda^T (X^T B X - I_Q)) \\ + \frac{1}{2}\mathrm{tr}(\Phi^T (W^T C W - I_Q)),\end{aligned} \tag{B.28}$$

where Φ is a $Q \times Q$ matrix corresponding to the Lagrange multipliers of the constraint $W^T C W = I_Q$. As in Appendix B.3, both Λ and Φ are symmetric. The factor $\frac{1}{2}$ has been added to the last two terms of the sum for practical purposes. The stationarity condition is then given by the following two equations

$$\nabla_X \mathcal{L}(X, W, \Lambda, \Phi) = -AW + BX\Lambda = 0, \tag{B.29}$$

$$\nabla_W \mathcal{L}(X, W, \Lambda, \Phi) = -A^T X + CW\Phi = 0. \tag{B.30}$$

Multiplying (B.29) by X^T from the left, and applying the constraint $X^T B X = I_Q$, we obtain $\Lambda = X^T A W$, while similar steps applied to (B.30) lead to $\Phi = W^T A^T X = \Lambda^T$. Since Λ is symmetric, we have

$$\Phi = \Lambda. \tag{B.31}$$

Assuming B and C to be invertible, (B.29) results in

$$X\Lambda = B^{-1} A W. \tag{B.32}$$

Then, by multiplying (B.30) by Λ from the right, and plugging in (B.31) and (B.32), we obtain

$$A^T X \Lambda = A^T B^{-1} A W = C W \Lambda^2, \tag{B.33}$$

leading to

$$C^{-1} A^T B^{-1} A W = W \Lambda^2. \tag{B.34}$$

We can then take Λ^2 to be the diagonal matrix containing the Q largest eigenvalues (EVLs) of $C^{-1} A^T B^{-1} A$ on its diagonal (see Appendix B.3), while

$$W^* = \mathrm{EVD}_Q(C^{-1} A^T B^{-1} A), \tag{B.35}$$

i.e., W contains in its columns the Q eigenvectors of $C^{-1} A^T B^{-1} A$ corresponding to its Q largest EVLs. If $C^{-1} A^T B^{-1} A$ is full rank, we then have

$$X^* = B^{-1} A W \Lambda^{-1}. \tag{B.36}$$

Alternatively, starting from (B.30) to obtain $W\Lambda = C^{-1}A^TX$ and following analogous steps, we obtain

$$X^* = \mathrm{EVD}_Q(B^{-1}AC^{-1}A^T) \tag{B.37}$$

and

$$W^* = C^{-1}A^TX\Lambda^{-1} \tag{B.38}$$

where Λ^2 is a diagonal matrix containing the Q largest EVLs of $B^{-1}AC^{-1}A^T$ in its diagonal. Note that the EVLs of $B^{-1}AC^{-1}A^T$ and $C^{-1}A^TB^{-1}A$ are equal due to the invariance of eigenvalues under cyclic shifts in products of matrices, hence confirming that $\Lambda = \Phi$. Therefore, X^* in (B.36) and (B.37) both represent the same solution, and similarly for W^* in (B.35) and (B.38). We refer to [23, 24] for further details on the CCA problem.

B.6 Linearly constrained minimum norm

The minimum norm problem with linear equality constraints is given by

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \frac{1}{2}\|CX - D\|_F^2 \\ \text{subject to} \quad & X^TA = B, \end{aligned} \tag{B.39}$$

which generalizes both the LS and the LCMV problems discussed in Appendices B.1 and B.2. The Lagrangian of the problem can be written as

$$\mathcal{L}(X,\Lambda) = \frac{1}{2}\|CX - D\|_F^2 + \mathrm{tr}(\Lambda^T(X^TA - B)), \tag{B.40}$$

where Λ is the matrix of Lagrangian multipliers, which has the same dimensions as B. The stationarity condition of Problem (B.39) is then written as

$$\nabla_X\mathcal{L}(X,\Lambda) = C^TCX - C^TD + A\Lambda^T = 0. \tag{B.41}$$

Equation (B.41), together with the constraint $X^TA = B$ forms the following linear equation

$$\begin{bmatrix} C^TC & A \\ A^T & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ \Lambda^T \end{bmatrix} = \begin{bmatrix} C^TD \\ B^T \end{bmatrix}. \tag{B.42}$$

If the 2×2 block matrix is invertible, we obtain

$$\begin{bmatrix} X \\ \Lambda^T \end{bmatrix} = \begin{bmatrix} C^TC & A \\ A^T & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} C^TD \\ B^T \end{bmatrix}. \tag{B.43}$$

In particular, if C^TC and $A^T(C^TC)^{-1}A$ are invertible, which is satisfied if C and A are full column rank, we have

$$\begin{bmatrix} C^TC & A \\ A^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} G - GA(A^TGA)^{-1}A^TG & GA(A^TGA)^{-1} \\ (A^TGA)^{-1}A^TG & -(A^TGA)^{-1} \end{bmatrix}, \tag{B.44}$$

where $G = (C^TC)^{-1}$. Combining (B.43) and (B.44), the solution of Problem (B.39) is given by

$$X^* = (C^TC)^{-1}\left(C^TD + A(A^T(C^TC)^{-1}A)^{-1}(B^T - A^T(C^TC)^{-1}C^TD)\right). \tag{B.45}$$

Note that we find the LCMV solution (B.14) when $D = 0$, while the solution of Problem (B.5) is obtained if additionally $C = I_M$.

B.7 Tikhonov regularization on the minimum norm

Let us solve the same problem as in the previous section but now with an additional quadratic inequality constraint, written as

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \frac{1}{2}\|CX - D\|_F^2 \\ \text{subject to} \quad & X^T\mathbf{a} = \mathbf{b},\ \|X\|_F^2 \leq \alpha^2, \end{aligned} \tag{B.46}$$

where C is full column rank[1]. Note that we consider in (B.46) a vector-valued linear equality constraint to avoid having too cumbersome equations, but the results can be generalized to the matrix case. We then have

$$\|\mathbf{b}\|^2 = \|X^T\mathbf{a}\|^2 \leq \|X\|_F^2 \cdot \|\mathbf{a}\|^2 \leq \alpha^2 \cdot \|\mathbf{a}\|^2, \tag{B.47}$$

where the first equality comes from taking the norm of the equality constraint of (B.46), the first inequality from the Cauchy-Schwarz inequality, while the last from the inequality constraint of the problem. This leads to the following condition for the feasibility of the problem

$$\alpha^2 \geq \frac{\|\mathbf{b}\|^2}{\|\mathbf{a}\|^2}. \tag{B.48}$$

Additionally, if this condition is met with equality, then the Cauchy-Schwarz inequality holds with equality: $\alpha^2 = \|\mathbf{b}\|^2/\|\mathbf{a}\|^2$, which can happen if only if the

[1]Further details on this problem and its solution can be found in this link.

columns of X are multiples of $\mathbf{a}$, i.e., there exists a vector $\mathbf{v}$ such that $X = \mathbf{a}\mathbf{v}^T$. Plugging this into the equality constraint, we obtain the only feasible point

$$X^* = \frac{\mathbf{a}\mathbf{b}^T}{\|\mathbf{a}\|^2}. \tag{B.49}$$

We now consider the case $\alpha^2 > \|\mathbf{b}\|^2/\|\mathbf{a}\|^2$. The Lagrangian of the problem is given by

$$\mathcal{L}(X, \boldsymbol{\mu}, \lambda) = \frac{1}{2}\|CX - D\|_F^2 + \boldsymbol{\mu}^T(X^T\mathbf{a} - \mathbf{b}) + \lambda \cdot (\|X\|_F^2 - \alpha^2), \tag{B.50}$$

where $\boldsymbol{\mu} \in \mathbb{R}^Q$ and $\lambda \in \mathbb{R}, \lambda \geq 0$ are the Lagrange multipliers for the equality and inequality constraints, respectively. The stationarity condition is then written as

$$\nabla_X \mathcal{L}(X, \boldsymbol{\mu}, \lambda) = C^T CX - C^T D + \mathbf{a}\boldsymbol{\mu}^T + \lambda \cdot X = 0, \tag{B.51}$$

where we have omitted the factor 2 in the last term since it can be integrated into the Lagrange multiplier λ. Combining this result with the equality constraint of the problem, we obtain the following linear equation

$$\begin{bmatrix} C^T C + \lambda I_M & \mathbf{a} \\ \mathbf{a}^T & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ \boldsymbol{\mu}^T \end{bmatrix} = \begin{bmatrix} C^T D \\ \mathbf{b}^T \end{bmatrix}. \tag{B.52}$$

The first matrix in the equation is invertible since C is full column rank and $\mathbf{a} \neq 0$, and its inverse is given by

$$\begin{bmatrix} C^T C + \lambda I_M & \mathbf{a} \\ \mathbf{a}^T & 0 \end{bmatrix}^{-1} = \frac{1}{\mathbf{a}^T R_\lambda^{-1} \mathbf{a}} \begin{bmatrix} \mathbf{a}^T R_\lambda^{-1}\mathbf{a} \cdot R_\lambda^{-1} - R_\lambda^{-1}\mathbf{a}\mathbf{a}^T R_\lambda^{-1} & R_\lambda^{-1}\mathbf{a} \\ \mathbf{a}^T R_\lambda^{-1} & -1 \end{bmatrix}, \tag{B.53}$$

with $R_\lambda = C^T C + \lambda \cdot I_M$. Solving (B.52) for X and $\boldsymbol{\mu}$ then results in

$$X_\lambda = R_\lambda^{-1} C^T D - \frac{R_\lambda^{-1}\mathbf{a}\mathbf{a}^T R_\lambda^{-1} C^T D}{\mathbf{a}^T R_\lambda^{-1}\mathbf{a}} + \frac{R_\lambda^{-1}\mathbf{a}\mathbf{b}^T}{\mathbf{a}^T R_\lambda^{-1}\mathbf{a}}. \tag{B.54}$$

It can be verified that $X_\lambda^T \mathbf{a} = \mathbf{b}$ for all values of λ.

Let us now look at the possible solutions of the problem. If the inequality constraint is not active, i.e., $\|X_\lambda\|_F^2 < \alpha^2$, then we must have $\lambda = 0$ due to the complementary slackness condition (A.30). Note that Problem (B.46) is the same as Problem (B.39) apart from the fact that the former has the additional inequality constraint, but if this constraint is not active, then the solutions of both problems must be the same, since adding a constraint cannot decrease the

minimal value of a problem. Taking $\lambda = 0$ in X_λ indeed gives the same matrix as in (B.45):

$$X^* = (C^T C)^{-1} \left(C^T D + \frac{\mathbf{a}}{\mathbf{a}^T (C^T C)^{-1} \mathbf{a}} (\mathbf{b}^T - \mathbf{a}^T (C^T C)^{-1} C^T D) \right), \quad \text{(B.55)}$$

which solves Problem (B.46).

If on the other hand the constraint is active, we have to solve the following equation for λ to find an optimal solution X^*

$$\lambda^* : \|X_{\lambda^*}\|_F^2 = \alpha^2, \; X^* = X_{\lambda^*}. \quad \text{(B.56)}$$

As a side note, if the linear equality constraint was not present in the equation, we could simply remove the term $\mathbf{a}\boldsymbol{\mu}^T$ from the stationarity equation (B.51), which would result in

$$X_\lambda = (C^T C + \lambda \cdot I_M)^{-1} C^T D. \quad \text{(B.57)}$$

The solution would then have to be found in the same way as in (B.56).

B.8 Linear objective with a quadratic constraint

Let us consider the following problem

$$\begin{aligned} &\underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} && \operatorname{tr}(X^T A) \\ &\text{subject to} && \operatorname{tr}(X^T B X) \leq 1, \end{aligned} \quad \text{(B.58)}$$

where $A \neq 0$ and B is assumed to be symmetric positive definite, making (B.58) a convex problem. The Lagrangian of Problem (B.58) is

$$\mathcal{L}(X, \lambda) = \operatorname{tr}(X^T A) + \lambda \cdot (\operatorname{tr}(X^T B X) - 1), \quad \text{(B.59)}$$

with λ scalar-valued, while the stationarity condition is written as

$$\nabla_X \mathcal{L}(X, \lambda) = A + 2\lambda B X = 0. \quad \text{(B.60)}$$

Due to the inequality constraint, an optimal pair (X, λ) of Problem (B.58) must also satisfy the dual feasibility condition (A.29) $\lambda \geq 0$ and the complementary slackness condition (A.30) $\lambda \cdot (\operatorname{tr}(X^T B X) - 1) = 0$. The latter condition implies that either $\lambda = 0$ or $\operatorname{tr}(X^T B X) - 1 = 0$, i.e., the constraint is satisfied with equality. However, from the stationarity condition (B.60), we see that if $\lambda = 0$

then an optimal solution can be obtained only if $A = 0$, which is not the case in this example. Therefore, we must have $\lambda > 0$ and $\mathrm{tr}(X^T BX) = 1$. From (B.60), we obtain

$$X = -\frac{1}{2\lambda} B^{-1} A, \tag{B.61}$$

such that

$$\mathrm{tr}(X^T BX) = \frac{1}{4\lambda^2} \mathrm{tr}(A^T B^{-1} B B^{-1} A) = 1. \tag{B.62}$$

Since $\lambda > 0$, this leads to

$$\lambda = \frac{1}{2} \sqrt{\mathrm{tr}(A^T B^{-1} A)} \tag{B.63}$$

and finally

$$X^* = -\frac{1}{\sqrt{\mathrm{tr}(A^T B^{-1} A)}} B^{-1} A. \tag{B.64}$$

B.9 Quadratic over linear

The quadratic over linear problem can be written in the following way

$$\begin{array}{ll} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} & \dfrac{\mathrm{tr}(X^T RX) + \mathrm{tr}(X^T A)}{\mathrm{tr}(X^T B) + c} \\ \text{subject to} & \mathrm{tr}(X^T B) + c > 0, \end{array} \tag{B.65}$$

where we consider R to be symmetric positive definite. We will derive the solution of Problem (B.65) in a similar way as to [180]. Taking $t = 1/\mathrm{tr}(X^T B) + c$ and making the change of variables $Y = Xt$, we obtain

$$\begin{array}{ll} \underset{Y \in \mathbb{R}^{M \times Q}, t \in \mathbb{R}}{\text{minimize}} & t \cdot \left(\mathrm{tr} \left(\dfrac{Y^T RY}{t^2} \right) + \mathrm{tr} \left(\dfrac{Y^T A}{t} \right) \right) \\ \text{subject to} & t > 0, \\ & \dfrac{1}{t} = \mathrm{tr} \left(\dfrac{Y^T B}{t} \right) + c. \end{array} \tag{B.66}$$

After rewriting the last constraint as $\mathrm{tr}\left(Y^T B\right) + ct = 1$, the Lagrangian of the problem is then given by

$$\mathcal{L}(Y, t, \mu, \lambda) = \frac{1}{t} \mathrm{tr}\left(Y^T RY\right) + \mathrm{tr}\left(Y^T A\right) - \lambda t + \mu \cdot \left(\mathrm{tr}\left(Y^T B\right) + ct - 1 \right), \tag{B.67}$$

with μ, λ scalar-valued, and $\lambda \geq 0$. Note that the Lagrangian multiplier λ must be zero due to the complementary slackness condition (A.30) since we would otherwise have $\mathrm{tr}(X^T B) + c = 0$, contradicting the constraint of Problem (B.65). The stationarity conditions are then written as

$$\nabla_Y \mathcal{L}(Y, t, \mu, \lambda = 0) = \frac{2}{t} RY + A + \mu B = 0, \tag{B.68}$$

$$\frac{d}{dt} \mathcal{L}(Y, t, \mu, \lambda = 0) = -\frac{1}{t^2} \mathrm{tr}\left(Y^T R Y\right) + \mu c = 0. \tag{B.69}$$

From the first equation, we obtain

$$\frac{Y}{t} = X = -\frac{1}{2} R^{-1}(A + \mu B), \tag{B.70}$$

which then gives for the second equation:

$$-\frac{1}{4} \mathrm{tr}\left((A + \mu B)^T R^{-1}(A + \mu B)\right) + \mu c = 0. \tag{B.71}$$

This is a quadratic equation for the variable μ, and defining $a = -\mathrm{tr}(A^T R^{-1} A)/4$, $b = -\mathrm{tr}(B^T R^{-1} B)/4$ and $d = -\mathrm{tr}(A^T R^{-1} B)/2$, we have

$$b\mu^2 + (c + d)\mu + a = 0, \tag{B.72}$$

for which the solutions are given by

$$\begin{aligned} \mu_\pm = & \frac{\mathrm{tr}(A^T R^{-1} B)/2 - c}{-\mathrm{tr}(B^T R^{-1} B)/2} \\ & \pm \frac{\sqrt{(c - \mathrm{tr}(A^T R^{-1} B)/2)^2 - \mathrm{tr}(A^T R^{-1} A)\mathrm{tr}(B^T R^{-1} B)/4}}{-\mathrm{tr}(B^T R^{-1} B)/2}. \end{aligned} \tag{B.73}$$

Since the Lagrange multiplier must be real-valued, we need to have

$$(c - \mathrm{tr}(A^T R^{-1} B)/2)^2 \geq \mathrm{tr}(A^T R^{-1} A)\mathrm{tr}(B^T R^{-1} B)/4, \tag{B.74}$$

which is a convex quadratic equation for c. The zeros of this equation are given by

$$2c_{0\pm} = \mathrm{tr}(A^T R^{-1} B) \pm \sqrt{\mathrm{tr}(A^T R^{-1} A)\mathrm{tr}(B^T R^{-1} B)}, \tag{B.75}$$

therefore, we need to have

$$c \geq c_{0+}, \text{ or } c \leq c_{0-} \tag{B.76}$$

for the problem to be feasible. Plugging (B.73) into (B.70) and in turn plugging this result in $\mathrm{tr}(Y^T B/t) + c$, we can obtain

$$\frac{1}{t} = \pm \sqrt{(c - \mathrm{tr}(A^T R^{-1} B)/2)^2 - \mathrm{tr}(A^T R^{-1} A)\mathrm{tr}(B^T R^{-1} B)/4}. \tag{B.77}$$

Since t must be positive, we select the solution with the "+" sign to finally obtain

$$X^* = -\frac{1}{2}R^{-1}(A + \mu_+ B). \tag{B.78}$$

On the other hand, we can solve auxiliary problems of the form

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \operatorname{tr}(X^T R X) + \operatorname{tr}(X^T A) - \rho \cdot (\operatorname{tr}(X^T B) + c) \\ \text{subject to} \quad & \operatorname{tr}(X^T B) + c > 0, \end{aligned} \tag{B.79}$$

instead of solving (B.65) directly as explained in Section 7.6.2. Given a fixed ρ, the Lagrangian of this problem can be written as

$$\mathcal{L}(X, \rho, \lambda) = \operatorname{tr}(X^T R X) + \operatorname{tr}(X^T A) - (\rho + \lambda) \cdot (\operatorname{tr}(X^T B) + c). \tag{B.80}$$

Again, we have $\lambda = 0$ due to the complementary slackness condition, such that the stationarity condition of the auxiliary problem (B.79) is given by

$$\nabla_X \mathcal{L}(X, \rho, \lambda = 0) = 2RX + A - \rho \cdot B = 0, \tag{B.81}$$

resulting in the solution

$$X^*_\rho = \frac{1}{2}R^{-1}(\rho \cdot B - A). \tag{B.82}$$

B.10 Regularized total least squares

The regularized total least squares problem given in [99, 181] can be extended from the vector case to the matrix one as follows

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \frac{\operatorname{tr}(X^T R X) - 2\operatorname{tr}(X^T A) + b}{1 + \|X\|_F^2} \\ \text{subject to} \quad & \|X^T C\|_F^2 \leq \delta^2, \end{aligned} \tag{B.83}$$

for which the Lagrangian is

$$\mathcal{L}(X, \lambda) = \frac{\operatorname{tr}(X^T R X) - 2\operatorname{tr}(X^T A) + b}{1 + \|X\|_F^2} + \lambda \cdot \left(\|X^T C\|_F^2 - \delta^2 \right), \tag{B.84}$$

where λ is the scalar-valued Lagrange multiplier such that $\lambda \geq 0$. Using the result in Example A.5, we find that the stationarity condition of Problem (B.83) is

$$\nabla_X \mathcal{L}(X, \lambda) = \frac{2RX - 2A}{1 + \|X\|_F^2} + 2X \cdot \frac{\operatorname{tr}(X^T R X) - 2\operatorname{tr}(X^T A) + b}{(1 + \|X\|_F^2)^2} + 2\lambda C C^T X = 0, \tag{B.85}$$

which, under the assumption that the constraint is active, can be solved using an iterative quadratic eigenvalue solver [99].

On the other hand, Problem (B.83) can be solved by iteratively solving auxiliary problems of the form

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \operatorname{tr}(X^T R X) - 2\operatorname{tr}(X^T A) + b - \rho\cdot(1+\|X\|_F^2) \\ \text{subject to} \quad & \|X^T C\|_F^2 \leq \delta^2, \end{aligned} \tag{B.86}$$

as detailed in [100] and Section 7.6.1. The Lagrangian of the problem is given by

$$\mathcal{L}(X,\rho,\lambda) = \operatorname{tr}\left(X^T(R-\rho\cdot I_M - \lambda C^T C)X\right) - 2\operatorname{tr}(X^T A) + b - \rho - \lambda\delta^2, \tag{B.87}$$

which results in the following stationarity condition

$$\nabla_X \mathcal{L}(X,\rho,\lambda) = 2(R-\rho\cdot I_M + \lambda C^T C)X - 2AX = 0. \tag{B.88}$$

Assuming that ρ is not equal to an eigenvalue of R, if the solution of the unconstrained problem

$$X_\rho^* = (R-\rho\cdot I_M)^{-1}AX, \tag{B.89}$$

satisfies $\|C^T X_\rho^*\|_F^2 < \delta^2$, i.e., if the inequality holds strictly, we have $\lambda = 0$ due to complementary slackness (A.30) and X_ρ^* would also solve (B.86). Otherwise, we have

$$X_{\rho,\lambda} = (R-\rho\cdot I_M + \lambda C^T C)^{-1}AX, \tag{B.90}$$

for which a solution is given by

$$\lambda^* : \|C^T X_{\rho,\lambda^*}\|_F^2 = \delta^2,\ X_\rho^* = X_{\rho,\lambda^*}. \tag{B.91}$$

Bibliography

[1] B. Somers, T. Francart, and A. Bertrand, "A generic EEG artifact removal algorithm based on the multi-channel wiener filter," *Journal of neural engineering*, vol. 15, no. 3, p. 036007, 2018.

[2] 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.

[3] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Muller, "Optimizing spatial filters for robust EEG single-trial analysis," *IEEE Signal processing magazine*, vol. 25, no. 1, pp. 41–56, 2007.

[4] H. Ramoser, J. Muller-Gerking, and G. Pfurtscheller, "Optimal spatial filtering of single trial EEG during imagined hand movement," *IEEE transactions on rehabilitation engineering*, vol. 8, no. 4, pp. 441–446, 2000.

[5] E. Björnson and L. Sanguinetti, "Scalable cell-free massive MIMO systems," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 4247–4261, 2020.

[6] L. Sanguinetti, E. Björnson, and J. Hoydis, "Toward massive MIMO 2.0: Understanding spatial correlation, interference suppression, and pilot contamination," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 232–257, 2019.

[7] E. Nayebi, A. Ashikhmin, T. L. Marzetta, and B. D. Rao, "Performance of cell-free massive MIMO systems with MMSE and LSFD receivers," in Proceedings of the *2016 50th Asilomar Conference on Signals, Systems and Computers*, pp. 203–207, 2016.

[8] A. Pezeshki, B. D. Van Veen, L. L. Scharf, H. Cox, and M. L. Nordenvaad, "Eigenvalue beamforming using a multirank MVDR beamformer and

subspace selection," *IEEE transactions on signal processing*, vol. 56, no. 5, pp. 1954–1967, 2008.

[9] 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.

[10] N. Furnon, R. Serizel, I. Illina, and S. Essid, "Distributed speech separation in spatially unconstrained microphone arrays," in Proceedings of the *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4490–4494, 2021.

[11] J. Zhang, R. Heusdens, and R. C. Hendriks, "Rate-distributed spatial filtering based noise reduction in wireless acoustic sensor networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2015–2026, 2018.

[12] J. Benesty, J. Chen, and Y. Huang, *Microphone array signal processing*. Springer Science & Business Media, 2008, vol. 1.

[13] S. Doclo and M. Moonen, "GSVD-based optimal filtering for single and multimicrophone speech enhancement," *IEEE Transactions on signal processing*, vol. 50, no. 9, pp. 2230–2244, 2002.

[14] M. S. Brandstein and D. B. Ward, "Microphone arrays - signal processing techniques and applications," in Proceedings of the *Microphone Arrays*, 2001.

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

[16] H. L. Van Trees, *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2002.

[17] A. B. Gershman, N. D. Sidiropoulos, S. Shahbazpanahi, M. Bengtsson, and B. Ottersten, "Convex optimization-based beamforming," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 62–75, 2010.

[18] S. A. Vorobyov, "Principles of minimum variance robust adaptive beamforming design," *Signal Processing*, vol. 93, no. 12, pp. 3264–3277, 2013.

[19] K. Liu, Y.-Q. Cheng, and J.-Y. Yang, "A generalized optimal set of discriminant vectors," *Pattern Recognition*, vol. 25, no. 7, pp. 731–739, 1992.

[20] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[21] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis." *Journal of machine learning research*, vol. 8, no. 5, 2007.

[22] M. Borga, "Learning multidimensional signal processing," Ph.D. dissertation, Linköping University Electronic Press, 1998.

[23] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[24] H. Abdi, V. Guillemot, A. Eslami, and D. Beaton, *Canonical Correlation Analysis*. New York, NY: Springer New York, 2018, pp. 177–192. [Online]. Available: https://doi.org/10.1007/978-1-4939-7131-2_110191

[25] A. Hjorungnes and D. Gesbert, "Complex-valued matrix differentiation: Techniques and key results," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2740–2746, 2007.

[26] T. Adali and P. J. Schreier, "Optimization and estimation of complex-valued signals: Theory and applications in filtering and blind source separation," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 112–128, 2014.

[27] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014, http://dx.doi.org/10.1561/2200000051.

[28] K. Kreutz-Delgado, "The complex gradient operator and the CR-calculus," *arXiv preprint arXiv:0906.4835*, 2009.

[29] I. S. Reed, J. D. Mallett, and L. E. Brennan, "Rapid convergence rate in adaptive arrays," *IEEE Transactions on Aerospace and Electronic Systems*, no. 6, pp. 853–863, 1974.

[30] S. Valaee, B. Champagne, and P. Kabal, "Localization of wideband signals using least-squares and total least-squares approaches," *IEEE Transactions on Signal Processing*, vol. 47, no. 5, pp. 1213–1222, 1999.

[31] C. E. Davila, "An efficient recursive total least squares algorithm for fir adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 42, no. 2, pp. 268–280, 1994.

[32] P. S. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Springer Nature, 2019.

[33] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2011.

[34] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[35] H.-F. Chen and G. Yin, "Asymptotic properties of sign algorithms for adaptive filtering," *IEEE transactions on automatic control*, vol. 48, no. 9, pp. 1545–1556, 2003.

[36] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A survey on wireless body area networks," *Wireless networks*, vol. 17, no. 1, pp. 1–18, 2011.

[37] B. Rivet, A. Souloumiac, V. Attina, and G. Gibert, "xDAWN algorithm to enhance evoked potentials: application to brain–computer interface," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 8, pp. 2035–2043, 2009.

[38] A. M. Narayanan and A. Bertrand, "Analysis of miniaturization effects and channel selection strategies for EEG sensor networks with application to auditory attention detection," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 1, pp. 234–244, 2020.

[39] A. Bertrand, "Applications and trends in wireless acoustic sensor networks: A signal processing perspective," in Proceedings of the *2011 18th IEEE symposium on communications and vehicular technology in the Benelux (SCVT)*, pp. 1–6, 2011.

[40] 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.

[41] A. Bertrand, J. Callebaut, and M. Moonen, "Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks," in Proceedings of the *Proc. of the International Workshop on Acoustic Echo and Noise Control (IWAENC)*, 2010.

[42] A. Bertrand and M. Moonen, "Distributed LCMV beamforming in a wireless sensor network with single-channel per-node signal transmission," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3447–3459, 2013.

[43] M. Cobos, F. Antonacci, A. Mouchtaris, and B. Lee, "Wireless acoustic sensor networks and applications," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 1085290, 3 pages, 2017.

[44] Z. Quan, S. Cui, and A. H. Sayed, "Optimal linear cooperation for spectrum sensing in cognitive radio networks," *IEEE Journal of selected topics in signal processing*, vol. 2, no. 1, pp. 28–40, 2008.

[45] S. Maleki, A. Pandharipande, and G. Leus, "Energy-efficient distributed spectrum sensing for cognitive sensor networks," *IEEE sensors journal*, vol. 11, no. 3, pp. 565–573, 2010.

[46] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in Proceedings of the *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 13–24, 2004.

[47] M. F. Othman and K. Shazali, "Wireless sensor network applications: A study in environment monitoring system," *Procedia Engineering*, vol. 41, pp. 1204–1210, 2012.

[48] C. Albaladejo, P. Sánchez, A. Iborra, F. Soto, J. A. López, and R. Torres, "Wireless sensor networks for oceanographic monitoring: A systematic review," *Sensors*, vol. 10, no. 7, pp. 6948–6968, 2010.

[49] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[50] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer networks*, vol. 51, no. 4, pp. 921–960, 2007.

[51] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[52] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in Proceedings of the *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 20–27, 2004.

[53] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.

[54] F. S. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," *IEEE Transactions on automatic control*, vol. 55, no. 9, pp. 2069–2084, 2010.

[55] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[56] M. Asam, T. Jamal, M. Adeel, A. Hassan, S. A. Butt, A. Ajaz, and M. Gulzar, "Challenges in wireless body area network," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 11, 2019.

[57] K. Hasan, K. Biswas, K. Ahmed, N. S. Nafi, and M. S. Islam, "A comprehensive review of wireless body area network," *Journal of Network and Computer Applications*, vol. 143, pp. 178–198, 2019.

[58] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1658–1686, 2014.

[59] R. Cavallari, F. Martelli, R. Rosini, C. Buratti, and R. Verdone, "A survey on wireless body area networks: Technologies and design challenges," *IEEE communications surveys & tutorials*, vol. 16, no. 3, pp. 1635–1657, 2014.

[60] A. J. Casson, D. C. Yates, S. J. Smith, J. S. Duncan, and E. Rodriguez-Villegas, "Wearable electroencephalography," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, no. 3, pp. 44–56, 2010.

[61] Y. M. Chi, S. R. Deiss, and G. Cauwenberghs, "Non-contact low power EEG/ECG electrode for high density wearable biopotential sensor networks," in Proceedings of the *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, pp. 246–250, 2009.

[62] Z. Sheng, S. Pfersich, A. Eldridge, J. Zhou, D. Tian, and V. C. Leung, "Wireless acoustic sensor networks and edge computing for rapid acoustic monitoring," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 64–74, 2019.

[63] F. Alías, R. M. Alsina-Pagès, *et al.*, "Review of wireless acoustic sensor networks for environmental noise monitoring in smart cities," *Journal of sensors*, vol. 2019, 2019.

[64] M. Cobos, F. Antonacci, A. Alexandridis, A. Mouchtaris, B. Lee, *et al.*, "A survey of sound source localization methods in wireless acoustic sensor networks," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[65] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[66] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM Journal on Optimization*, vol. 7, no. 4, pp. 913–926, 1997.

[67] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4064–4077, 2007.

[68] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in Proceedings of the *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 6698–6703, 2005.

[69] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[70] 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.

[71] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE transactions on signal processing*, vol. 58, no. 3, pp. 1035–1048, 2009.

[72] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012.

[73] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, 2012.

[74] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[75] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[76] 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.

[77] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in Proceedings of the *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 5445–5450, 2012.

[78] 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.

[79] 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.

[80] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, 2018.

[81] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: models, limits, and algorithms," *IEEE Transactions on Signal processing*, vol. 59, no. 1, pp. 386–398, 2010.

[82] 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.

[83] C. Manss, D. Shutin, and G. Leus, "Distributed splitting-over-features sparse Bayesian learning with alternating direction method of multipliers," in Proceedings of the *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3654–3658, 2018.

[84] C. Gratton, N. K. Venkategowda, R. Arablouei, and S. Werner, "Distributed learning over networks with non-smooth regularizers and feature partitioning," in Proceedings of the *2021 29th European Signal Processing Conference (EUSIPCO)*, pp. 1840–1844, 2021.

[85] B. Zhang, J. Geng, W. Xu, and L. Lai, "Communication efficient distributed learning with feature partitioned data," in Proceedings of the *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, 2018.

[86] 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.

[87] 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.

[88] S. Doclo, M. Moonen, T. Van den Bogaert, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 1, pp. 38–51, 2009.

[89] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks-—Part I: Sequential node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5277–5291, 2010.

[90] S. Markovich-Golan, S. Gannot, and I. Cohen, "Distributed multiple constraints generalized sidelobe canceler for fully connected wireless acoustic sensor networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 343–356, 2012.

[91] 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.

[92] 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.

[93] Y.-F. Guo, S.-J. Li, J.-Y. Yang, T.-T. Shu, and L.-D. Wu, "A generalized Foley–Sammon transform based on generalized Fisher discriminant criterion and its application to face recognition," *Pattern Recognition Letters*, vol. 24, no. 1-3, pp. 147–158, 2003.

[94] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, "Trace ratio vs. ratio trace for dimensionality reduction," in Proceedings of the *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[95] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 729–735, 2009.

[96] S. Yan and X. Tang, "Trace quotient problems revisited," in Proceedings of the *European Conference on Computer Vision*, pp. 232–244, 2006.

[97] F. Nie, S. Xiang, Y. Jia, C. Zhang, and S. Yan, "Trace ratio criterion for feature selection." in Proceedings of the *AAAI*, vol. 2, pp. 671–676, 2008.

[98] C. Shen, H. Li, and M. J. Brooks, "Supervised dimensionality reduction via sequential semidefinite programming," *Pattern Recognition*, vol. 41, no. 12, pp. 3644–3652, 2008.

[99] D. M. Sima, S. Van Huffel, and G. H. Golub, "Regularized total least squares based on quadratic eigenvalue problem solvers," *BIT Numerical Mathematics*, vol. 44, no. 4, pp. 793–812, 2004.

[100] A. Beck, A. Ben-Tal, and M. Teboulle, "Finding a global optimal solution for a quadratically constrained fractional quadratic problem with applications to the regularized total least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 2, pp. 425–445, 2006.

[101] 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*, vol. 71, pp. 1863–1878, 2023.

[102] 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.

[103] C. A. Musluoglu, M. Moonen, and A. Bertrand, "Improved tracking for distributed signal fusion optimization in a fully-connected wireless sensor network," in Proceedings of the *2022 30th European Signal Processing Conference (EUSIPCO)*, pp. 1836–1840, 2022.

[104] C. A. Musluoglu and A. Bertrand, "A distributed adaptive algorithm for node-specific signal fusion problems in wireless sensor networks," *Accepted in Proceedings of the 2023 31st European Signal Processing Conference (EUSIPCO)*, pp. 1–5, 2023.

[105] C. A. Musluoglu and A. Bertrand, "Distributed adaptive trace ratio optimization in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3653–3670, 2021.

[106] C. A. Musluoglu and A. Bertrand, "Distributed trace ratio optimization in fully-connected sensor networks," in Proceedings of the *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 1991–1995, 2021.

[107] C. A. Musluoglu and A. Bertrand, "Distributed adaptive signal fusion for fractional programs," *Submitted for review*, pp. 1–13, 2023.

[108] C. A. Musluoglu and A. Bertrand, "A computationally efficient algorithm for distributed adaptive signal fusion based on fractional programs," in

Proceedings of the *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[109] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear Gauss–Seidel method under convex constraints," *Operations research letters*, vol. 26, no. 3, pp. 127–136, 2000.

[110] C. A. Musluoglu and A. Bertrand, "DASF Toolbox," 2022. [Online]. Available: https://github.com/AlexanderBertrandLab/DASF_toolbox

[111] H. Cox, R. Zeskind, and M. Owen, "Robust adaptive beamforming," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 10, pp. 1365–1376, 1987.

[112] J. Wouters, P. Patrinos, F. Kloosterman, and A. Bertrand, "Multi-pattern recognition through maximization of signal-to-peak-interference ratio with application to neural spike sorting," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6240–6254, 2020.

[113] Y. Wang, S. Gao, and X. Gao, "Common spatial pattern method for channel selelction in motor imagery based brain-computer interface," in Proceedings of the *2005 IEEE engineering in medicine and biology 27th annual conference*, pp. 5392–5395, 2006.

[114] A. Bertrand and M. Moonen, "Distributed node-specific LCMV beamforming in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 233–246, 2011.

[115] S. Kim, R. Pasupathy, and S. G. Henderson, "A guide to sample average approximation," *Handbook of simulation optimization*, pp. 207–243, 2015.

[116] J. Hadamard, "Sur les problèmes aux dérivées partielles et leur signification physique," *Princeton university bulletin*, pp. 49–52, 1902.

[117] Y.-h. Zhou, J. Yu, H. Yang, and S.-w. Xiang, "Hadamard types of well-posedness of non-self set-valued mappings for coincide points," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 63, no. 5-7, pp. e2427–e2436, 2005.

[118] D. W. Peterson, "A review of constraint qualifications in finite-dimensional spaces," *Siam Review*, vol. 15, no. 3, pp. 639–654, 1973.

[119] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

[120] P.-A. Absil, R. Mahony, and B. Andrews, "Convergence of the iterates of descent methods for analytic cost functions," *SIAM Journal on Optimization*, vol. 16, no. 2, pp. 531–547, 2005.

[121] A. A. Goldstein, "On steepest descent," *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, vol. 3, no. 1, pp. 147–151, 1965.

[122] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[123] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[124] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. SIAM, 2000.

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

[126] A. Ben-Israel and T. N. Greville, *Generalized inverses: theory and applications*. Springer Science & Business Media, 2003, vol. 15.

[127] K. Fukunaga, *Introduction to statistical pattern recognition*. Elsevier, 2013.

[128] T. Kato, *Perturbation theory for linear operators*. Springer Science & Business Media, 2013, vol. 132.

[129] T. T. Ngo, M. Bellalij, and Y. Saad, "The trace ratio optimization problem," *SIAM review*, vol. 54, no. 3, pp. 545–569, 2012.

[130] D. P. Bertsekas, *Nonlinear Programming: 2nd Edition*. Athena Scientific, 1999.

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

[132] B. Sterckx, "Luistert de volgende wearable naar onze hersenen ?" 2022-10-02.

[133] S. Markovich, S. Gannot, and I. Cohen, "Multichannel eigenspace beamforming in a reverberant noisy environment with multiple interfering speech signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1071–1086, 2009.

[134] S. Doclo, T. J. Klasen, T. Van den Bogaert, J. Wouters, and M. Moonen, "Theoretical analysis of binaural cue preservation using multi-channel wiener filtering and interaural transfer functions," in Proceedings of the *Proc. Int. Workshop Acoust. Echo Noise Control (IWAENC)*, pp. 1–4, 2006.

[135] J. Szurley, A. Bertrand, and M. Moonen, "Topology-independent distributed adaptive node-specific signal estimation in wireless sensor networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 130–144, 2016.

[136] S. Markovich-Golan, S. Gannot, and I. Cohen, "A reduced bandwidth binaural MVDR beamformer," in Proceedings of the *Proc. of the International Workshop on Acoustic Echo and Noise Control (IWAENC), Tel-Aviv, Israel*, 2010.

[137] X. Guo, M. Yuan, Y. Ke, C. Zheng, and X. Li, "Distributed node-specific block-diagonal LCMV beamforming in wireless acoustic sensor networks," *Signal Processing*, vol. 185, p. 108085, 2021.

[138] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2733–2748, 2015.

[139] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3448–3460, 2015.

[140] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[141] J. W. Sammon, "An optimal discriminant plane," *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 826–829, 1970.

[142] D. H. Foley and J. W. Sammon, "An optimal set of discriminant vectors," *IEEE Transactions on Computers*, vol. 100, no. 3, pp. 281–289, 1975.

[143] L.-H. Zhang, L.-Z. Liao, and M. K. Ng, "Fast algorithms for the generalized Foley–Sammon discriminant analysis," *SIAM journal on matrix analysis and applications*, vol. 31, no. 4, pp. 1584–1605, 2010.

[144] L.-H. Zhang, W. H. Yang, and L.-Z. Liao, "A note on the trace quotient problem," *Optimization Letters*, vol. 8, no. 5, pp. 1637–1645, 2014.

[145] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," in Proceedings of the *Advances in neural information processing systems*, pp. 1569–1576, 2005.

[146] 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.

[147] D. Cai, X. He, J. Han, and H.-J. Zhang, "Orthogonal Laplacianfaces for face recognition," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3608–3614, 2006.

[148] R. E. Prieto, "A general solution to the maximization of the multidimensional generalized Rayleigh quotient used in linear discriminant analysis for signal classification," in Proceedings of the *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 6, pp. VI–157, 2003.

[149] S. Crandall, "Iterative procedures related to relaxation methods for eigenvalue problems," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 207, no. 1090, pp. 416–423, 1951.

[150] P. Lancaster, "A generalised Rayleigh quotient iteration for lambda-matrices," *Archive for Rational Mechanics and Analysis*, vol. 8, no. 1, p. 309, 1961.

[151] B. N. Parlett, *The symmetric eigenvalue problem*. siam, 1998, vol. 20.

[152] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2012, vol. 3.

[153] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[154] 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.

[155] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[156] S. Schaible and T. Ibaraki, "Fractional programming," *European journal of operational research*, vol. 12, no. 4, pp. 325–338, 1983.

[157] A. Charnes and W. W. Cooper, "Programming with linear fractional functionals," *Naval Research logistics quarterly*, vol. 9, no. 3-4, pp. 181–186, 1962.

[158] S. Schaible, "Parameter-free convex equivalent and dual programs of fractional programming problems," *Zeitschrift für Operations Research*, vol. 18, no. 5, pp. 187–196, 1974.

[159] W. Dinkelbach, "On nonlinear fractional programming," *Management science*, vol. 13, no. 7, pp. 492–498, 1967.

[160] R. Jagannathan, "On some properties of programming problems in parametric form pertaining to fractional programming," *Management Science*, vol. 12, no. 7, pp. 609–615, 1966.

[161] S. Schaible, "Fractional programming. II, on Dinkelbach's algorithm," *Management science*, vol. 22, no. 8, pp. 868–873, 1976.

[162] J. Crouzeix, J. Ferland, and S. Schaible, "An algorithm for generalized fractional programs," *Journal of Optimization Theory and Applications*, vol. 47, no. 1, pp. 35–49, 1985.

[163] J.-P. Crouzeix and J. A. Ferland, "Algorithms for generalized fractional programming," *Mathematical Programming*, vol. 52, no. 1, pp. 191–207, 1991.

[164] J. Crouzeix, J. Ferland, and S. Schaible, "A note on an algorithm for generalized fractional programs," *Journal of Optimization Theory and Applications*, vol. 50, no. 1, pp. 183–187, 1986.

[165] J.-P. Crouzeix, J. A. Ferland, and H. Van Nguyen, "Revisiting Dinkelbach-type algorithms for generalized fractional programs," *Opsearch*, vol. 45, no. 2, pp. 97–110, 2008.

[166] H. Zhu, G. Leus, and G. B. Giannakis, "Sparse regularized total least squares for sensing applications," in Proceedings of the *2010 IEEE 11th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2010.

[167] A. Pruessner and D. P. O'Leary, "Blind deconvolution using a regularized structured total least norm algorithm," *SIAM journal on matrix analysis and applications*, vol. 24, no. 4, pp. 1018–1037, 2003.

[168] N. H. Younan and X. Fan, "Signal restoration via the regularized constrained total least squares," *Signal Processing*, vol. 71, no. 1, pp. 85–93, 1998.

[169] C. Hovine and A. Bertrand, "A distributed adaptive algorithm for non-smooth spatial filtering problems," in Proceedings of the *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[170] A. Hassani, J. Plata-Chaves, M. H. Bahari, M. Moonen, and A. Bertrand, "Multi-task wireless sensor network for joint distributed node-specific signal enhancement, LCMV beamforming and DOA estimation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 3, pp. 518–533, 2017.

[171] X. Zhao, S.-Y. Tu, and A. H. Sayed, "Diffusion adaptation over networks under imperfect information exchange and non-stationary data," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3460–3475, 2012.

[172] W. Zhu, X. Ma, X.-H. Zhu, K. Ugurbil, W. Chen, and X. Wu, "Denoise functional magnetic resonance imaging with random matrix theory based principal component analysis," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 11, pp. 3377–3388, 2022.

[173] L. Yang, M. R. McKay, and R. Couillet, "High-dimensional MVDR beamforming: Optimized solutions based on spiked random matrix models," *IEEE Transactions on Signal Processing*, vol. 66, no. 7, pp. 1933–1947, 2018.

[174] R. Couillet and M. Debbah, "Signal processing in large systems: A new paradigm," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 24–39, 2012.

[175] S. X. Wu, H.-T. Wai, L. Li, and A. Scaglione, "A review of distributed algorithms for principal component analysis," *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1321–1340, 2018.

[176] S. Srinivasan and N. Panda, "What is the gradient of a scalar function of a symmetric matrix?" *Indian Journal of Pure and Applied Mathematics*, pp. 1–13, 2022.

[177] N. Boumal, *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[178] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[179] B. Ghojogh, F. Karray, and M. Crowley, "Eigenvalue and generalized eigenvalue problems: Tutorial," *arXiv preprint arXiv:1903.11240*, 2019.

[180] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

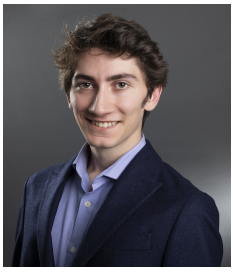
[181] G. H. Golub, P. C. Hansen, and D. P. O'Leary, "Tikhonov regularization and total least squares," *SIAM journal on matrix analysis and applications*, vol. 21, no. 1, pp. 185–194, 1999.

Acknowledgments

This work was carried out at the ESAT-STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics (Department of Electrical Engineering), KU Leuven.

Cem Ates Musluoglu was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 802895), the FWO (Research Foundation Flanders) project G081722N, and the Flemish Government (AI Research Program).

Curriculum vitae



Cem Ates Musluoglu was born in Ankara, Turkey on February 1st 1996. He received the B.Sc. and M.Sc. degrees in electrical and electronic engineering from École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 2017 and 2019, respectively. Since September 2019, he has been working toward the Ph.D. degree in the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics of the Electrical Engineering Department, KU Leuven, Leuven, Belgium. His research interests include statistical and distributed signal processing, optimization, and algorithm design.

List of Publications

Articles in internationally reviewed journals

1. **C. A. Musluoglu** and A. Bertrand, "Distributed adaptive trace ratio optimization in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3653-3670, 2021.
2. **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*, vol. 71, pp. 1863-1878, 2023.
3. **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.

Articles submitted for review

1. **C. A. Musluoglu** and A. Bertrand, "Distributed Adaptive Signal Fusion for Fractional Programs," Submitted for review, pp. 1-13, 2023.

Papers in proceedings of international conferences

1. **C. A. Musluoglu** and A. Bertrand, "Distributed trace ratio optimization in fully-connected sensor networks," in Proceedings of the *2020 28th European Signal Processing Conference (EUSIPCO)*, Amsterdam, The Netherlands, pp. 1991-1995, 2021.

2. **C. A. Musluoglu**, M. Moonen, and A. Bertrand, "Improved Tracking for Distributed Signal Fusion Optimization in a Fully-Connected Wireless Sensor Network," in Proceedings of the *2022 30th European Signal Processing Conference (EUSIPCO)*, Belgrade, Serbia, pp. 1836-1840, 2022.

3. **C. A. Musluoglu** and A. Bertrand, "A computationally efficient algorithm for distributed adaptive signal fusion based on fractional programs," in Proceedings of the *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes, Greece, pp. 1-5, 2023.

4. **C. A. Musluoglu** and A. Bertrand, "A Distributed Adaptive Algorithm for Node-Specific Signal Fusion Problems in Wireless Sensor Networks," Accepted in Proceedings of the *2023 31st European Signal Processing Conference (EUSIPCO)*, Helsinki, Finland, pp. 1-5, 2023.

Abstracts in proceedings of international conferences

1. **C. A. Musluoglu**, C. Hovine, and A. Bertrand, "A Distributed Adaptive Signal Fusion Framework for Spatial Filtering within a Wireless Sensor Network," *SITB2022: 42nd WIC Symposium on Information Theory and Signal Processing in the Benelux*, Louvain-la-Neuve, Belgium, 1-2 June 2022. Poster presentation.

Tutorial presentations

1. A. Bertrand, **C. A. Musluoglu**, and C. Hovine, "Distributed Spatial Filtering and Signal Fusion," Tutorial presented at *2023 31st European Signal Processing Conference (EUSIPCO)*, Helsinki, Finland, 04 Sep 2023-08 Sep 2023.

Software

1. **C. A. Musluoglu** and A. Bertrand, "DASF Toolbox," 2022. [Online]. Available at `https://github.com/AlexanderBertrandLab/DASF_toolbox`

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
STADIUS CENTER FOR DYNAMICAL SYSTEMS, SIGNAL PROCESSING AND DATA ANALYTICS
Kasteelpark Arenberg 10 box 2446
B-3001 Leuven
cemates.musluoglu@esat.kuleuven.be
http://esat.kuleuven.be/stadius