KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
AFDELING ESAT-SCD: SISTA/COSIC/DOCARCH
Kasteelpark Arenberg 10 – B-3001 Leuven

SIGNAL PROCESSING ALGORITHMS FOR WIRELESS ACOUSTIC SENSOR NETWORKS

Promotor:
Prof. dr. ir. M. Moonen

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Alexander BERTRAND

Mei 2011

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
AFDELING ESAT-SCD: SISTA/COSIC/DOCARCH
Kasteelpark Arenberg 10, B-3001 Leuven

SIGNAL PROCESSING ALGORITHMS FOR WIRELESS ACOUSTIC SENSOR NETWORKS

Jury:
Prof. dr. ir. P. Van Houtte, voorzitter
Prof. dr. ir. M. Moonen, promotor
Prof. dr. ir. J. Vandewalle
Prof. dr. ir. H. Van hamme
Prof. dr. ir. D. Van Compernolle
Prof. dr. ir. S. Doclo
(Universität Oldenburg, Duitsland)
Prof. dr. ir. P. C. W. Sommen
(Technische Universiteit Eindhoven, Nederland)
Prof. dr. ir. S. Gannot
(Bar-Ilan University, Israel)

Proefschrift voorgedragen
tot het behalen van de
graad van Doctor in de
Ingenieurswetenschappen

door

Alexander BERTRAND

Mei 2011

ISBN 978-94-6018-329-4

D/2011/7515/31

Voorwoord

Na een avontuur van vier jaar ben ik aan het moeilijkste deel van mijn doctoraatstraject aanbeland: het schrijven van een voorwoord. Het opstellen van de volgende vier paginas kan dan ook gezien worden als een korte samenvatting van de voorbije vier jaar als doctoraatsstudent: een moeilijk proces, met verschillende slapeloze nachten, maar een geweldig gevoel eenmaal de inspiratie (eindelijk) naar boven komt drijven. Het belangrijkste onderdeel van een voorwoord is uiteraard een dankwoord, dus laat ik daarmee beginnen.

Op de eerste plaats komt -zoals het hoort[1]- mijn promotor Marc Moonen. Ik wil Marc bedanken voor zijn vertrouwen, de opportuniteiten en de vrijheid die hij me gaf, de goede begeleiding, de uitgebreide paper-verbeteringen, de interessante 'friday' en e-mail discussies, en uiteraard zijn geweldige (en soms wat scherpe) humor. En dit allemaal ondanks de grote tegenstrijdigheid inzake onze muzikale interesses ('Is er iets mis met je computer Alexander, die maakt zo'n raar geluid?').

Een andere -niet te onderschatten- factor voor de goede afloop van dit doctoraat is het onderwerp waarover ik onderzoek kon doen. Ik ben de eerste om toe te geven dat ik hiermee ontzettend veel geluk heb gehad. Hiervoor wil ik dan ook Simon Doclo en Marc opnieuw bedanken. Zij hebben het lumineuze idee gehad om het onderzoeksdomein van akoestische sensornetwerken aan te boren, wat een onuitputtelijke bron van interessante problemen en nieuwe algoritmes bleek te zijn. Ook het DB-MWF algoritme van Marc en Simon was een ontzettend goede aanzet voor de ontwikkeling van het DANSE algoritme, dat zowat de rode draad vormt in deze doctoraatsthesis.

Een doctoraat kan natuurlijk niet tot een goed einde gebracht worden zonder een examencommissie. *Therefore, I would like to thank all the members of the jury for their efforts to read my text, their valuable comments and suggestions, and their critical questions*: Prof. Marc Moonen, Prof. Simon Doclo, Prof. Joos Vandewalle, Prof. Hugo Van hamme, Prof. Dirk Van Compernolle, Prof.

[1] en geheel terecht!

Piet Sommen, Prof. Sharon Gannot en de voorzitter Prof. Paul Van Houtte.

Verder zijn er nog een aantal mensen die -binnen de context van deze thesis wel te verstaan- in mijn ogen een speciale vermelding verdienen om diverse redenen. Bram Cornelis, die samen met mij zijn doctoraat startte, en met wie ik zowel tijdens als naast het werk menige DSP conversaties en discussies heb gehad, soms tot groot ongenoegen van de rest van het gezelschap[2]. William Vandenberghe, die als 'allesweter' altijd klaarstond om te luisteren en te discussiëren over de lastige wiskundige obstakels die ik tegenkwam (ook al bleken velen daarvan spijtig genoeg niet 'William-oplosbaar' te zijn). Paschalis Tsiaflakis, die mij -hoorde ik achteraf- heeft aangeprezen bij Marc toen ik nog een nietsvermoedende ingenieurstudent was, en die mij 3 maanden heeft moeten verdragen als huisgenoot tijdens ons verblijf in Los Angeles. Peter Ruckebusch van UGent, die heel wat werk heeft gestoken in het maken geluidsopnames met het IBBT sensor netwerk testbed, en met wie ik (in samenwerking met Prof. I. Moerman), ondanks de sterk verschillende wetenschappelijke jargons, heel wat interessante discussies heb gehad.

A special thank you also goes to Prof. Ali H. Sayed, for giving me the opportunity to visit his research group at UCLA. And of course, I want to thank all the guys of the Adaptive Systems Laboratory at UCLA (Zaid, Paolo, Xiaochuan, Jianshu, Victor, Shang Kee, Jae-Woo and Shine) for all the great moments during my stay in LA, and all the help and discussions on the 'big so' whiteboard.

Naast bovengenoemde personen zijn er natuurlijk nog heel wat mensen die een vermelding verdienen, omdat zij onrechtstreeks een steuntje in de rug waren gedurende mijn doctoraat. *Let me start with all my colleagues in the DSP-group at ESAT. In spatio-temporal order, starting with my office buddies: (Papa-)Pepe, Joe, Geert, Ann, Simon, Toon, Bram, Gert, Sam, Deepak, Rodrigo, Kim, Sylwek, Pascal, Bruno, Javier, Vincent, Jan, Beier, Amir, Romain, Prabin and Geert. Thanks for all the great times!* Daarnaast komen natuurlijk ook alle (andere) vrienden, en de hele familie. Om begrijpelijke redenen zal ik jullie hier niet exhaustief opsommen, maar weet dat ik jullie niet vergeten ben!

Ik ben IWT dankbaar voor de financiële ondersteuning van mijn onderzoek gedurende mijn doctoraat, alsook FWO Vlaanderen voor de financiële ondersteuning van mijn onderzoeksverblijf op UCLA.

Mijn gepromoveerde collega's beweren dat de periode van het schrijven van de doctoraatsthesis en de voorbereiding van de preliminaire verdediging een van de

[2]Een welgemeende sorry daarvoor aan Joris, Gerry, Lieboud, Bram, Karen, Fleur, William, Joram, Eleonor, Pieter, en vooral Roel, die zijn ongenoegen hieromtrent vaak niet onder stoelen of banken stak.

moeilijkste en meest stresserende periodes is van een doctoraat. Na vier jaren hard labeur om naar dit moment toe te werken is er echter toch iemand op een of andere manier in geslaagd om net tijdens deze laatste cruciale fase mijn hoofd op hol te doen slaan. Maar dit bleek uiteindelijk eerder een zegen te zijn dan een hinderpaal, waardoor deze bewering van mijn collega's absoluut niet opging in mijn geval (integendeel). Lieve Eline, ook al was het op de valreep, je was er bij op het belangrijkste moment en dat vond ik fijn.

En dan komen we uiteindelijk bij enkelen die eigenlijk niks -maar tegelijk ook alles- aan dit doctoraat hebben bijgedragen.

Allereerst veel dank aan Nele, mijn allerliefste petekind en oudste zus Sophie, 'de broeren' Thomas en Simon, en mijn jongste zusje Louise a.k.a. Wieze, voor alle fijne en gezellige momenten in Rumbeke en daarbuiten.

Dan is er nog iemand die ik heb moeten teleurstellen dat ik mijn doctoraat niet heb afgekregen in de 3 jaar die hij in gedachten had ('Duurt dat 4 jaar!? Zeg maar tegen die prof dat 3 jaar meer dan genoeg is.'), maar stiekem wel blij was dat er nog een sprankeltje hoop was om een van zijn zonen uiteindelijk toch 'Dr.' te zien worden (ook al was dat oorspronkelijk misschien in een andere context). Dan is er ook iemand die me tijdens mijn doctoraat gelukkig af en toe hielp herinneren dat 'geen resultaat ook een resultaat is'[3], en die me er zo nu en dan op wees dat er betere alternatieven zijn om geld te verdienen in plaats van een doctoraat in 'elektromechanica'[4], maar uiteindelijk wel heel fier was ondanks mijn atypische carrièrekeuze. En tot slot was er nog iemand die me altijd uitermate nuttige input gaf als ik vast zat met mijn werk ('heb je het al eens geprobeerd met determinanten?'), mij altijd de nodige complimentjes en erkenning gaf wanneer ik fier mijn afgewerkte papers liet zien ('zot ventje'), maar vooral een geweldige broer is.

Papa, mama en Jan, zoals gewoonlijk zonder veel woorden, maar oprecht: ik ben jullie heel dankbaar voor alles.

Ik zou deze thesis graag willen opdragen aan opa, van wie we met pijn in het hart afscheid hebben moeten nemen vorig jaar. Hij vroeg altijd vol interesse hoe het ging met mijn 'onderzoek in de hoorapparaten', waarna hij spontaan alle praktische problemen met zijn gehoorapparaat begon op te sommen. Opa, bedankt voor je eeuwige goedheid en positieve kijk op alles. Ik kon me geen betere peter voorstellen.

Tot slot richt ik mij tot diegenen die nog wat verder zullen lezen dan deze eerste vier bladzijden. Ik heb met hart en ziel gewerkt aan dit doctoraat, en heb een

[3]Een tip voor iedereen: beste pep-talk die je kan geven aan een onderzoeker die in een dipje zit!

[4]Lees: elektrotechniek.

ontzettend leerrijk traject ondergaan met veel ups en downs. Zoals elke onderzoeker is mijn grootste wens dan ook dat het hierbij niet stopt, en dat de kennis en de ideeën die in dit boek staan uiteindelijk ook anderen zullen inspireren. Ik hoop dan ook dat dit werk uiteindelijk iets kan betekenen voor toekomstige nieuwe boeiende technologieën. Al is het maar dat ene dominosteentje die de keten omver duwt, dat ene vonkje die de motor in gang zet...

Alexander Bertrand

Leuven, april 2011

Abstract

Recent academic developments have initiated a paradigm shift in the way spatial sensor data can be acquired. Traditional localized and regularly arranged sensor arrays are replaced by sensor nodes that are randomly distributed over the entire spatial field, and which communicate with each other or with a master node through wireless communication links. Together, these nodes form a so-called 'wireless sensor network' (WSN). Each node of a WSN has a local sensor array and a signal processing unit to perform computations on the acquired data. The advantage of WSNs compared to traditional (wired) sensor arrays, is that many more sensors can be used that physically cover the full spatial field, which typically yields more variety (and thus more information) in the signals. It is likely that future data acquisition, control and physical monitoring, will heavily rely on this type of networks. Most contributions in this thesis focus on (but are not limited to) the application of WSNs for distributed noise reduction in speech recordings. Noise reduction for speech enhancement is crucial in many applications such as hearing aids, mobile phones, video conferencing, hands-free telephony, automatic speech recognition, etc.

In this thesis, we develop novel signal and parameter estimation techniques that rely on distributed in-network processing, i.e., without gathering all the sensor data in a central processor as it is the case in centralized estimation algorithms. In WSNs, a distributed approach is often preferred, especially so when it is scalable in terms of its communication bandwidth requirement, transmission power and local computational complexity. In almost all distributed estimation techniques that are proposed in this thesis, the goal is to obtain the same estimation performance as in a centralized estimation algorithm. We distinguish between two different types of distributed estimation problems: signal estimation and parameter estimation. Both problems usually have to be tackled in very different ways. In distributed signal estimation, the number of estimation variables grows linearly with the number of temporal observations, i.e. for each sample time of the sensors, a new sample of the desired signal(s) has to be estimated. Iterative refinement of these signal estimates would require that intermediate signal estimates are retransmitted multiple times between the same node pairs, which is usually not feasible in real-time systems with high sampling rates. In

distributed parameter estimation problems on the other hand, the number of estimation variables are either fixed, i.e., it does not grow with the number of temporal observations, or the data acquisition happens at a very low sampling rate such that sufficient time is available to iteratively refine intermediate estimates.

In the context of distributed signal estimation in WSNs, we propose a distributed adaptive node-specific signal estimation (DANSE) algorithm, which operates in a fully connected WSN. The term 'node-specific' refers to the fact that each node estimates a different signal, although the desired signals of all nodes have to share a common low-dimensional signal subspace. In this case, DANSE significantly reduces the exchange of data between nodes, while still obtaining an optimal estimator in each node, as if all nodes have access to all the sensor signal observations in the network. In the original version of DANSE, the local fusion rules of each node are iteratively updated in a sequential round-robin fashion. The DANSE algorithm is then extended to the case where nodes update their local fusion rules simultaneously, which allows the algorithm to adapt more swiftly to changes in the environment. Both versions of the algorithm are then applied in a speech enhancement context. To this end, the algorithm is extended to a more robust version, to avoid numerically ill-conditioned quantities that often arise in such practical settings. The DANSE algorithm is also extended to operate in WSNs with a tree topology, hence relaxing the constraint that the network has to be fully connected, i.e., each node only has to communicate with nearby nodes. Finally, the DANSE algorithm is extended with node-specific linear constraints, yielding an optimal node-specific linearly-constrained minimum variance beamformer in each node.

In the second part of this thesis, we tackle distributed linear regression problems, based on distributed parameter estimation techniques. In particular, we focus on the case where the data or regression matrix is noisy, for which traditional least-squares methods yield biased results. To reduce this bias, we propose two novel methods. The first one is a distributed version of the well-known total least squares estimation technique, which yields unbiased estimates if the regressor noise is white. A second method, that can also cope with colored noise, is based on a bias-compensated recursive least squares algorithm with diffusion adaptation. This algorithm is analyzed in an adaptive filtering context, where it is demonstrated that the cooperation between nodes indeed reduces the bias, and furthermore reduces the variance of the local parameter estimates at each node.

In the third part of this thesis, we propose two supporting techniques that can be used in WSNs for (acoustic) signal estimation. The first one is an energy-based multi-speaker voice activity detection algorithm, that aims to track the individual speech power of multiple speakers talking simultaneously. Finally, we propose a technique for sensor subset selection, which is an efficient greedy approach to select the subset of sensors that contribute the most to the

estimation. The other nodes can then be put to sleep to save energy. This method also yields efficient formulas to compute optimal fall-back estimators in the case of link failure.

Korte Inhoud

Recente academische ontwikkelingen hebben een paradigmaverschuiving teweeg gebracht in de manier waarop we spatiale sensormetingen kunnen verkrijgen. Traditionele gelokaliseerde en regelmatig geordende sensorroosters zullen in de toekomst vervangen worden door sensoren die willekeurig over de geobserveerde omgeving verspreid worden, en die draadloos met elkaar kunnen communiceren. Dit is het domein van de zogenaamde draadloze sensornetwerken (*wireless sensor networks*, of WSNs). Een WSN bestaat uit sensornodes die elk over een sensor(rooster) en een verwerkingseenheid beschikken om de geobserveerde data te verwerken. Het voordeel in vergelijking met traditionele (bedrade) sensorroosters is dat er meer sensoren kunnen gebruikt worden die fysisch een veel grotere omgeving omspannen, wat typisch meer variëteit (en dus meer informatie) in de opgemeten signalen oplevert. Er wordt verwacht dat toekomstige data acquisitie en regel- en observatiesystemen veelvuldig gebruik zullen maken van dergelijke sensornetwerken. De meeste contributies in dit doctoraatsproefschrift zijn gericht op (maar niet gelimiteerd tot) WSNs voor ruisonderdrukking in spraakopnames. Ruisonderdrukking is cruciaal in vele spraaktoepassingen zoals gehoorapparaten, mobiele telefonie, video conferenties, handenvrije telefonie, automatische spraakherkenning, etc.

In dit doctoraatsproefschrift ontwikkelen we nieuwe gedistribueerde signaal- en parameterschattingstechnieken voor WSNs, waarbij de sensordata binnen het netwerk zelf wordt verwerkt, i.e., door de sensornodes zelf, zonder alle sensorobservaties te verzamelen in een centrale verwerkingseenheid zoals in gecentraliseerde schattingstechnieken. Gedistribueerde verwerking biedt vaak schalingsvoordelen met betrekking tot communicatiebandbreedte, transmissievermogen en lokale rekenkracht, en geniet daarom meestal de voorkeur. In bijna alle voorgestelde gedistribueerde schattingstechnieken is het doel om dezelfde schattingsperformantie te behalen als in een gecentralizeerd algoritme. We onderscheiden twee verschillende types schattingsproblemen: signaalschatting en parameterschatting. Beide problemen worden meestal op sterk verschillende manieren opgelost. In gedistribueerde signaalschatting neemt het aantal schattingsvariabelen lineair toe met het aantal sensorobservaties, d.w.z., voor elk bemonsteringstijdstip aan de sensoren moet een nieuw monster van het

gewenste signaal geschat worden. Iteratieve verbetering van deze signaalschattingen zou dan betekenen dat tussentijdse schattingen van dezelfde signalen meerdere keren moeten worden uitgewisseld tussen hetzelfde paar nodes, wat meestal niet mogelijk is in real-time systemen met hoge bemonsteringsfrequenties. In gedistribueerde parameterschatting is de situatie anders. Ofwel ligt het aantal schattingsvariabelen vast, ofwel gebeurt de data acquisitie aan een trage bemonsteringssnelheid zodat er genoeg tijd is om tussentijdse schattingen iteratief te verbeteren.

In het kader van gedistribueerde signaalschatting in WSNs stellen we een gedistribueerd adaptief node-specifiek signaalschattingsalgoritme voor ('*distributed adaptive node-specific signal estimation*' of DANSE), dat eerst wordt beschreven voor volledig geconnecteerde WSNs. De term '*node-specific*' duidt aan dat elke node een ander signaal schat, hoewel er verondersteld wordt dat deze signalen een gemeenschappelijke laagdimensionele signaalruimte delen. Indien hieraan voldaan is, dan kan DANSE de uitwisseling van data tussen de nodes sterk reduceren, en toch de optimale schatter bekomen in elke node, alsof alle nodes toegang hebben tot alle sensorsignalen in het volledige netwerk. In de oorspronkelijke versie van DANSE worden de lokale schatters in elke node iteratief en sequentieel aangepast. Het DANSE algoritme wordt daarna uitgebreid zodat nodes hun lokale schatters gelijktijdig kunnen aanpassen, wat toelaat om veel sneller te reageren op veranderingen in de omgeving. Beide versies van het algoritme worden dan toegepast in een spraakverbeteringscontext. Hiervoor wordt het DANSE algoritme uitgebreid naar een robuustere versie om numerieke problemen -die regelmatig opduiken in dergelijke praktische opstellingen- te vermijden. Het DANSE algoritme wordt daarna ook verder uitgebreid naar netwerken met een boomtopologie, zodanig dat elke node niet per se hoeft te communiceren met elke andere node in het netwerk. Een laatste uitbreiding van DANSE bestaat erin dat er node-specifieke lineaire beperkingen kunnen opgelegd worden in elk lokaal schattingsprobleem.

Een tweede deel van dit doctoraatsproefschrift richt zich op gedistribueerde parameterschatting, in het bijzonder op lineaire regressieproblemen waar de data- of regressiematrix met ruis gecontamineerd is, waarvoor traditionele kleinste kwadratenschatters een *bias* vertonen. Om deze bias the reduceren, stellen we twee nieuwe methoden voor. De eerste is een gedistribueerde versie van *total least squares* schatting, die de bias elimineert indien de regressieruis wit is. Een andere methode, die ook voor gekleurde ruis werkt, past bias-compensatie toe op een recursief kleinste kwadratenalgoritme met diffusie adaptatie. Dit algoritme wordt geanalyseerd in een adaptieve filtering context, en we tonen aan dat samenwerking tussen de nodes inderdaad de bias reduceert, en bovendien de variantie op de lokale parameterschattingen verkleint.

In het laatste deel beschrijven we twee ondersteunende technieken voor (akoestische) signaalschatting in WSNs. De eerste is een energie-gebaseerde multispreker spraakdetector, die als doel heeft om het spraakvermogen van indivi-

duele sprekers, die tegelijk aan het praten zijn, te schatten. Tenslotte stellen we een efficiënte *greedy* sensorselectietechniek voor die de set van sensors selecteert die het meeste invloed hebben op de finale signaalschatting. De andere -minder belangrijke- sensornodes kunnen dan uitgezet worden om energie te besparen. Deze methode geeft als bijproduct ook efficiënte formules om de optimale schatter te herberekenen indien er plots een draadloze link uitvalt.

Glossary

Mathematical operators and constants

$\forall$	for all
$\exists$	there exists
$\in$	belongs to
$\subset$	is subset of
$\approx$	approximately equal to
$\triangleq$	defined as
$\ll$	much less than
$\gg$	much greater than
$\mathbf{X} \succeq \mathbf{Y}$	$(\mathbf{X} - \mathbf{Y})$ is positive (semi)definite
$\odot$	Hadamard product (elementwise multiplication)
$\otimes$	Kronecker product
$(\cdot)^*$	complex conjugation
$(\cdot)^T$	matrix transpose
$(\cdot)^H$	matrix conjugate transpose
$(\cdot)^{-1}$	matrix inverse
$(\cdot)^\dagger$	(Moore-Penrose) pseudoinverse
$\rho(.)$	spectral radius of a matrix
$\lambda_{\min}(.)$	minimal eigenvalue
rank(.)	rank of a matrix
$\mathcal{D}\{\mathbf{X}\}$	sets all off-diagonal entries of the matrix $\mathbf{X}$ to zero
$\mathbf{I}$	identity matrix
$\mathbf{1}$ or $\mathbb{1}$	vector containing only unity entries
$\mathbf{O}$	zero matrix
Tr{.} or $tr(.)$	trace of a matrix, i.e., sum of diagonal constants
blockdiag{.}	block-diagonal matrix with arguments on block-diagonal
col{.}	column vector based on stacked arguments
diag{.}	diagonal matrix with arguments on diagonal
$\cap$	set intersection
$\cup$	set union
$\setminus$	set exclusion
$\wedge$	logic 'and'

$\|\cdot\|$	absolute value (real numbers)
	or modulus (complex numbers)
	or cardinality (set)
$\|\cdot\|$ or $\|\cdot\|_2$	Euclidian vector norm, L2 norm
$\|\cdot\|_F$	Frobenius matrix norm
$\mathbb{N}$	the set of natural numbers
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_0^+$	the set of strictly positive real numbers
$\mathbb{C}$	the set of complex numbers
$\mathbb{R}^{M\times N}$	the set of real $M \times N$ matrices
$\mathbb{C}^{M\times N}$	the set of complex $M \times N$ matrices
$\Re z$	real part of complex number z
$\Im z$	imaginary part of complex number z
∇J	gradient of function J
x mod a	x modulo a (remainder after dividing x by a)
sup{.}	supremum
$\min\{x, y\}$	minimum of scalars x and y
$\max\{x, y\}$	maximum of scalars x and y
$\min_x$	minimize over x
$\max_x$	maximize over x
$E\{.\}$	expected value operator
$Pr(A)$	Probability that event A happens

Acronyms and Abbreviations

AD-MoM	alternating direction method of multipliers
AGSSS	adaptive greedy sensor subset selection
AO	alternating optimization
APA	affine projection algorithm
AR	auto-regressive
ASR	automatic speech recognition
ATC	adapt then combine
AWGN	additive white Gaussian noise
BC-RLS	bias-compensated recursive least squares
BHA	binaural hearing aid
BLUE	best linear unbiased estimator
BP	belief propagation
BSS	blind source separation
CA	consensus averaging
CE	compress-estimate
CGS	centralized Gauss-Seidel
CO	constrained optimization
CTA	combine then adapt

DANSE	distributed adaptive node-specific signal estimation
dB	decibel
DB-MWF	distributed multi-channel Wiener filter
DBSA	dual based subgradient algorithm
DEF	direct estimation filter
DFT	discrete Fourier transform
DKLT	distributed Karhunen-Loeve transform
DRR	direct-to-reverberant ratio
DSP	digital signal processing
D-TLS	distributed total least squares
EC	estimate-compress
e.g.	*exempli gratia*: for example
FC	fusion center
GTLS	generalized total least squares
HA	hearing aid
HINT	hearing-in-noise test
Hz	Hertz
ICA	independent component analysis
i.e.	*id est*: that is
IP	interior point
IP-KKT	interior point Karush-Kuhn-Tucker
KLT	Karhunen-Loeve transform
LASSO	least-absolute shrinkage and selection operator
LC-DANSE	linearly constrained DANSE
LCMV	linearly constrained minimum variance
LLS	linear least squares
LMMSE	linear minimum mean squared error
LMS	least mean squares
LPC	linear predictive coding
LS	least squares
MC	Monte-Carlo
MIMO	multiple-input multiple-output
MMSE	minimum mean squared error
M-NICA	multiplicative non-negative independent component analysis
MSD	mean square deviation
MSE	mean squared error
MVUE	minimum variance unbiased estimator
MWF	multi-channel Wiener filter
NBSS	non-negative blind source separation
NICA	non-negative independent component analysis
NMF	non-negative matrix factorization
NPCA	non-negative principal component analysis
PCA	principal component analysis
pdf	probability density function
QCQP	quadratically constrained quadratic program

Q.E.D.	*quod erat demonstrandum*: what was required to be proved
R1-MWF	rank-1 SDW-MWF
rA-DANSE	relaxed asynchronous-DANSE
R-DANSE	robust-DANSE
RFC	receiver feedback cancellation
RIR	room impulse response
RLS	recursive least squares
rS-DANSE	relaxed simultaneous-DANSE
s.t.	subject to
S-DANSE	simultaneous-DANSE
SDP	semidefinite program
SDR	signal-to-distortion ratio or semidefinite relaxation
SDW-MWF	speech-distortion-weighted MWF
SER	signal-to-error ratio
SIMO	single-input multiple-output
SNR	signal-to-noise ratio
SSS	sensor subset selection
SVD	singular value decomposition
T-DANSE	tree-DANSE
TDOA	time difference of arrival
TFC	transmitter feedback cancellation
TLS	total least squares
VAD	voice activity detection
WASN	wireless acoustic sensor network
w.l.o.g.	without loss of generality
WSN	wireless sensor network

Contents

IV Supporting Techniques for Signal Estimation

Part I

Introduction

Chapter 1

Introduction and Overview

This thesis addresses crucial problems in the domain of signal and parameter estimation in wireless sensor networks (WSNs), and wireless acoustic sensor networks (WASNs) in particular. Most chapters focus on (but are not limited to) the application of WASNs for distributed noise reduction in speech recordings. Noise reduction for speech enhancement is important in many applications such as hearing aids, mobile phones, video conferencing, hands-free telephony, automatic speech recognition, etc. By using WASNs, many more microphone signals become available, which can greatly improve the noise reduction performance in these applications.

In **Part I** (this introduction), we first explain the concept of wireless sensor networks, together with their major advantages and disadvantages, and we address some important aspects in the algorithm design for estimation in WSNs (Section 1.1). We then describe some basic concepts and state-of-the-art techniques for acoustic noise reduction for speech enhancement (Section 1.2). These acoustically-oriented problem statements will serve as target applications for many of the distributed algorithms that are described in this thesis. We then review some general estimation problems for WSNs, and we briefly describe state-of-the-art distributed estimation techniques to solve them (Sections 1.3 and 1.4). Due to the extensive literature on sensor networks, and the large variety in applications and estimation problems, we will only restrict ourselves to certain types of problems that are either related to the contributions in this thesis, or that allow us to position these contributions in the broad spectrum or classification of distributed estimation problems. Throughout the introduction, we will often comment on how the addressed techniques relate to the work in this thesis. In Section 1.5, we define the problem statement and the challenges that are addressed in this thesis. In Section 1.6, we provide a brief chapter-by-chapter description of the main contributions. The introduction ends with an

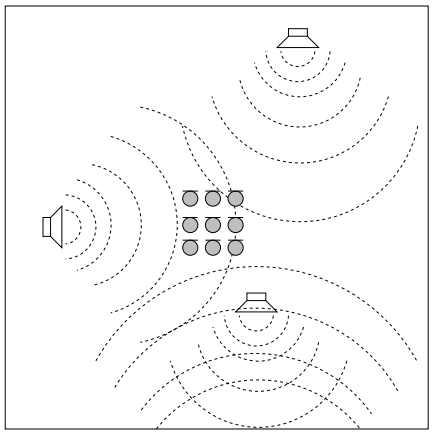


Figure 1.1: Schematic example of a local regularly arranged sensor array.

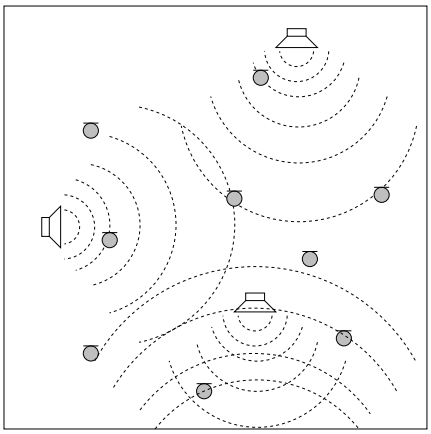


Figure 1.2: Schematic example of a randomly distributed sensor array.

overview of the publications that are included in the remaining chapters.

In **Part II** of this thesis, we focus on contributions that involve distributed *signal estimation* problems. In **Part III**, we propose techniques for distributed linear *parameter estimation* for WSNs with noisy observations. In **Part IV** we provide algorithms that can serve as supporting techniques for signal estimation with spatially distributed sensors. Conclusions and comments on future research challenges are given in **Part V**.

1.1 Wireless Sensor Networks (WSNs)

1.1.1 Background and Definition

Recent academic developments in the area of digital signal processing (DSP) initiated a paradigm shift in the way sensor data can be acquired. For temporal data acquisition, new and promising sampling techniques have been discovered that break with the famous Nyquist-Shannon sampling theorem [1]. At the same time, also spatial data acquisition is changing. Traditional localized and regularly arranged sensor arrays (Fig. 1.1) are replaced by randomly placed sensors, distributed over the entire spatial field (Fig. 1.2). This is the area of 'wireless sensor networks' (WSNs), which saw a tremendous boost during the last couple of years [2–4]. It is likely that future data acquisition, control and physical monitoring, will heavily rely on this type of networks.

A WSN consists of a set of sensor nodes, randomly distributed over an environment, which communicate with each other or with a master node through

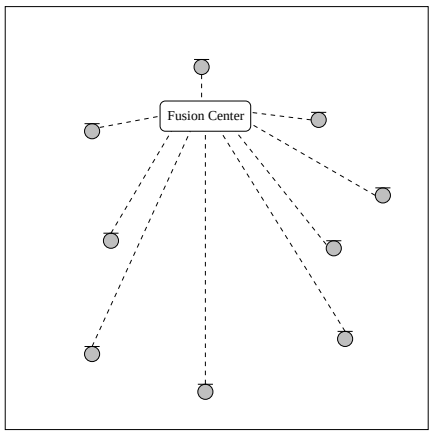


Figure 1.3: Schematic example of centralized data fusion by means of a fusion center.

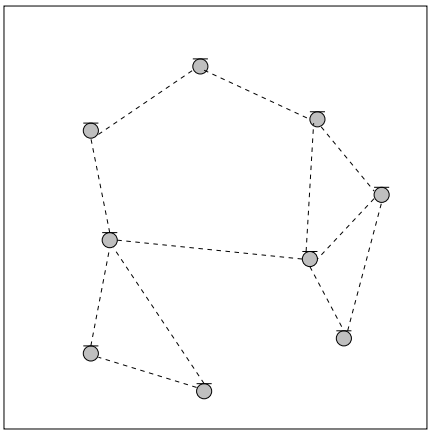


Figure 1.4: Schematic example of distributed data fusion in a WSN with an ad hoc topology.

wireless communication links. Each node has a local sensor (array) and a signal processing unit to perform computations on the acquired data. The advantage compared to traditional (wired) sensor arrays, is that many more sensors can be used that physically cover the full spatial field, which typically yields more variety (and thus more information) in the signals. A general objective is to utilize all sensor signal observations available in the entire network to perform a certain task, such as the estimation of a parameter or signal, or the detection of a physical phenomenon (the latter is often referred to as distributed detection or decision making). In this thesis, we will focus on the former, i.e., distributed estimation.

One important challenge in designing algorithms for WSNs, is that the acquired data from all the nodes must somehow be combined and processed to generate a useful output. This process is often referred to as *data fusion*, which can happen in a centralized fashion (Fig. 1.3), where all the nodes send their raw data to a master node who does all the processing (the 'fusion center'), or in a distributed fashion (Fig. 1.4), where the processing is shared between all the nodes, and the nodes in the network exchange data with each other. Hybrid cases are also possible, where some local processing of the observed data is performed at each node, e.g., for compression, and then transmitted to a fusion center.

A centralized approach may require a large communication bandwidth and transmission power. It also requires a dedicated device (the fusion center), which must be able to receive and process many different communication channels in real-time. This is often a limiting factor, especially when operating at

high sampling rates. The required communication bandwidth, and the computational power at the fusion center may increase drastically with the number of sensor nodes[1]. A distributed approach is therefore often preferred, especially so when it is scalable in terms of its communication bandwidth requirement and local computational complexity. However, the design of such distributed signal processing algorithms is a lot more challenging, and usually one needs to settle for a suboptimal solution compared to the centralized case. Indeed, in the centralized case, all data is available at one place, which allows to compute variables that often cannot be computed in a distributed case, such as the full cross-correlation between all the sensor signal observations. Therefore, a centralized approach in general yields better estimates, which can be used as a reference point to assess the performance of distributed estimation algorithms.

1.1.2 Design Aspects

In the algorithm design to solve estimation problems in WSNs, several aspects should be taken into consideration, depending on the requirements of the target application:

- ***Estimation performance***: The main goal of the network is to obtain a good estimate of a certain parameter or signal, based on as much observations as possible. The estimation performance of the algorithm is therefore the main design parameter, and it is often highly influenced by the choices that are made with respect to the other design parameters that are mentioned in the sequel.

- ***Communication bandwidth***: It is important that the network can operate with a small communication bandwidth. A centralized approach, where raw sensor data is transmitted to a fusion center, can be viewed as a worst-case scenario with respect to bandwidth usage. If nodes only share data with their closest neighbors (in a distributed setting), less transmission power is required and spatial reuse of the frequency spectrum is possible. Furthermore, to reduce the required communication bandwidth, local compression of sensor data is of great importance. Compression and estimation are often jointly attacked in WSNs, instead of treating them as independent problems.

- ***Energy awareness***: Since the nodes of a WSN are usually powered by batteries and sometimes even by energy scavenging[2], it is important that the sensor nodes do not consume too much energy. Therefore, the *computational complexity* of the algorithm should be as low as possible.

[1] For example, in multi-channel Wiener filtering (see Subsection 1.2.2), the computational power increases quadratically with the total number of microphones.

[2] Energy scavenging is the process by which energy is derived from external sources (e.g., solar power, thermal energy, wind energy, kinetic energy, etc.), captured, and stored.

Furthermore, the required *transmission power* is also an important factor[3]. The latter depends on the network topology and the distance (and physical obstacles) between the different nodes. A fully connected topology or a star topology[4] are usually considered as the worst-case scenario in terms of transmission power. The best approach with respect to transmission power is the nearest-neighbor-based topology, where nodes only share data with nodes that are close by and not obstructed by obstacles.

- ***Scalability***: A distributed algorithm is scalable if the communication bandwidth and/or the power consumption per node does not or only partially depend on the total amount of nodes in the network. Basically, it means that adding an extra sensor has no impact on the computational load or transmission power of the nodes that are not directly connected to this extra node. Scalability is very important in large-scale networks, or networks for signal estimation at high sampling rates (where communication bandwidth usage and power consumption are a limiting factor, even in small networks). Centralized algorithms or algorithms for fully connected networks usually do not scale well (although this can be improved in some cases, see Chapter 2). Distributed algorithms that allow simply connected[5] or ad hoc network topologies are usually scalable in both communication bandwidth and power consumption.

- ***Robustness to noisy communication links***: The data that is transmitted between the nodes is usually compressed and quantized, which introduces distortion (in the case of lossy compression) and quantization noise. Furthermore, due to interference and fading, bit errors can occur during the transmission of data. Depending on the quality of the links, it can be important to incorporate these aspects in the design of the estimation algorithm, to make it more robust to distortions on the transmitted data.

- ***Adaptivity***: Adaptivity refers to the fact that the network or the algorithm can adapt to changes in the environment, such as changes in positions of the nodes, changes in the topology of the network, or changes in the physical processes that are sensed by the network. A fully adaptive algorithm also has the facilitating property that it does not require a prior training or calibration phase before operation of the algorithm. This is particularly interesting for WSNs with an ad hoc deployment. A fixed algorithm (without adaptation) usually relies on prior knowledge that cannot be measured during operation of the algorithm, such as the

[3] It can be shown that the energy required to transmit 1kb over 100m (i.e., 3 J) is equivalent to the energy required to execute 3 million instructions [5, 6].

[4] This corresponds to a centralized approach, where all the nodes are connected with a single master node, who forms the center of the star.

[5] Simply connected networks are networks that are not fully connected.

cross-correlation between sensor signals of nodes that are not directly connected by a wireless link. Hybrid cases are also possible, where the algorithm can adapt to certain changes in the environment, but not to all of them. For example, the noise scenario is sometimes assumed to be fixed, while the statistics of the target sources can change during operation of the algorithm.

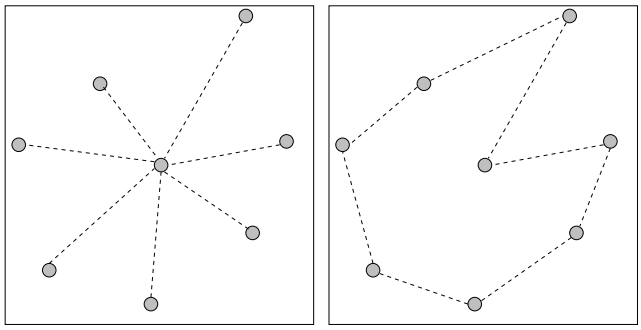
- ***Convergence speed***: In iterative (adaptive) algorithms, the convergence speed is important when the environment can change rapidly or abruptly. To track or respond to these changes, the algorithm must have good convergence properties.

- ***Blindness***: In many cases, the positions of the sensor nodes are not known a priori, due to the random placement of the sensor nodes. For some estimation tasks, such as localization or signal estimation based on spatial separation (beamforming), supporting algorithms are often required to estimate node and/or source positions. In the context of WSNs, blind algorithms that do not require this side information are usually preferred.

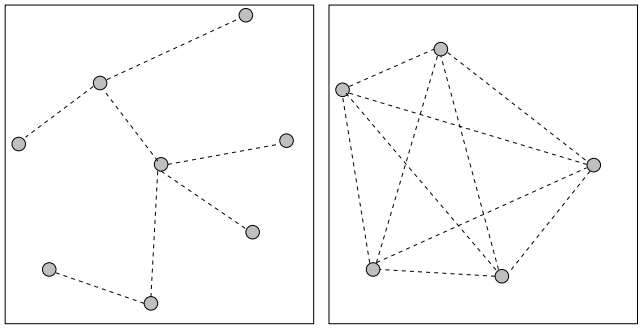
- ***Network topology***: Algorithms for WSNs can be designed for specific network topologies, such as a centralized (star) topology [7–12], a ring topology [13, 14], a tree topology (see Chapter 5), a fully connected topology (see Chapter 2), etc. These four common topologies are depicted in Fig. 1.5. Setting up such a predefined topology usually requires some upper-layer protocol. Furthermore, such algorithms are often suboptimal in the sense that they do not exploit all the available links (due to link pruning to obtain the desired topology), or because they require extra links over long distances or through obstacles, which usually have a very bad quality. Therefore, algorithms that do not make any assumptions on the topology are usually preferred, especially in ad hoc deployed WSNs.

- ***Self-healing properties***: The communication links in a WSN are often not very robust, due to the low-power communication. This often introduces significant packet loss, or even permanent failing of certain links. The algorithm must therefore be able to cope with dynamic configurations of the network, such that there is no single point of failure. This 'self-healing' property is closely related to the adaptivity of the algorithm, and the prior assumptions on the network topology. For example, some algorithms require a so-called Hamiltonian cycle, i.e., a path in the network that starts and ends in the same node, and visits every node only once[6] (see e.g. [13]). If a certain link on this cycle fails, a new cycle needs to be determined.

[6] This corresponds to pruning the network to a ring topology (Fig. 1.5(b)).



(a) Centralized or star topology (b) Ring topology



(c) Tree topology (d) Fully connected topology

Figure 1.5: Special network topologies.

- ***Uniformity***: In some applications, it is important that each node essentially performs the same task. Often, this requirement has economic reasons, as it is cheaper to produce 'many of the same'. However, it can also be imposed to avoid points of failure in the network. If certain nodes have important function or specific roles in the network, the failure of these nodes can have severe consequences for the performance of the WSN.

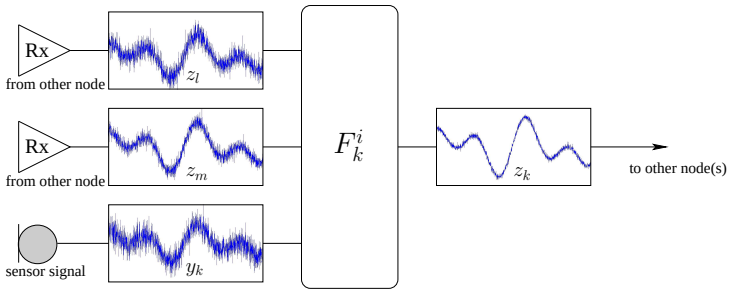
- ***Sensor subset selection***: In many cases, it is not worth it to use the data of all the nodes of the network. Often a good estimate can be computed by only using a subset of nodes that have the most useful data. The other (less useful) sensor nodes can then be put to sleep to save energy. The selection of a useful subset is usually a difficult problem on its own, which requires supporting algorithms that can either run independently from the estimation algorithm, or that can use side information from the estimation algorithm.

- ***Clock synchronization***: A critical component in WSNs is the clock synchronization. Since each node has its own clock, and since each clock has imperfections in its oscillator, the length of the clock cycles will be slightly different at each node (usually around 40 μs difference per second [6]). This results in sampled signals that drift away from each other when time flows, with a speed that depends on the sampling frequency and the clock imperfections. This signal drift can be very harmful for both the estimation algorithm and the communication protocol in the wireless links. A good clock synchronization algorithm should therefore provide a common time frame for all the nodes, which is essential for many algorithms. These supporting clock synchronization algorithms can be classified in two different types of algorithms. The first one is based on time stamps, often referred to as *packet coupling*, which is fully implementable in software (see [6] for an overview). The other class consists of *pulse-coupling* techniques, which use signal injection on the physical communication layer [15–17].
  Many estimation algorithms are very sensitive to clock drift, but some can cope with significant clock drift and only require minor synchronization constraints. The latter class usually contains all the energy-based methods. Since the used data then consists of energy observations, which are squared averages over blocks of many data samples, only very large clock drifts will have a significant impact.

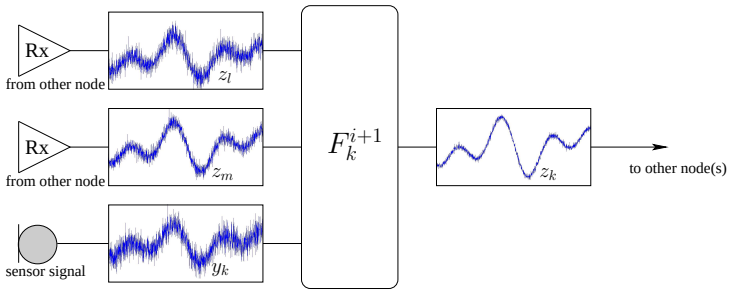
1.1.3 Signal vs. Parameter Estimation

In this thesis, we distinguish between two types of distributed estimation problems: *signal* estimation and *parameter* estimation. Although both terms are often used interchangeably in the WSN sensor network literature, it is important to make this distinction since both problems usually need to be tackled in very different ways.

In distributed *signal* estimation, the goal is to estimate a signal in real-time, while suppressing interfering noise. This means that the number of estimation variables grows linearly with the number of temporal observations, i.e. for each sample time of the sensors, a new sample of the desired signal(s) needs to be estimated. In this case, fused or compressed sensor observations are exchanged between nodes, rather than derived parameters (as it is the case in parameter estimation). The estimation then usually relies on (lossy) 'compress-and-fuse' techniques [7–12], fusion of sensor data within a one-hop neighborhood [18], or linear spatio-temporal filtering (beamforming), as often used in signal enhancement [19]. Distributed signal estimation often assumes some (short-term) stationarity of the signal statistics or the spatial characteristics, such that fusion rules are not sample-specific, i.e., the same fusion rules are used for observations at different time instances. In the case of adaptive signal estimation



(a) Signal estimation (iteration i)



(b) Signal estimation (iteration $i + 1$)

Figure 1.6: Two subsequent signal estimation iterations at node k.

algorithms, the fusion rules can be iteratively and recursively updated, based on previous observations, to improve the overall estimation performance for future signal observations. This means that the algorithm does not iterate over the estimates themselves, but over the local fusion rules at the nodes. Iterative refinement of the actual estimates, as it is often the case in parameter estimation, would require that estimates of the same signal are retransmitted multiple times between the same node pairs, which significantly increases the communication bandwidth. Although the latter could improve the estimation performance, it is usually not feasible in real-time systems with high sampling rates. The wireless links of a WSN for signal estimation usually need to be quite robust, since packet loss can result in instantaneous signal degradation at the output.

A typical distributed signal estimation framework is schematically depicted in Fig. 1.6 for a single sensor node with label k. The sensor signal y_k is fused with the signals z_l and z_m that node k receives from neighboring nodes l and m, respectively. The output signal z_k is then forwarded to the other nodes in the neighborhood of node k. It should be noted that only the fusion rule F is refined over the different iterations, which only has an effect on future

signal observations. Previous (fused) signal observations are not retransmitted or re-estimated.

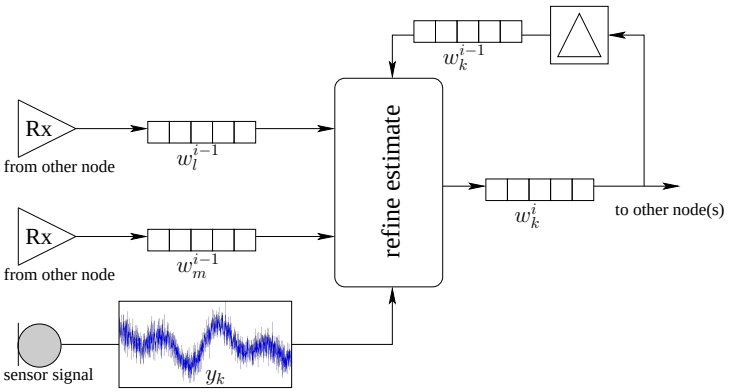
In distributed *parameter* estimation problems on the other hand, the number of estimation variables are either fixed, i.e., it does not grow with the number of temporal observations, or the data acquisition happens at a very low sampling rate such that sufficient time is available to iteratively refine intermediate estimates [13, 20–29]. Often, only parameters that are derived from the sensor observations (e.g., a regression vector) are exchanged between nodes, without sharing actual sensor observations. Because these parameters usually change rather slowly over time (compared to the sampling clock), this allows for iterative and incremental strategies. The latter also holds in sensing applications where the sampling rate is low, e.g., for the estimation of temperature, chemical compositions, wind speed, humidity, etc. Furthermore, the exchange of parameters or low-data-rate measurements typically requires less communication bandwidth, such that the network usually consumes less energy than in signal estimation applications, and the nodes can be kept small, cheap and possibly even disposable.

A typical distributed parameter estimation framework is schematically depicted in Fig. 1.7 for a single sensor node with label k, where the goal is to estimate a latent parameter w. The local estimate at node k is denoted by w_k. At iteration i, node k refines this estimate, based on its previous estimate w_k^{i-1}, and the estimates w_l^{i-1} and w_m^{i-1} that node k has received from nodes l and m, respectively. If new sensor data is obtained, this new information can also be incorporated in the new estimate. The refined estimate w_k^i is then transmitted to other nodes in the neighborhood, who will incorporate this in their local estimate in the next iteration. It should be noted that the iterations are now performed directly on the estimated parameter, which is retransmitted and re-estimated multiple times.

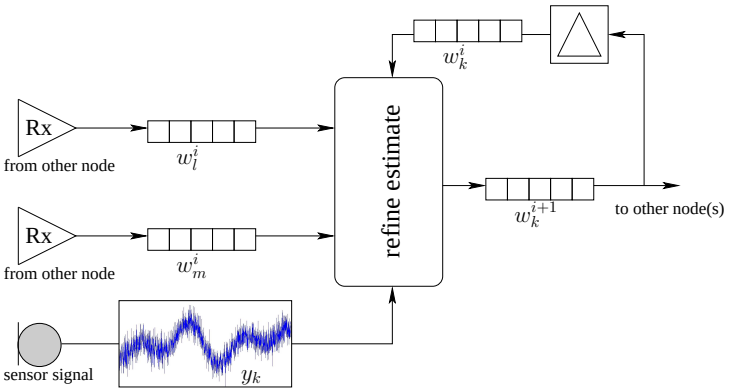
This thesis contains contributions for both types of estimation problems. Signal estimation for WSNs is addressed in **Part II**, and parameter estimation in **Part III**.

1.1.4 Wireless Acoustic Sensor Networks (WASNs)

Wireless sensor networks can also be used for acoustical applications, and then the network consists of randomly distributed microphones. This is often referred to as wireless acoustic sensor networks (WASNs). However, the high data rates and the rapidly changing characteristics of typical audio signals (e.g. speech signals) make the use of WSNs for acoustical applications very challenging. As a result, the existing literature on WASNs is still very limited. However, many important acoustical problems can significantly benefit from spatially distributed microphone arrays (some examples are source lo-



(a) Parameter estimation (iteration i)



(b) Parameter estimation (iteration $i + 1$)

Figure 1.7: Two subsequent parameter estimation iterations at node k.

calization, acoustic noise reduction, blind source separation, speech analysis, voice activity detection, etc.). Traditional (wired) microphone arrays sample the spatial acoustic field only locally (see Fig. 1.1), and then the array is often at a large distance from the relevant sound sources, resulting in signals with a low signal-to-noise ratio (SNR) and low direct-to-reverberant ratio (DRR). As a rule of thumb, the sound level decreases by 6 dB for each doubling of the distance between the microphone and the sound source[7]. Furthermore, the physical size of the array and the number of available microphones are often limited due space or power constraints imposed by the target application. For example, only two or three microphones can fit in a hearing aid, and the available power is limited due to the small batteries.

[7]This is not always true in practice. In particular, if there is a lot of reverberation and/or if the source-microphone distances are large, this model does not hold anymore.

With a WASN, many more microphone signals become available, at places where it is difficult or undesirable to place wired microphones. Furthermore the microphones physically cover a much larger area, which increases the probability that a subset of microphones is close to a relevant sound source (see Fig. 1.2). If this is a desired sound source, this will yield recordings with a high SNR and DRR. Nodes that are close to an interfering sound source provide good noise references. In scenarios where some of the source positions are known a priori, the microphones can be placed strategically near these sources. For example, they can be placed close to noisy machinery or equipment, or they can be pinned on the shirt of desired speakers. Because of the aforementioned advantages, since small microphones can now be produced at low cost, and since the computational power exponentially increases over time, it is believed that WASNs will soon become very popular in both the academic and industrial sector.

Originally, WASNs were only used for localization of sound sources with (centralized) methods based on long-term sound energy measurements, hence avoiding the problems with large temporal variability of sound signals [30–32]. However, also spatio-temporal correlation methods for localization in low-reverberant scenarios were developed, e.g., [33, 34]. In the context of noise reduction, only simple heuristic methods have been developed. In [34], a suboptimal noise reduction scheme is described, based on a hierarchy of cascaded beamformers, distributing the computational load evenly over the different nodes. In [35], a technique is proposed for SNR-based spectral combining of two or more microphone signals that are recorded at significantly different positions. In the context of hearing aids (HAs), systems are tested where a remote FM microphone is used as a direct input for the HA, instead of the local microphones in the HA itself [36, 37]. This is useful, for example, in a classroom scenario where the lecturer's microphone can be directly connected with a HA through a wireless link. However, since only the unprocessed remote microphone signal is played at the HA, the listener loses all other acoustic information about the environment. Voice activity detection (VAD) with distributed sensor nodes that transmit local decisions to a fusion center, has been considered in [38]. Finally, the influence of clock drift and some synchronization algorithms have been considered for some well-known acoustic problems such as blind source separation and echo cancellation [17, 39].

In the sparse literature on WASNs, it is almost always assumed that a fusion center is available. Truly distributed algorithms for WASNs only started to emerge during the past four years. The distributed multi-channel Wiener filter (DB-MWF) [40] was one of the first practical distributed acoustic noise reduction algorithms (see Subsection 1.3.3). It was developed for a binaural hearing aid setting, where a hearing aid is worn at both ears, both exchanging (compressed) microphone signals through a wireless link. This is essentially a 2-node

WASN. The DB-MWF algorithm forms the basis of the DANSE[8] algorithm, which is one of the main contributions in this thesis. An important target application of DANSE is speech enhancement in WASNs, e.g., for automatic speech recognition with spatially distributed microphone nodes, noise reduction in hearing aids or cochlear implants, audio surveillance in noisy buildings, hands-free telephony, etc. It is also an important enabler for so-called 'ambient intelligence' [41], where sensing and computing is (invisibly) distributed over an area, and where the environment is aware of the presence and needs of the user.

1.2 Acoustic Signal Estimation Problems

Before continuing our overview of state-of-the-art estimation techniques for WSNs, we first have to address some basic concepts and algorithms in the field of speech enhancement. The reason is that many contributions in this thesis were implicitly designed for distributed noise reduction in WASNs. The acoustically-oriented problem statements described in this section will therefore often appear as target applications for the algorithms that are described in this thesis.

1.2.1 Noise Reduction for Speech Enhancement

Noise reduction algorithms can significantly improve speech understanding in background noise, which is crucial in many speech recording applications, such as hearing aids, mobile phones, video conferencing, hands-free telephony, automatic speech recognition, etc.

Noise reduction algorithms for speech enhancement can be classified in single-microphone techniques and multi-microphone techniques. Single-microphone techniques can only exploit spectral characteristics of the noise and the target speech. Basically, their goal is to suppress frequencies where the noise is dominant over the speech. This will always introduce a significant temporal and/or spectral distortion in the desired speech signal, and this distortion usually increases with the amount of noise that is suppressed. Furthermore, single-microphone techniques do not work well if the noise is non-stationary.

In this thesis, we will focus on multi-microphone techniques. Their major advantage is that they can also exploit spatial characteristics of the acoustic scenario, in addition to the spectral characteristics of the sources. Since the target speech source and the noise sources usually have different positions, they can be spatially separated. These algorithms exploit the spatio-temporal

[8] DANSE = Distributed Adaptive Node-specific Signal Estimation.

(cross-)correlation between all available microphone signals to compute a good signal estimate of the target source. Due to the close relationship with the literature on antenna arrays, they are often referred to as beamforming techniques, since they basically 'steer a beam' in the direction of a target source, while suppressing sounds from other directions.

There is a vast amount of literature on multi-microphone noise reduction or acoustical beamforming. Many beamformers assume a fixed regularly arranged microphone array with accurately known microphone positions, and they usually also require knowledge of the direction of the desired sound source. These techniques often have the disadvantage that they are sensitive to microphone mismatch[9] and microphone positions, and therefore they need to be carefully calibrated before operation, although techniques exist to make them more robust to such non-idealities [42–47].

Blind beamforming and blind source separation techniques also exist, which do not assume prior knowledge of the microphone and source positions, and which are usually also robust to microphone mismatch [42, 48–53]. They are therefore well-suited for noise reduction in ad hoc deployed WASNs. In the remaining of this section, we will focus on two blind multi-channel noise reduction techniques: the multi-channel Wiener filter and the blind linearly constrained minimum variance (LCMV) beamformer. These techniques form the backbone of the DANSE and linearly constrained DANSE algorithms, which will be introduced in chapters 2 and 6, respectively.

1.2.2 Multi-channel Wiener Filtering (MWF)

The multi-channel Wiener filter (MWF) is a successful blind noise reduction technique that estimates a desired speech signal in an arbitrarily chosen reference microphone [42, 48]. Consider a scenario as in Fig. 1.8, where a person produces a speech signal $s(\omega)$, with ω denoting the frequency-domain variable[10]. This signal is recorded by a microphone array with M microphones. Due to reflections on the walls and the objects in the room, the signal $x_m(\omega)$ that is observed at microphone m is a distorted version of the dry source signal $s(\omega)$, i.e., $s(\omega)$ is filtered by the room impulse response (RIR). Furthermore, microphone m also observes an additive noise component $v_m(\omega)$. The actual recorded signal $y_m(\omega)$ at microphone m can therefore be decomposed in

$$y_m(\omega) = x_m(\omega) + v_m(\omega), \;\; m = 1, ..., M \tag{1.1}$$

[9] Different microphones usually have different gains when recording sound.

[10] For the sake of an easy exposition, we will describe the MWF estimation theory in the frequency-domain. This allows us to describe all microphone signals as instantaneous mixtures of source signals, instead of time-domain convolutive mixtures. Both domains are theoretically equivalent, but they may give different results in practical applications due to the use of finite DFT-sizes.

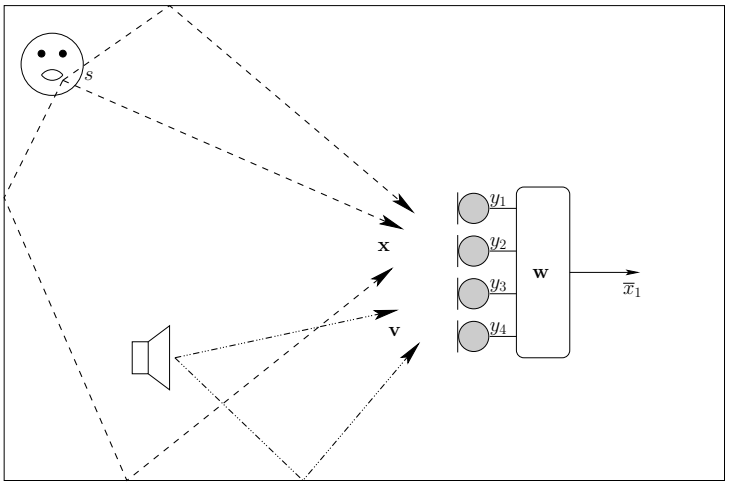


Figure 1.8: A typical scenario for multi-channel noise reduction.

where $x_m(\omega)$ is the desired speech component and $v_m(\omega)$ the undesired noise component. It should be noted that $x_m(\omega)$ can be a superposition of multiple desired speech signals, e.g., when recording a conversation. Furthermore, although $x_m(\omega)$ is referred to as the desired speech component, $v_m(\omega)$ is not necessarily non-speech, i.e., undesired speech sources may be included in $v_m(\omega)$. All microphone signals $y_m(\omega)$ are stacked in an M-dimensional column vector $\mathbf{y}(\omega) = [y_1(\omega)\, y_2(\omega)\, \ldots\, y_M(\omega)]^T$, and the vectors $\mathbf{x}(\omega)$ and $\mathbf{v}(\omega)$ are similarly constructed. The data model for the full microphone array can then be written as $\mathbf{y}(\omega) = \mathbf{x}(\omega) + \mathbf{v}(\omega)$.

The goal is to estimate the desired speech component $x_m(\omega)$ as it is observed in the m-th microphone, selected to be the reference microphone. Without loss of generality (w.l.o.g.), it is assumed that the reference microphone corresponds to $m = 1$. We filter each microphone signal with a particular filter, and then sum the M filter outputs to generate an estimate of $x_1(\omega)$. Let the M-dimensional column vector $\mathbf{w}(\omega)$ denote the stacked version of the M filter coefficients at frequency ω, then the estimate is generated with the filter-and-sum operation

$$\overline{x}_1(\omega) = \mathbf{w}(\omega)^H \mathbf{y}(\omega) \tag{1.2}$$

where the superscript H denotes the conjugate transpose operator. The filter coefficients of $\mathbf{w}(\omega)$ are chosen based on a minimum mean squared error (MMSE) criterion, i.e., by minimizing the following MSE cost function

$$J(\mathbf{w}(\omega)) = E\left\{|x_1(\omega) - \mathbf{w}(\omega)^H \mathbf{y}(\omega)|^2\right\} \tag{1.3}$$

where $E\{.\}$ denotes the expected value operator. It should be noted that such an optimization problem needs to be solved for each frequency ω. For

conciseness, the frequency-domain variable ω will be omitted in the sequel. By setting the gradient $\nabla J(\mathbf{w})$ to zero, the minimum of (1.3) is found to be:

$$\hat{\mathbf{w}} = \mathbf{R}_{yy}^{-1}\mathbf{R}_{yx}\mathbf{e}_1 \tag{1.4}$$

with $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$, $\mathbf{R}_{yx} = E\{\mathbf{y}\mathbf{x}^H\}$ and $\mathbf{e}_1$ an M-dimensional vector with the first entry set to 1 and all other entries set to 0, which selects the column of $\mathbf{R}_{yx}$ corresponding to the reference microphone. The filter (1.4) is referred to as the multi-channel Wiener filter.

The microphone signal correlation matrix $\mathbf{R}_{yy}$ can be estimated based on temporal averaging. Long term averaging over long data windows will result in an MWF that mainly exploits spatial characteristics and long-term noise spectra, since rapidly varying temporal information cannot be captured in this way. Since the desired signal $\mathbf{x}$ is unknown, we cannot directly compute $\mathbf{R}_{yx}$. However, if the desired speech sources in $\mathbf{x}$ are uncorrelated to the noise in $\mathbf{v}$, as it is usually the case, then $\mathbf{R}_{yx} = \mathbf{R}_{xx} = E\{\mathbf{x}\mathbf{x}^H\}$. The matrix $\mathbf{R}_{xx}$ is unknown, but by again relying on independence between $\mathbf{x}$ and $\mathbf{v}$, it can be computed as

$$\mathbf{R}_{xx} = \mathbf{R}_{yy} - \mathbf{R}_{vv} \tag{1.5}$$

with $\mathbf{R}_{vv} = E\{\mathbf{v}\mathbf{v}^H\}$. The noise correlation matrix $\mathbf{R}_{vv}$ can be (re-)estimated during noise-only periods and $\mathbf{R}_{yy}$ can be (re-)estimated during speech-and-noise periods, requiring a voice activity detection (VAD) mechanism (see, e.g., [54–56]). Even when the noise sources and the speech source are not stationary, the MWF (1.4) is observed to yield good noise reduction performance for long-term estimates of $\mathbf{R}_{yy}$ and $\mathbf{R}_{xx}$ [40, 48, 57], mainly due to the exploitation of spatial information, which usually changes relatively slowly over time.

The MWF can be extended to include a trade-off between speech distortion and noise reduction, referred to as the speech-distortion-weighted MWF (SDW-MWF) [58]. To this end, we rewrite the MSE cost function (1.3) as

$$J(\mathbf{w}) = E\left\{|x_1 - \mathbf{w}^H\mathbf{x}|^2\right\} + E\left\{|\mathbf{w}^H\mathbf{v}|^2\right\} \tag{1.6}$$

where we used the additive-noise model $\mathbf{y} = \mathbf{x} + \mathbf{v}$. It is observed that the first term of (1.6) denotes the speech distortion due to the filter $\mathbf{w}$, whereas the second term denotes the actual noise reduction. To derive the SDW-MWF, we add a trade-off parameter μ to the second term, to put more or less emphasis on the noise reduction:

$$J_{\text{SDW-MWF}}(\mathbf{w}) = E\left\{|x_1 - \mathbf{w}^H\mathbf{x}|^2\right\} + \mu E\left\{|\mathbf{w}^H\mathbf{v}|^2\right\} \ . \tag{1.7}$$

The SDW-MWF filters are then computed as

$$\hat{\mathbf{w}} = \left(\mathbf{R}_{xx} + \mu\mathbf{R}_{vv}\right)^{-1}\mathbf{R}_{xx}\mathbf{e}_1 \tag{1.8}$$

where a large value of μ puts more weight on the noise reduction, but generally results in more speech distortion. When $\mu \to \infty$, all the weight is on the noise

reduction, and distortion is ignored, yielding the trivial filter $\mathbf{w} = \mathbf{0}$. The limit case where $\mu \to 0$, corresponds to a distortionless response, where the remaining degrees of freedom are used for the noise reduction.

For the case of a single desired speech source, the speech correlation matrix $\mathbf{R}_{xx}$ has rank 1, and then the SDW-MWF solution (1.8) can be rewritten as [59]

$$\hat{\mathbf{w}}_{\text{R1-MWF}} = \mathbf{R}_{vv}^{-1}\mathbf{R}_{xx}\mathbf{e}_1 \frac{1}{\mu + \text{Tr}\{\mathbf{R}_{vv}^{-1}\mathbf{R}_{xx}\}} \tag{1.9}$$

where Tr{.} denotes the trace operator, i.e., the sum of the diagonal elements of the matrix. This is referred to as the rank-1 SDW-MWF (R1-MWF), and it is shown in [59] that the implementation based on (1.9) is numerically more favorable than an implementation based on the general SDW-MWF formula (1.8). Based on this rank-1 assumption, it can also be shown that the case where $\mu = 0$ is equivalent to the minimum variance distortionless response (MVDR) beamformer (see Subsection 1.2.3).

It should be noted that (1.5) and (1.9) are theoretical results. The estimated $\mathbf{R}_{xx}$ will not have rank 1 in practice and it may even not be positive (semi-)definite. Although these properties are not required to be able to use (1.9), the noise reduction performance of the MWF-based algorithms may be affected if these properties are not enforced. Furthermore, (1.9) may become unstable or ill-conditioned if $\mathbf{R}_{xx}$ becomes indefinite.

1.2.3 LCMV Beamforming

Linearly constrained minimum variance (LCMV) beamforming is a well-known beamforming technique [19, 60, 61], which aims to reduce the output variance of a filter-and-sum operator, under certain linear constraints, e.g., to preserve the target source signals. In the context of speech enhancement, its main advantage over SDW-MWF is that it is able to reduce noise without distorting the target speech sources. Furthermore, it allows to add extra constraints to fully suppress certain spatially located noise sources.

Assume an acoustic scenario where there are K relevant spatial point sources (we consider a point source as relevant, if this source is incorporated in the constraints of the LCMV beamformer, as explained later). We can then assume that $\mathbf{y}$ is generated by the following linear data model (in the frequency domain)

$$\mathbf{y}(\omega) = \mathbf{A}(\omega)\mathbf{s}(\omega) + \mathbf{v}(\omega) \tag{1.10}$$

where $\mathbf{s}(\omega)$ is a stacked signal vector containing the K relevant source signals, $\mathbf{A}(\omega)$ is an $M \times K$ steering matrix that contains the transfer functions from the K sources to the M microphones (modelling the room acoustics), and $\mathbf{v}(\omega)$ is a noise component. Sources that are not used in the linear constraints (see below)

are incorporated in $\mathbf{v}(\omega)$, even when they are spatially located and therefore in principle could be described by the first term. For conciseness, we will again omit the frequency variable ω in the sequel.

Let us first assume that the matrix $\mathbf{A}$ is known (this requires a fixed deterministic scenario, and it involves a prior training and calibration phase). The goal is to design a filter-and-sum beamformer to generate a signal $\overline{d} = \mathbf{w}^H\mathbf{y}$ with minimum variance, and hence removing as much noise as possible. However, to avoid the trivial solution $\mathbf{w} = \mathbf{0}$, several linear constraints are added. For example, if the goal is to obtain an undistorted estimate of s_k, i.e., the k-th source in $\mathbf{s}$, we need to make sure that $\mathbf{w}^H\mathbf{a}_k = 1$, where $\mathbf{a}_k$ denotes the k-th column of $\mathbf{A}$. If we also want to fully suppress the q-th source in $\mathbf{s}$, we also need that $\mathbf{w}^H\mathbf{a}_q = 0$. The general LCMV optimization problem is given by

$$\min_{\mathbf{w}} \; E\{\|\mathbf{w}^H\mathbf{y}\|^2\} \tag{1.11}$$

s.t.

$$\mathbf{A}^H\mathbf{w} = \mathbf{f} \tag{1.12}$$

where $\mathbf{f}$ contains the desired responses for each of the signals in $\mathbf{s}$. Usually, $f_k = 1$ if source k is desired, and $f_k = 0$ if source k is an interferer, where f_k denotes the k-th entry of $\mathbf{f}$. The solution of (1.11)-(1.12) is [19]:

$$\hat{\mathbf{w}} = \mathbf{R}_{yy}^{-1}\mathbf{A}\left(\mathbf{A}^H\mathbf{R}_{yy}^{-1}\mathbf{A}\right)^{-1}\mathbf{f}\,. \tag{1.13}$$

In many cases, a blind approach is preferred, where there is no prior knowledge required on the acoustic scenario, i.e., where the steering matrix $\mathbf{A}$ is unknown. This is especially important in adaptive beamforming applications where the source and microphone positions can change. Blind LCMV beamforming [51] then usually relies on a detection algorithm that detects if a desired source is active or not. Let $\mathcal{I}^d$ denote the set of indices that correspond to the N desired sources from $\mathbf{s}$ we want to preserve in the output of the LCMV beamformer. The other $P = K - N$ sources from $\mathbf{s}$ are assumed to be interferers, and their indices define the set $\mathcal{I}^n$. Let $\mathbf{Q}^d$ denote the $M \times N$ matrix with its columns defining a unitary basis for the desired subspace spanned by the columns of $\mathbf{A}$ with indices in $\mathcal{I}^d$. Similarly, let $\mathbf{Q}^n$ denote the $M \times P$ matrix containing a unitary basis for the interferer subspace corresponding to $\mathcal{I}^n$. Although it is usually difficult or impossible to estimate the individual columns of $\mathbf{A}$ in a blind fashion, the matrices $\mathbf{Q}^d$ and $\mathbf{Q}^n$ can often be estimated blindly from the sensor signals $\mathbf{y}$ (see e.g. [50]). In the sequel, we assume that these matrices can indeed be estimated blindly.

Since there is no knowledge on the true steering matrix, we aim to obtain a distortionless estimate of the mixture of the N desired signals from $\mathbf{s}$ as they impinge on one of the microphones, referred to as the reference microphone (assume w.l.o.g. that this is the first sensor, i.e. y_1). It is noted that we do not

necessarily intend to unmix the sources in $\mathcal{I}^d$, since this requires the estimation of each column of $\mathbf{A}$. In a multiple speaker scenario, to estimate the steering vectors of each speaker separately, the VAD must be able to distinguish between different speakers (e.g., with [62] or the technique proposed in Chapter 10). Furthermore, there should be sufficient segments with non-overlapping speech, for each speaker separately, to blindly estimate the subspace of each speaker. By estimating a mixture of desired sources, the problem is relaxed to estimating multi-dimensional subspaces, such that overlapping signal segments within $\mathcal{I}^d$ and $\mathcal{I}^n$ can also be used for the estimation of the desired and interference subspaces.

To this end, we compute the $\mathbf{w}$ that minimizes the variance of $\overline{d} = \mathbf{w}^H\mathbf{y}$, while preserving the desired signals in $\mathcal{I}^d$. If required, other constraints can be added, e.g. to (fully or partially) block the interferers in $\mathcal{I}^n$. More specifically, we solve the following centralized LCMV problem:

$$\min_{\mathbf{w}} \; E\{|\mathbf{w}^H\mathbf{y}|^2\} \tag{1.14}$$

s.t.

$$\mathbf{Q}^H\mathbf{w} = \mathbf{f} \tag{1.15}$$

with

$$\mathbf{Q} = \left[\begin{array}{cc} \mathbf{Q}^d & \mathbf{Q}^n \end{array} \right] \tag{1.16}$$

$$\mathbf{f} = \left[\begin{array}{c} \mathbf{q}_1^d \\ \epsilon\mathbf{q}_1^n \end{array} \right] \tag{1.17}$$

where $\mathbf{q}_1^d$ and $\mathbf{q}_1^n$ denote the first column of $\mathbf{Q}^{d\,H}$ and $\mathbf{Q}^{n\,H}$ respectively (corresponding to the reference microphone), and where ϵ is a user-defined gain[11]. Similarly to (1.13), the solution of this problem is given by:

$$\hat{\mathbf{w}} = \mathbf{R}_{yy}^{-1}\mathbf{Q}\left(\mathbf{Q}^H\mathbf{R}_{yy}^{-1}\mathbf{Q}\right)^{-1}\mathbf{f}\;. \tag{1.18}$$

It can be shown [50] that the signal components of $\mathbf{s}$ in the output $\hat{d} = \hat{\mathbf{w}}^H\mathbf{y}$, are equal to the signals as they impinge on the reference microphone (except for a scaling by ϵ), i.e., we obtain

$$\hat{d} = \sum_{l\in\mathcal{I}^d} a_{1l}s_l + \epsilon\sum_{l\in\mathcal{I}^n} a_{1l}s_l + \hat{\mathbf{w}}^H\mathbf{v} \tag{1.19}$$

with a_{kl} denoting the entry in the k-th row and l-th column of $\mathbf{A}$. It should be noted that this procedure yields a distortionless response, which is not the case in SDW-MWF based beamforming techniques (see Subsection 1.2.2). However,

[11] Usually $\epsilon = 0$ to fully cancel the interferers. However in some cases it may be important to retain some undistorted residual noise, e.g. for hearing aid users to be able to mentally reconstruct the acoustic environment.

the constraints that enforce this distortionless response remove some degrees of freedom, yielding less noise reduction in the residual $\hat{\mathbf{w}}^H\mathbf{v}$.

It should be noted that the minimum variance distortionless response (MVDR) beamformer [60, 61, 63] is a special case of LCMV beamforming, where $\mathcal{I}^n$ is empty and where $\mathcal{I}^d$ is a singleton, i.e., there is a single desired source which is the only relevant signal ($K = 1$). In this case, the steering matrix $\mathbf{A}$ becomes a steering vector $\mathbf{a}$, and the multi-channel signal $\mathbf{s}$ becomes a single-channel signal s. Let $\mathbf{x} = \mathbf{a}s$ denote the desired speech component in each microphone of the array, and let P_s denote the signal power of s, i.e., $P_s = E\{|s|^2\}$. The desired speech correlation matrix $\mathbf{R}_{xx} = E\{\mathbf{x}\mathbf{x}^H\}$ can then be written as

$$\mathbf{R}_{xx} = P_s\mathbf{a}\mathbf{a}^H \, . \tag{1.20}$$

Using (1.5) and (1.20), we can rewrite (1.18) as

$$\hat{\mathbf{w}} = \left(\mathbf{R}_{vv} + P_s\|\mathbf{a}\|^2\mathbf{q}\mathbf{q}^H\right)^{-1}\mathbf{q}\,\frac{1}{\mathbf{q}^H\left(\mathbf{R}_{vv} + P_s\|\mathbf{a}\|^2\mathbf{q}\mathbf{q}^H\right)^{-1}\mathbf{q}}\,\frac{a_1^*}{\|\mathbf{a}\|} \tag{1.21}$$

where $\mathbf{q} = \frac{1}{\|\mathbf{a}\|}\mathbf{a}$, and where we set $\mathbf{f} = \frac{a_1^*}{\|\mathbf{a}\|}$ with a_1 denoting the first entry of $\mathbf{a}$ (this corresponds to the choice of $\mathbf{f}$ in (1.17)). By applying the matrix inversion lemma [64], and using the notation $\rho = \mathbf{q}^H\mathbf{R}_{vv}^{-1}\mathbf{q}$, we can rewrite (1.21) as

$$\begin{aligned}\hat{\mathbf{w}} &= \mathbf{R}_{vv}^{-1}\mathbf{q}\frac{1 - \left(\frac{1}{P_s\|\mathbf{a}\|^2} + \rho\right)^{-1}\rho}{\rho - \left(\frac{1}{P_s\|\mathbf{a}\|^2} + \rho\right)^{-1}\rho^2}\,\frac{a_1^*}{\|\mathbf{a}\|} \\ &= \mathbf{R}_{vv}^{-1}\mathbf{q}\frac{1}{\rho}\frac{a_1^*}{\|\mathbf{a}\|} \, .\end{aligned} \tag{1.22}$$

Since $\rho = \mathbf{q}^H\mathbf{R}_{vv}^{-1}\mathbf{q} = \mathrm{Tr}\{\mathbf{q}^H\mathbf{R}_{vv}^{-1}\mathbf{q}\} = \mathrm{Tr}\{\mathbf{R}_{vv}^{-1}\mathbf{q}\mathbf{q}^H\} = \frac{1}{P_s\|\mathbf{a}\|^2}\mathrm{Tr}\{\mathbf{R}_{vv}^{-1}\mathbf{R}_{xx}\}$, we eventually obtain

$$\hat{\mathbf{w}} = \mathbf{R}_{vv}^{-1}\mathbf{R}_{xx}\mathbf{e}_1\frac{1}{\mathrm{Tr}\{\mathbf{R}_{vv}^{-1}\mathbf{R}_{xx}\}} \, . \tag{1.23}$$

As mentioned earlier, this is exactly the same expression as the rank-1 SDW-MWF in (1.9) with $\mu = 0$. Therefore, MVDR beamforming can be considered as a special case of SDW-MWF.

1.3 Techniques for Distributed Signal Estimation in WSNs

In this section, we give an overview of some state-of-the-art linear signal estimation techniques for wireless sensor networks. As mentioned in Section 1.1.3,

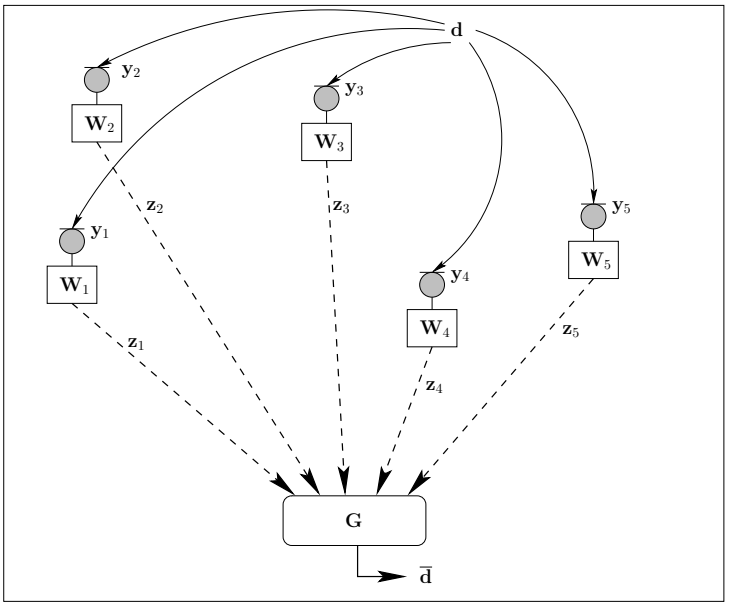


Figure 1.9: A typical compress-and-fuse scenario.

signal estimation usually cannot rely on iterative techniques since the number of variables that are estimated increases linearly with the number of sensor observations. Due to the large amount of literature on linear estimation in WSNs, we restrict this overview to a particular selection of algorithms and estimation problems, including (w.l.o.g.) some acoustically-oriented distributed signal estimation algorithms. However, the latter can also be applied to more general (non-acoustical) signal estimation problems. Although not truly being a signal estimation problem per se, we also incorporate distributed source coding in this section (see Subsection 1.3.4), since it is closely related and highly relevant in signal estimation problems.

1.3.1 Compress and Fuse

General Problem Statement

A lot of work on signal estimation in WSNs adopts a centralized architecture with one-way data transmission from the sensor nodes to a fusion center (FC) [7–12]. The idea is then to let each node compress its sensor observations locally, and transmit the compressed observations to the FC who computes the final signal estimate, as depicted in Fig. 1.9. We will denote this as 'compress and fuse'.

Consider a WSN with a set of J nodes $\mathcal{J} = \{1, \ldots, J\}$. At time t, node k

observes an M-dimensional data vector $\mathbf{y}_k[t]$, which can be a set of time samples of a single sensor signal, or a stacked version of samples of multiple sensor signals available at node k. It is assumed that these samples are generated by stationary stochastic processes. Node k compresses this sample vector with a $K \times M$ compression matrix $\mathbf{W}_k$ to obtain a K-dimensional[12] vector $\mathbf{z}_k[t]$ with reduced dimension ($K < M$), i.e., $\mathbf{z}_k[t] = \mathbf{W}_k\mathbf{y}_k[t]$. The FC then collects the stacked data vector $\mathbf{z}[t] = [\mathbf{z}_1[t]^T \ \ldots \ \mathbf{z}_J[t]^T]^T$ and computes the final estimate $\overline{\mathbf{d}}[t] = \mathbf{G}\mathbf{z}[t]$, where $\mathbf{G}$ is a $Q \times (KJ)$ matrix[13]. It is noted that for each observation time t, a new vector $\overline{\mathbf{d}}[t]$ is estimated, which should be as close as possible to the true desired vector $\mathbf{d}[t]$, according to some optimality criterion, e.g., best linear unbiased estimation (BLUE), also known as linear minimum mean squared error (LMMSE). The challenge is then to design the local compression rules at the nodes (the $\mathbf{W}_k$'s), and the fusion rule (the matrix $\mathbf{G}$) at the FC. It is noted that, if all the $\mathbf{W}_k$'s are equal to the $M \times M$ identity matrix $\mathbf{I}_M$, then there is no compression, which is referred to as centralized fusion.

One way to tackle this problem, is the so-called standard estimation fusion [7, 65]. In this case, a node k computes a local estimate $\overline{\mathbf{d}}_k[t]$, according to the same optimality criterion as used in the FC. This local estimate is then transmitted to the FC which combines the local estimates of the different nodes to a single final estimate $\overline{\mathbf{d}}[t]$. This approach is of course only useful if $Q < M$, otherwise there is no compression in the transmitted data. In this way, the design of all the $\mathbf{W}_k$'s and $\mathbf{G}$ is decoupled into $J + 1$ separate estimation problems, and therefore no joint statistics are required. The limitations of this approach are however obvious. It is almost surely suboptimal, since the design is partitioned in decoupled sub-problems. Furthermore, it does not incorporate explicit bandwidth constraints, as the dimension of the transmitted data vectors is always Q.

To obtain a better or optimal estimation fusion, all the $\mathbf{W}_k$'s and $\mathbf{G}$ should be designed jointly. However, except for some special cases, the design of such compress-and-fuse algorithms usually relies on prior knowledge of the signal statistics and cross-correlations between sensor signals and/or the target signal. Therefore, the compression and fusion rules are usually computed offline before operation of the estimation algorithm, and hence there is no adaptation. In the remaining of this subsection, we address some strategies for optimal and suboptimal compress-and-fuse estimation.

Best Linear Unbiased Estimator

[12] For the sake of an easy exposition, it is assumed here (w.l.o.g.) that the dimensions of all the $\mathbf{y}_k$'s are equal (M) and the dimensions of all the $\mathbf{z}_k$'s are equal (K). However, the dimension for all these vectors may also be node-specific.

[13] In many cases, $Q = M$, e.g., when $\mathbf{y}_k$ contains time samples of a single sensor signal, and the goal is to estimate a hidden signal component in this sensor signal.

For notational convenience, we omit the time index t in the sequel, and we assume (w.l.o.g.) that the true vector $\mathbf{d}$, and all sensor observations $\mathbf{y}_k$ are zero-mean. The goal is to obtain the BLUE or LMMSE estimator that minimizes

$$\arg\min_{\mathbf{G},\mathbf{W}} J(\mathbf{G},\mathbf{W}) = E\{\|\mathbf{d}-\mathbf{G}\mathbf{W}\mathbf{y}\|^2\} \tag{1.24}$$

s.t.

$$\mathbf{W} = \text{blockdiag}\{\mathbf{W}_1,\ldots,\mathbf{W}_J\} \tag{1.25}$$

where blockdiag{.} is a block-diagonal matrix with its arguments on the block diagonal, and where $\mathbf{y} = \left[\mathbf{y}_1^T \ \ldots \ \mathbf{y}_J^T\right]^T$.

In [7, 10], the matrix $\mathbf{G}$ is a priori chosen as the BLUE estimator[14] with respect to the collected compressed data vector $\mathbf{z}$, i.e.,

$$\mathbf{G}_{\text{BLUE}} = \mathbf{R}_{dz}\mathbf{R}_{zz}^{-1} \tag{1.26}$$

where $\mathbf{R}_{dz} = E\{\mathbf{d}\mathbf{z}^H\}$ and $\mathbf{R}_{zz} = E\{\mathbf{z}\mathbf{z}^H\}$. Note how this is similar to the MWF solution in a single frequency bin (1.4), which is also a BLUE estimator. This fusion rule minimizes the MSE between $\mathbf{d}$ and $\mathbf{G}\mathbf{z}$. The local compression rules at the nodes, i.e., the entries in $\mathbf{W}$, can be obtained by minimizing

$$\arg\min_{\mathbf{W}} \widetilde{J}(\mathbf{W}) = E\{\|\mathbf{d}-\mathbf{G}_{\text{BLUE}}\mathbf{W}\mathbf{y}\|^2\} \tag{1.27}$$

$$= \text{Tr}\left\{\mathbf{R}_{dd} - \mathbf{R}_{dy}\mathbf{W}\left(\mathbf{W}^H\mathbf{R}_{yy}\mathbf{W}\right)^{-1}\mathbf{W}^H\mathbf{R}_{dy}^H\right\} \tag{1.28}$$

s.t.

$$\mathbf{W} = \text{blockdiag}\{\mathbf{W}_1,\ldots,\mathbf{W}_J\} \tag{1.29}$$

where $\mathbf{R}_{dd} = E\{\mathbf{d}\mathbf{d}^H\}$, $\mathbf{R}_{dy} = E\{\mathbf{d}\mathbf{y}^H\}$ and $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$.

Special Cases

There is no general closed-form expression for the solution of the highly non-convex optimization problem (1.27)-(1.29). However, in [7], three special cases are considered for which a closed form expression can be found: the single-node case ($J = 1$), the case where the sensor signals are uncorrelated ($E\{\mathbf{y}_k\mathbf{y}_q^H\} = 0$ for any $k \neq q$), and the case with identically correlated sensor signals ($E\{\mathbf{y}_k\mathbf{y}_q^H\} = \mathbf{R}_1$, $\forall\, k,q \in \mathcal{J}$ and $E\{\mathbf{d}\mathbf{y}_k^H\} = \mathbf{R}_2$, $\forall\, k \in \mathcal{J}$). However, these special cases rarely match with practical scenarios.

The case with uncorrelated sensor signals ($E\{\mathbf{y}_k\mathbf{y}_q^H\} = 0$ for any $k \neq q$) is also addressed in [9], based on the general formulation (1.24)-(1.25). It is shown that

[14] In [9], it is shown that this a priori choice may yield suboptimal results, i.e., sometimes a lower MSE can be obtained at the same compression rate.

the optimal strategy is then to let node k compute a local LMMSE estimate $\overline{\mathbf{d}}_k$, based on the locally[15] available $\mathbf{y}_k$, and then compress this estimate by means of a local principal component analysis (PCA) implemented by a Karhunen-Loève transform (KLT) [66]. The matrix $\mathbf{G}$ is then chosen as the stacked decompression matrices of the local KLTs. This is referred to as 'estimate-compress' (EC), since first an estimate is computed, which is then compressed to reduce the communication bandwidth. It is also shown that the reversed approach (compress-estimate or CE), as in [10], is usually suboptimal.

The fact that EC outperforms CE is often observed in WSNs with limited or partial spatial correlation. The intuitive reason is that, in CE, the compression-scheme does not incorporate the fact that noise needs to be reduced, and therefore also preserves the information in the noise component. This means that the available bandwidth is allocated to both the noise and the signal component. In EC, the noise is first reduced by a prior estimation stage, and then the compression stage will focus more on the desired signal, and will spill less bandwidth on transmitting the noise. It is noted that this is only advantageous in the case where the noise is uncorrelated in the different nodes (see also Subsection 1.3.3, where a similar result is obtained in a speech enhancement context). If there is spatial correlation in the noise component, it may be beneficial to also include the noise component in the signals that are transmitted to the FC. In this case, EC may yield highly suboptimal results.

The General Case

A common method to solve (1.27)-(1.29) [7] or (1.24)-(1.25) [8] for the general case, is to use an offline Gauss-Seidel iteration algorithm, which is basically a block-coordinate descent method [7, 67]. In each iteration a single variable is optimized, while fixing the other variables to their current value, until all optimization variables have converged. However, this only provides a stationary point of the cost function, which is not necessarily the optimum. Simulation results in [7, 9] demonstrate that this procedure often works well, and usually an estimator is obtained that is close to the optimal solution with respect to the imposed constraints. It should be noted that this procedure yields fixed fusion rules, based on full knowledge of the sensor signal cross-correlations.

It is obvious that, in general

$$\widetilde{J}(\mathbf{W}) \geq \widetilde{J}(\mathbf{I}_{MJ}) \tag{1.30}$$

i.e., compression results in a larger MSE than in the case of centralized fusion. However, if[16] $K \geq P = \text{rank}\left(\mathbf{R}_{dy}^H \mathbf{R}_{dy} \mathbf{R}_{yy}^{-1}\right)$, the data can be compressed

[15] The fact that only locally available data is required for an optimal performance facilitates adaptive and distributed implementations.

[16] For example, this is satisfied if $K \geq Q$, i.e., the dimension of the compressed sensor data $\mathbf{z}_k$ is at least as large as the dimension of the final estimate $\overline{\mathbf{d}}$.

without information loss for estimation fusion [7, 8], i.e., we obtain the same performance as the centralized fusion. To obtain the optimal performance, the compression matrices need to be chosen as

$$\begin{bmatrix} \mathbf{W}_1 \\ \vdots \\ \mathbf{W}_J \end{bmatrix} = \mathbf{R}_{yy}^{-1}\mathbf{U} \tag{1.31}$$

where the columns of $\mathbf{U}$ contain the eigenvectors corresponding to the P non-zero eigenvalues of $\mathbf{R}_{dy}^H\mathbf{R}_{dy}\mathbf{R}_{yy}^{-1}$. It is noted that the data compression still relies on prior knowledge on the correlation between sensor signal observations. If a sensor compresses its observations by only considering its local information, generally there is information loss, unless extra assumptions are imposed [8].

The fact that the centralized BLUE estimator can be obtained if $K \geq Q$, will also be exploited in Chapter 2 to derive a distributed adaptive node-specific signal estimation (DANSE) algorithm. However, the latter does not require prior knowledge of $\mathbf{R}_{yy}$. Instead, it estimates and re-estimates all required statistical quantities on the *compressed* data during operation, which makes it fully adaptive, as opposed to the fixed fusion rules obtained with a centralized Gauss-Seidel approach. Furthermore, in DANSE, each node is allowed to estimate a node-specific signal, which allows for blind estimation based on local reference sensors.

Extension: Linear Sensor Data Model

Up to this point, we have not imposed any assumptions on the signal observations in $\mathbf{y}$ (except for stationarity). By doing so, some other interesting cases can be solved and analyzed. In [8], a linear sensor data model is adopted, i.e.,

$$\mathbf{y}_k = \mathbf{A}_k\mathbf{d} + \mathbf{v}_k \tag{1.32}$$

where $\mathbf{A}_k$ is a fixed $M \times Q$ matrix, and $\mathbf{v}_k$ is additive zero-mean white noise that is independent of $\mathbf{d}$. Furthermore, it is assumed that the noise is white ($E\{\mathbf{v}_k\mathbf{v}_k^H\} = \mathbf{I}_M$) and spatially uncorrelated ($E\{\mathbf{v}_k\mathbf{v}_q^H\} = 0$ if $k \neq q$).

It is assumed that the $\mathbf{A}_k$'s are known, which removes the need for $\mathbf{R}_{dy}$, i.e., the signal statistics of $\mathbf{d}$ do not need to be known. If $K \geq Q$, the resulting MMSE problem based on the general formulation (1.24)-(1.25) has a closed-form solution that, in contrast to (1.31), does not depend on the cross-correlation between sensor data [8]. Only knowledge on the local mixing matrix $\mathbf{A}_k$ is required to compute $\mathbf{W}_k$. If this matrix can be estimated and communicated to the FC during operation of the algorithm, this allows for an adaptive compress-and-fuse algorithm.

The DANSE algorithm introduced in Chapters 2 to 5 is able to obtain the centralized LMMSE solution in an adaptive iterative fashion, without knowl-

edge of the $\mathbf{A}_k$'s (and actually without imposing a sensor data model such as (1.32)). However, this is only possible if there is 2-way communication such that information can also flow to the sensor nodes, i.e., the sensor nodes must know what is going on in the rest of the network.

1.3.2 Distributed Signal Estimation in Ad hoc Sensor Networks

In the previous subsection, it is explained how signal estimation is tackled in a network with a star topology, where the center node is a fusion center. Signal estimation in networks with an ad hoc topology is very different, and is usually tackled in an ad hoc way, by a priori choosing a local fusion rule for the sensor nodes, and then optimizing the parameters to obtain minimum variance over all the estimates of all the nodes. An extra challenge is to guarantee that the estimated signals remain stable in each node, which is non-trivial due to feedback paths through the network.

As an example, we briefly address the estimator proposed in [18]. It is an adaptive algorithm for signal estimation in networks with an ad hoc topology, based on the following observation model at node k:

$$y_k[t] = d[t] + v_k[t] \tag{1.33}$$

where d is the desired signal that needs to be estimated, v_k is a zero-mean white noise signal that is uncorrelated with d, and y_k is the signal as observed by node k. Note that there is no mixture or steering matrix as in (1.32), i.e., each node observes an undistorted version of the desired signal $d[t]$ (when ignoring the uncorrelated noise component). The noise is also assumed to be spatially uncorrelated, i.e., $E\{v_k v_q\} = 0$ if $k \neq q$. This restrictive data model and the aforementioned assumptions on the noise are important to perform a stability analysis of the algorithm, and to define a cost function that can be solved locally by each node.

Using this data model, the following local estimator is a priori assumed at node k:

$$\overline{d}_k[t] = \sum_{q \in \mathcal{N}_k \cup \{k\}} g_{kq}[t] \overline{d}_q[t-1] + \sum_{q \in \mathcal{N}_k \cup \{k\}} w_{kq}[t] y_q[t] \tag{1.34}$$

where $\mathcal{N}_k$ denotes the set of nodes that are directly connected to node k (node k excluded). This estimator makes a linear combination of the new sensor observations and the estimates of the previous sample from all the nodes in the neighborhood. The weights $\{g_{kq}\}$ and $\{w_{kq}\}$ are time-dependent to obtain an adaptive algorithm. It is noted that this is a heuristic estimator that is very different from traditional centralized BLUE estimators, and therefore highly suboptimal. It only uses instantaneous measurements from a 1-hop neighborhood of node k. Observations that originated beyond this neighborhood can

only be incorporated in the estimate of future samples. Therefore, it is expected that the estimator only exploits data of the full network if the target signal d changes slowly over time, i.e. if $d[t-L] \approx d[t]$, where L denotes the maximum number of hops between any pair of nodes. This implies that the signal d should be heavily oversampled when the algorithm is applied in large networks.

The minimum variance unbiased estimator (MVUE) for (1.34) is obtained if $g_{kq}[t] = 0$ for any k, q and t, and if

$$w_{kq}[t] = w_{qk}[t] = \begin{cases} \frac{1}{|\mathcal{N}_k|+1} & \text{if } q \in \mathcal{N}_k \cup \{k\} \\ 0 & \text{otherwise} \end{cases} \tag{1.35}$$

where $|\mathcal{S}|$ denotes the cardinality of the set $\mathcal{S}$. This means that an unbiased estimate can only be obtained in a partially isolated case where each node only uses raw measurements from nodes in a one-hop neighborhood. To further reduce the error variance, biased estimates need to be considered. To this end, let $\mathbf{G}[t]$ and $\mathbf{W}[t]$ denote the weighting matrices, i.e., the entry at the k-th row and the q-th column of $\mathbf{G}[t]$ is $g_{kq}[t]$, where $g_{kq}[t] = 0$ if node k and q are not connected, and similarly for $w_{kq}[t]$ in $\mathbf{W}[t]$. It can be shown that the bias of the estimator (1.34) is bounded if $\forall\, t \in \mathbb{N}, \exists\, \delta > 0 :\ |d[t] - d[t-1]| < \delta$, and if the following two conditions are satisfied:

$$(\mathbf{G}[t] + \mathbf{W}[t])\, \mathbf{1} = \mathbf{1} \tag{1.36}$$

where $\mathbf{1}$ is a vector with each entry set to unity. Furthermore assume that there exists a $0 \leq \sigma_0 < 1$ such that

$$\sigma_{\max}\left(\mathbf{G}[t]\right) \leq \sigma_0 \tag{1.37}$$

where $\sigma_{\max}\left(\mathbf{G}[t]\right)$ denotes the largest singular value of $\mathbf{G}[t]$. The upperbound on the bias is then given by

$$\lim_{t\to\infty} |E\{d - \overline{d}_k\}| \leq \frac{\sqrt{J}\delta\sigma_0}{1-\sigma_0} \tag{1.38}$$

where J denotes the number of nodes in the network. This means that the bias tends to increase if the number of nodes J increases, and if the desired signal d changes more rapidly (δ increases). However, the increase of these variables usually yields a decrease in the error variance $E\{|e_k - E\{e_k\}|^2\}$, where $e_k = d - \overline{d}_k$.

In [18], it is explained how the weights of node k, i.e., $\{g_{kq}\}$ and $\{w_{kq}\}$, $\forall\, q \in \mathcal{N}_k \cup \{k\}$, can be computed adaptively in a fully distributed fashion, such that the *local* variance of the estimator (1.34) is minimized. It is noted that this is based on *local* minimization problems, and hence this is not a minimization of the overall variance $\sum_{k=1}^{J} E\{|e_k - E\{e_k\}|^2\}$, since the weights of different nodes influence each other.

The advantage of signal estimation techniques as in the above example, is that they have many of the properties that are desired in large-scale WSNs, i.e. scalability, ad hoc topologies, adaptation, uniformity, and self-healing properties. However, this comes at the price of obtaining a signal estimate that is usually biased and highly suboptimal, and there are no steady-state convergence results. Furthermore, the estimation algorithm relies on restrictive sensor data models such as (1.33), and the spatial correlation can only be fully exploited if the sensor signals are heavily oversampled, i.e., if the signal varies slowly in comparison to the sampling clock. Although this is strictly spoken a signal estimation algorithm, the latter assumption actually implies that the method is more or less based on iterative refinement of previous estimates, and therefore akin to a parameter estimation framework where a slowly varying scalar parameter is tracked.

From the example above, it should be clear that signal estimation in simply connected networks (without fusion center) is a difficult problem, and one often has to settle for suboptimal heuristic techniques to solve it. In Chapter 5 of this thesis, we tackle this problem in a more fundamental way, i.e., we aim to compute the optimal centralized LMMSE or BLUE estimate in an adaptive distributed fashion, and without imposing any assumptions on the sensor data model. It is shown that it is only possible to achieve this optimal performance if there are no feedback paths in the signal flow graph. Therefore, the network links need to be pruned to a tree topology to avoid cycles or loops in the network graph. For such networks, the tree-DANSE (T-DANSE) algorithm is then defined, based on a similar (but slightly different) parametrization as in (1.34), and which can be shown to converge to the optimal centralized LMMSE estimator.

1.3.3 Distributed Noise Reduction in Binaural Hearing Aids

In this subsection, we address a specific distributed signal estimation problem, i.e., MWF-based acoustic noise reduction in hearing aids (HAs). In particular, we consider the case of binaural hearing aids (BHAs), i.e., a pair of wirelessly-connected hearing aids (one at both ears). The hearing aids can then exchange audio signals to improve their local noise reduction performance. This is essentially a two-node WASN, where both nodes have multiple microphones (the current-generation HAs usually have 2 or 3 microphones to allow for multi-microphone noise reduction). Assuming that there are bandwidth constraints on the wireless link that do not allow transmission of all the microphone signals, an important question is then how the microphone signals can be optimally combined and compressed to satisfy these constraints.

Decoupled Distributed Noise Reduction

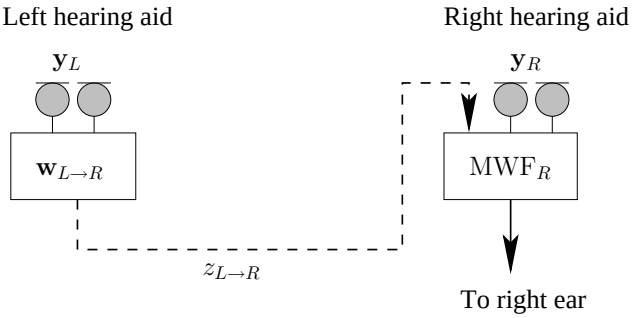


Figure 1.10: The decoupled distributed noise reduction problem in binaural hearing aids.

One way to tackle this problem is to decouple the noise reduction problems in the left HA and the right HA, i.e., ignoring the fact that there is two-way communication between the HAs. This is schematically depicted in Fig. 1.10: the (frequency-domain) microphone signals of the left HA, stacked in $\mathbf{y}_L$, are linearly combined with a compression filter $\mathbf{w}_{L\rightarrow R}$ to obtain a single-channel audio signal $z_{L\rightarrow R}$ that is transmitted to the right HA, which combines it with its own microphone signals. Notice that this is basically a compress-and-fuse problem, where the left HA serves as a sensor node, and where the right HA is the fusion center[17]. We assume that the filtering at the right HA is based on the MWF (see Chapter 1.2.2) with the signals $\mathbf{y}_R$ and $z_{L\rightarrow R}$ as its inputs, and with one of the channels of $\mathbf{y}_R$ selected to be the reference microphone[18]. This decoupled MWF-based noise reduction problem has been investigated in [40, 68] for different choices of the compression filter $\mathbf{w}_{L\rightarrow R}$:

- ***Fixed superdirective beamformer*** [40]: Choose $\mathbf{w}_{L\rightarrow R}$ as a fixed superdirective beamformer steered towards the front. The obvious disadvantage of this approach is that it relies on the assumption that the target source is exactly in front of the HA user, which is not always the case.
- ***Desired speech estimate*** [40, 68]: Choose $\mathbf{w}_{L\rightarrow R}$ as the MWF that estimates the desired speech source based on the microphone signals of the left HA. In [40], this was referred to as MWF-Contra.
- ***Interferer estimate*** [68]: Instead of transmitting an estimate of the target signal, an alternative is to transmit a noise reference to the right HA. In this case, $\mathbf{w}_{L\rightarrow R}$ creates an estimate of the interferer (in LMMSE sense).
- ***Raw microphone signal*** [40, 68]: Choose $\mathbf{w}_{L\rightarrow R} = \mathbf{e}_m$, where $\mathbf{e}_m$ is

[17]The microphone signals from the right HA can be treated as uncompressed sensor observations that the fusion center receives from another virtual sensor node.

[18]The goal for the right HA is to estimate the desired speech signal as it impinges on one of its own microphones. This means that each HA estimates a different version of the same signal. This is important to preserve auditory cues for spatial hearing [57].

an all-zero vector, except for the m-th entry, which is equal to one. This corresponds to the case where the m-th microphone signal is selected and transmitted to the right HA.

The above choices have been investigated in different acoustic scenarios, and their performance is observed to heavily depend on the scenario in which they are used. For the cases that are investigated in [68], a rate-distortion analysis is also provided, where rate constraints in the wireless link are explicitly incorporated (see also Subsection 1.3.4). This means that, before transmission, the signal $z_{L\rightarrow R}$ is optimally compressed (in an information-theoretic sense) to match a given bit rate.

In general, the fixed superdirective beamformer is observed to give the worst performance, unless for some particular scenarios with properties that more or less match with the design requirements of the beamformer. The MWF-contra has a good overall performance, i.e., in most scenarios it performs better than transmitting a raw microphone signal or a fixed beamformer output. It is also shown in [40] that this distributed estimator is LMMSE-optimal in a scenario where the noise at the two hearing aids is uncorrelated. This is similar to the optimality of the estimate-compress (EC) strategy in the case of uncorrelated sensor observations in compress-and-fuse techniques, as addressed in Subsection 1.3.1.

The analysis in [68] demonstrates that, in scenarios with a significantly loud interfering source, transmitting a raw microphone signal performs better than transmitting a desired signal estimate or a noise reference. An intuitive reason could be that both the desired signal estimate and the interferer estimate remove important information (i.e., the cancelled source) that can be useful for the receiving HA. In absence of a significant interferer, transmitting a desired signal estimate only gives a marginal improvement in SNR compared to transmitting the raw signal. Therefore, it is suggested in [68] to always transmit a raw microphone signal, since it yields the best performance in the most relevant scenarios, and it requires much less computational power at the transmitting node. However, it is important to remark that the results obtained in [68] only hold for scenarios with a single interfering source, and therefore do not contradict the results in [40], where MWF-contra was observed to be the better strategy.

Optimal Distributed Noise Reduction

Obviously, the above addressed ad hoc techniques are -in most scenarios- suboptimal compared to the case where both HAs have access to all the microphone signals of both ears (from now on referred to as the centralized MWF). However, under the assumption of a single desired speech source, it is possible to obtain the optimal centralized MWF solution, without increasing the commu-

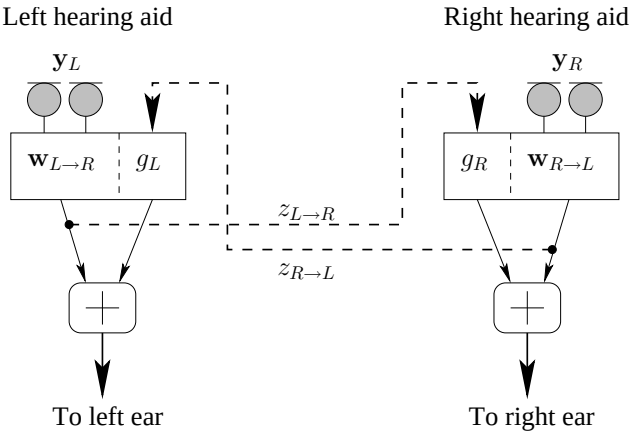


Figure 1.11: The distributed multi-channel Wiener filter for noise reduction in binaural hearing aids.

nication bandwidth, if we couple the noise reduction problems of both HAs. In [40], an iterative distributed noise reduction algorithm has been proposed, referred to as distributed MWF (DB-MWF), which indeed converges to the optimal centralized MWF as if both hearing aids have access to all microphone signals.

A schematic illustration of the DB-MWF algorithm is depicted in Fig. 1.11. The algorithm is similar to the MWF-contra procedure, but instead of choosing $\mathbf{w}_{L\to R}$ as the MWF with $\mathbf{y}_L$ as its input signals, we compute the MWF with $\mathbf{y}_L$ and $z_{R\to L}$ as input signals, i.e., we exploit the availability of a filtered mixture of the microphone signals of the right HA. The compression filter $\mathbf{w}_{L\to R}$ is then defined by the part of this local MWF that is applied to the microphone signals of the left HA itself. The part of the local MWF that is applied to the signal $z_{R\to L}$ is denoted by g_L. The dual problem is then solved at the right HA. In particular, the DB-MWF algorithm consists of the following steps:

The DB-MWF Algorithm [40]

1. Initialize $\mathbf{w}_{L\rightarrow R}$ with a random non-zero filter $\mathbf{w}_{L\rightarrow R}^{0}$.
2. Initialize $i \leftarrow 0$.
3. At the left HA, compute $z_{L\rightarrow R}^{i} = \mathbf{w}_{L\rightarrow R}^{i\,H}\mathbf{y}_L$, and transmit this signal to the right HA.
4. At the right HA, compute the MWF with inputs $\mathbf{y}_R$ and $z_{L\rightarrow R}^{i}$, i.e.,
$$\begin{bmatrix} \mathbf{w}_{R\rightarrow L}^{i+1} \\ g_R^{i+1} \end{bmatrix} = \arg\min_{\mathbf{w}} E\left\{ \left| x_R - \mathbf{w}^H \begin{bmatrix} \mathbf{y}_R \\ z_{L\rightarrow R}^{i} \end{bmatrix} \right|^2 \right\} \tag{1.39}$$
where x_R is the speech component in the reference microphone of the right HA.
5. At the right HA, compute $z_{R\rightarrow L}^{i+1} = \mathbf{w}_{R\rightarrow L}^{i+1\,H}\mathbf{y}_R$, and transmit this signal to the left HA.
6. At the left HA, compute the MWF with inputs $\mathbf{y}_L$ and $z_{R\rightarrow L}^{i+1}$, i.e.,
$$\begin{bmatrix} \mathbf{w}_{L\rightarrow R}^{i+1} \\ g_L^{i+1} \end{bmatrix} = \arg\min_{\mathbf{w}} E\left\{ \left| x_L - \mathbf{w}^H \begin{bmatrix} \mathbf{y}_L \\ z_{R\rightarrow L}^{i+1} \end{bmatrix} \right|^2 \right\} \tag{1.40}$$
where x_L is the speech component in the reference microphone of the left HA.
7. $i \leftarrow i+1$
8. Return to step 3.

In each iteration, each HA minimizes a local MWF cost function, similar to (1.3), for a local reference microphone. In [40] it is shown that this procedure converges in the case of a single desired speech source. Furthermore, it turns out that the resulting filters obtain the same output signals as the centralized MWF that has access to all the microphones of both hearing aids, i.e.,

$$\begin{bmatrix} \mathbf{w}_{R\rightarrow L}^{\infty} \\ \mathbf{w}_{L\rightarrow R}^{\infty}\, g_R^{\infty} \end{bmatrix} = \mathbf{w}_R \tag{1.41}$$

where $\mathbf{w}_R$ is the centralized MWF solution at the right HA, i.e.,

$$\mathbf{w}_R = \arg\min_{\mathbf{w}} E\left\{ \left| x_R - \mathbf{w}^H \begin{bmatrix} \mathbf{y}_R \\ \mathbf{y}_L \end{bmatrix} \right|^2 \right\} . \tag{1.42}$$

The MWF cost functions in the DB-MWF algorithm can also be replaced by SDW-MWF cost functions (1.6), without harming the convergence and optimality results. In [69], a similar distributed algorithm is considered for

distributed MVDR beamforming in binaural HAs, for which convergence and optimality is proven, again for the case of a single speech source. DB-MWF can be easily transformed to this distributed MVDR by using the R1-MWF formula (1.9) to solve (1.39) and (1.40), where the trade-off parameter is set to $\mu = 0$ (see also Subsection 1.2.3).

The DB-MWF algorithm suggests that a block of data is iteratively refined, i.e., re-estimated and retransmitted multiple times. However, this may require a bandwidth and computational power[19] that is larger than transmitting the raw microphone signals, i.e., directly computing the centralized MWF. Furthermore, for short data blocks (e.g. for real-time processing), it is not possible to compute this iterative refinement, since both the local speech correlation matrix and speech-plus-noise correlation matrix (represented by $\mathbf{R}_{yy}$ and $\mathbf{R}_{xx}$ in Subsection 1.2.2) need to be re-estimated in each iteration. This is only possible if there is both a noise-segment and a speech-plus-noise segment in the considered block of data. However, the different iterations of the DB-MWF algorithm can be spread out over different data blocks in a time-recursive implementation, such that each block of data is transmitted and estimated only once. Since the spectrum of speech signals changes rapidly in time, these rapid variations will probably not be captured by the estimation process i.e., mostly spatial information will be exploited.

The DANSE algorithm that is introduced in Chapter 2, and further extended and modified in Chapters 3 to 6, is based on the DB-MWF algorithm for BHAs. DANSE generalizes DB-MWF to the multi-speaker case, and to fully connected WSNs with any number of nodes. Further extensions are simultaneous node updating (Chapter 3), robust estimation (Chapter 4), tree topology networks (Chapter 5), and node-specific distributed LCMV beamforming (Chapter 6).

1.3.4 Source Coding in WSNs

All techniques that were previously addressed aimed to estimate a signal in a distributed fashion, by fusing multi-sensor observations into a final estimate, while decreasing the required bandwidth based on dimensionality reduction. To further reduce the bandwidth, and to match specific rate constraints in the wireless links, the transmitted signals must be encoded with efficient source coding techniques, and then reconstructed at the the receiving node. The goal is then to transmit a signal as efficient as possible, without adding too much distortion between the original and the decoded signal. Although coding and estimation are actually different research fields, they have an important mutual interaction in a WSN context. Indeed, if the distortion due to coding is large, this may result in a significant performance decrease in the estimation algorithm.

[19] Experiments in [40], demonstrate that the DB-MWF algorithm usually converges after two or three iterations.

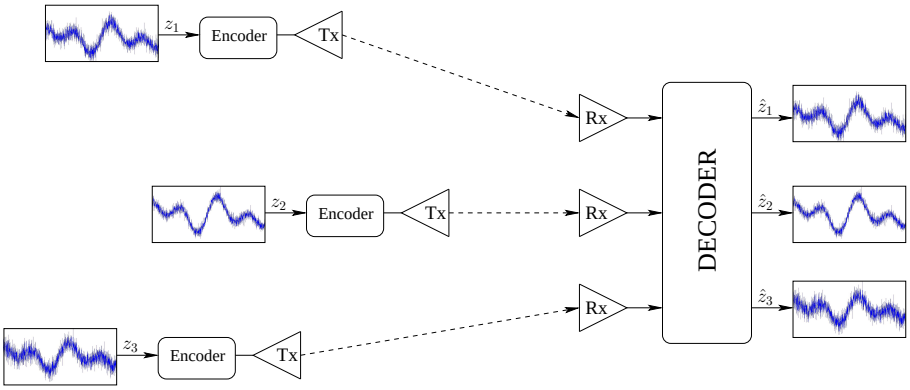


Figure 1.12: A typical source coding scenario in a WSN: three nodes encode their locally preprocessed signals and transmit it to a fourth node, who decodes all three signals.

Theoretical optimal bounds can be derived for many source coding problems based on rate-distortion theory, which is an important branch of information theory [70]. These bounds are described by so-called rate-distortion curves, which show the minimum bit rate that needs to be available to not exceed a specific distortion value or, vice versa, the minimum distortion that can be achieved for a given bit rate. However, these are theoretical bounds based on information theory, but they do not provide any practical coding scheme to achieve these bounds. Another important aspect of coding theory is then to construct coding schemes that aim to reach the bounds given by rate-distortion curves.

Since all signal estimation algorithms in the following chapters assume lossless transmission of data between nodes, we only briefly address some state-of-the-art results on distributed source coding in WSNs. However, the impact of source coding on DB-MWF (see Subsection 1.3.3) and on the DANSE algorithm (see Chapter 2) has been investigated in other work [71, 72].

Source Coding vs. Estimation in WSNs

A typical distributed source coding scenario is depicted in Fig. 1.12. The receiving node on the right collects encoded versions of three different signals (z_1, z_2 and z_3) from three different nodes. The decoder at the receiving node needs to decode all three original signals and provide the reconstructed signals ($\hat{z}_1$, $\hat{z}_2$ and $\hat{z}_3$) to the local estimation algorithm. For node 1, the goal is thus to transmit the signal z_1 with the smallest possible distortion, i.e., minimize $E\{|z_1 - \hat{z}_1|^2\}$, given a certain available bit rate in the wireless link. It is noted that this problem is significantly different from the compress-and-fuse problem

statement described in Subsection 1.3.1 (see Fig. 1.9) or distributed estimation in general, where the main goal is to estimate a hidden signal while reducing the dimensionality of the transmitted data at the same time. In the source coding problem, a signal is given that needs to be preserved as a whole, i.e., there is no distinction between desired and noise signal components. Notice that, in Fig. 1.12, the goal is to reconstruct *all* the transmitted signals at the receiver, while in Fig. 1.9, the goal is to estimate a signal component that is hidden in the data that is obtained from the sensor nodes. Estimation and coding can be viewed as complimentary techniques at different layers: the former is solved in the application layer, while the latter is solved in the coding/communication layer. There is also limited work where both layers are jointly analyzed in an information-theoretic framework [73].

Side Information Unaware vs. Side Information Aware Coding

The encoders in the distributed source coding scenario in Fig. 1.12 can be designed in two different ways. The simplest way is to merely encode the signal z_1 by removing the inherent redundancy in the signal z_1 itself. This is often referred to as 'side information unaware' (SIU) coding, since it ignores the mutual information in the signals of other nodes. However, since the receiving node also has access to encoded versions of z_2 and z_3, and since these signals usually contain a lot of the inherent information in z_1, the latter can be transmitted with significantly less bits while keeping the same distortion. This is referred to as 'side information aware' (SIA) coding, i.e., the encoders are designed to jointly remove the mutual redundancy in all the signals z_1, z_2 and z_3. Obviously, SIA usually performs better than SIU, but the former cannot be designed without prior knowledge on the mutual information in z_1, z_2 and z_3.

A remarkable and very important result for the SIA case, was established by Slepian and Wolf [74], and later generalized by Wyner and Ziv [75]. They proved that, to optimally encode z_1 in an information-theoretic sense, the signals z_2 and z_3 do not need to be available at node 1, i.e., there exists an optimal encoding scheme that can be implemented without providing the encoder with instantaneous signal observations of the other signals that are available in the decoder. However, the encoder at node 1 must have certain knowledge on the cross-correlation structure between all three signals, but does not need the actual signals z_2 and z_3 to optimally encode z_1. Therefore, SIA coding is able to provide optimal encoding schemes in distributed architectures such as WSNs. Furthermore, it shifts the computational complexity from the encoder side to decoder side.

The results established in [74] and [75] are however theoretical, and do not provide any insight in how to achieve this optimal encoding. Practical (but suboptimal) approaches for SIA coding and rate-constrained coding in WSNs

can be found in, e.g., [76–78]. Besides a practical bit allocation scheme for compression, [78] also describes a linear (suboptimal) distributed compression technique, referred to as the distributed Karhunen-Loève transform (DKLT), since it is a distributed implementation of the traditional and well-known KLT [66]. The DKLT aims for dimensionality reduction of local multi-dimensional sensor data at the nodes, while taking into account that the receiving node has extra side information obtained from other nodes. Similar to the general compress-and-fuse problem (see Chapter 1.3.1), it is solved by an offline Gauss-Seidel type iteration. Again, it is noted that there is a subtle difference between DKLT and compress-and-fuse techniques, in the fact that the former aims to compress data for rate-constrained transmission with small distortion, while the latter compresses data with the goal to have a good final estimate in the fusion center.

Influence of Distributed Source Coding for Distributed Acoustical Noise Reduction

Most literature on distributed speech enhancement focuses on the specific case of binaural hearing aids where compressed audio signals are transmitted between the two hearing aids over a wireless link. The influence of SIU source coding in several distributed speech enhancement algorithms has also been investigated in such a BHA context [68, 71, 73]. For a given bit rate in the wireless link, information-theoretic bounds for several noise reduction algorithms are derived. These bounds can only be achieved in practice if the optimal source coding scheme is known for the signals that are transmitted, which is not the case. However, they provide useful theoretical insights in the rate-constrained distributed noise reduction problem for BHAs.

In [68], the influence of optimal source coding on the suboptimal decoupled noise reduction strategies listed in Subsection 1.3.3 is investigated for different bit rates. The case of distributed MWF is investigated in [71], where it is demonstrated that DB-MWF also performs well in rate-constrained scenarios with low bit-rates, unless the available bit rate is shared between multiple iterations of the DB-MWF, i.e., if DB-MWF would be used for iterative refinement of a single data block. The influence of SIU source coding in the DANSE algorithm, introduced in Chapter 2, has also been investigated in an information-theoretic framework [72].

Besides the case of SIU-coding, [73] also provides bounds for SIA rate-constrained signal estimation in BHAs, without distinguishing between estimation and source coding as independent cascaded techniques. This means that rate-distortion curves are derived for the case where the distortion is defined as the MSE between the *speech component* in the reference sensor and the *estimated signal* at the HA output. For a given bit rate, this yields bounds on the optimal performance of *any* distributed noise reduction technique. These

rate-distortion curves can however only be computed if both the speech and the noise sources are modeled as jointly Gaussian stationary random processes, which usually does not match with reality, especially in the case of speech signals which are known to follow a Laplacian distribution [79]. Furthermore, it is doubtful that these optimal bounds can be achieved without exploiting prior knowledge on the mutual information between the microphone signals.

For a scenario where the noise is uncorrelated over the microphones, SIA coding is only useful for high SNR scenarios. If the SNR decreases, the benefit of SIA coding over SIU disappears, and both techniques give almost the same rate-distortion curves if the SNR is 0 dB [73]. The authors also address the case where the bit rate of the wireless link needs to be divided by the left and the right HA to obtain minimum overall distortion. The optimal strategy for SIA coding is then to let the HA with smallest SNR not transmit any data, unless the total available bit rate is larger than a given threshold. It is re-iterated that all these results only hold in scenarios with spatially uncorrelated noise and jointly Gaussian stationary random processes. In practical experiments, it is found that SIA is only beneficial over SIU in simple acoustic scenarios with a small amount of localized noise sources.

1.4 Techniques for Distributed Parameter Estimation in WSNs

In this section, we give an overview of some state-of-the-art linear parameter estimation techniques for wireless sensor networks. As mentioned in Section 1.1.3, parameter estimation techniques usually operate at low sampling rates, or they estimate or track a fixed set of variables over time. These techniques can therefore rely on iterative refinement of intermediate estimates. Due to the large amount of literature on linear parameter estimation in WSNs, we restrict this overview to a particular selection of algorithms and estimation problems that are more or less related to the contributions in this thesis.

1.4.1 Consensus Averaging

One of the most elegant and best-known distributed parameter estimation techniques is consensus averaging (CA). Assuming an ad hoc connected sensor network containing J nodes (the set of nodes is denoted by $\mathcal{J}$), where each node observes an M-dimensional vector $\mathbf{y}_k$, $\forall\, k \in \mathcal{J}$, then the goal of CA is to compute the average vector $\overline{\mathbf{y}} = \frac{1}{J}\sum_{k\in\mathcal{J}} \mathbf{y}_k$ in a distributed iterative fashion. This is the BLUE estimator for the hidden parameter $\mathbf{d}$ if the observation $\mathbf{y}_k$ is assumed to satisfy the sensor observation model

$$\mathbf{y}_k = \mathbf{d} + \mathbf{n}_k \tag{1.43}$$

where the noise $\mathbf{n}_k$ is zero-mean, spatially uncorrelated and white ($E\{\mathbf{n}_k\mathbf{n}_k^T\} = \mathbf{I}_M$).

The basic CA algorithm is given by the following iterative procedure:

The CA Algorithm

1. Each node collects a sensor measurement $\mathbf{y}_k^0$, $\forall\, k \in \mathcal{J}$.
2. Choose weighting coefficients α_{kq}, $\forall\, k \in \mathcal{J}, \forall\, q \in \mathcal{N}_k \cup \{k\}$.
3. $i \leftarrow 0$
4. Repeat for all nodes $k \in \mathcal{J}$ simultaneously:
   - Node k transmits $\mathbf{y}_k^i$ to all of its neighboring nodes in $\mathcal{N}_k$.
   - Node k updates its local estimate according to

$$\mathbf{y}_k^{i+1} = \sum_{q \in \mathcal{N}_k \cup \{k\}} \alpha_{kq} \mathbf{y}_q^i \,. \tag{1.44}$$

   - $i \leftarrow i + 1$

The CA algorithm iteratively refines each local estimate based on a weighted average of its own previous estimate and previous estimates of neighboring nodes. Let $\mathbf{A}$ denote the network-wide weighting matrix, i.e., the entry on the k-th row and q-th column of $\mathbf{A}$ is equal to α_{kq} if nodes k and q are connected, and zero otherwise. It can then be shown that [80, 81]

$$\lim_{i \to \infty} \mathbf{y}_k^i = \overline{\mathbf{y}},\ \ \forall\, k \in \mathcal{J} \tag{1.45}$$

if and only if

$$\mathbf{1}^T\mathbf{A} = \mathbf{1}^T,\ \ \mathbf{A}\mathbf{1} = \mathbf{1},\ \ \rho\left(\mathbf{A} - \frac{1}{J}\mathbf{1}\mathbf{1}^T\right) < 1 \tag{1.46}$$

where $\rho(.)$ denotes the spectral radius of a matrix, i.e., the eigenvalue with largest absolute value. If (1.46) is satisfied, then the estimate in each node will converge to $\overline{\mathbf{y}}$, i.e., the nodes will reach a consensus.

For a fixed graph, the weighting matrix $\mathbf{A}$ can be optimized to obtain the fastest asymptotic convergence rate, based on semi-definite programming [80]. However, in an ad hoc deployed WSN, it is preferable to compute a weighting matrix, satisfying (1.46), in a distributed fashion without computationally expensive algorithms. Two popular rules can be used to this end:

- ***Maximum-degree weights***:

$$\alpha_{kq} = \begin{cases} \frac{1}{J} & \text{if } q \in \mathcal{N}_k \\ 1 - \frac{|\mathcal{N}_k|}{J} & \text{if } q = k \\ 0 & \text{otherwise} \end{cases} \tag{1.47}$$

- ***Metropolis weights***:

$$\alpha_{kq} = \begin{cases} \frac{1}{1+\max\{|\mathcal{N}_k|,|\mathcal{N}_q|\}} & \text{if } q \in \mathcal{N}_k \\ 1 - \sum_{l \in \mathcal{N}_k} \alpha_{kl} & \text{if } q = k \\ 0 & \text{otherwise} \end{cases} \tag{1.48}$$

If the network graph is connected, both techniques yield weighting matrices that satisfy (1.46). The Metropolis weights are only based on local information, whereas for the maximum-degree weights, the number of nodes J needs to be known to each node (or at least an upper bound[20]).

There is a large amount of literature on the CA problem, with many extensions, e.g., CA in stochastic graphs with randomly changing topology [82], adaptive weighting matrices [83], and running consensus averaging where new observations are included during the averaging step [84].

1.4.2 Distributed Regression Problems

Another important parameter estimation problem for WSNs is distributed linear regression and its time-recursive implementations, extending well-known linear adaptive filtering algorithms, such as least mean squares (LMS) [13, 20, 22, 24, 25], recursive least squares (RLS) [21, 26], affine projection algorithm (APA) [14], Kalman filtering [85], etc., to the distributed case.

The distributed linear regression problem is formulated as follows. Consider an ad hoc WSN with the set of nodes $\mathcal{J} = \{1, \ldots, J\}$ and with a random (connected) topology. Node k collects observations of an $M \times P$ data matrix $\mathbf{U}_k$ and an M-dimensional data vector $\mathbf{d}_k$. The goal is then to solve a network-wide least-squares (LS) problem, i.e. to compute the P-dimensional regression vector $\hat{\mathbf{w}}$ defined by

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) = \sum_{k \in \mathcal{J}} \|\mathbf{d}_k - \mathbf{U}_k \mathbf{w}\|^2 \ . \tag{1.49}$$

The goal is to find the common network-wide regression vector $\hat{\mathbf{w}}$ in a distributed fashion, without gathering all the data in a fusion center.

The above regression problem can also be analyzed in a stochastic framework, where the rows of $\mathbf{U}_k$ correspond to observations of a stochastic row-vector[21]

[20] Replacing J in (1.47) with $N \geq J$ still yields a weighting matrix that satisfies (1.46).

[21] The regressor vector $\mathbf{u}_k$ is defined as a row vector to be consistent with the notation in Chapter 8, which adopts the notation of [86].

variable $\mathbf{u}_k$, and where the entries in $\mathbf{d}_k$ correspond to observations of a scalar stochastic variable d_k. Assume that these processes are related to each other according to the linear model

$$d_k(i) = u_{k,i}\mathbf{w}^o + v_k(i) \tag{1.50}$$

where $d_k(i)$ is the i-th observation of d_k, $u_{k,i}$ is the i-th observation[22] of $\mathbf{u}_k$, $\mathbf{w}^o$ is the fixed parameter vector that we want to estimate, and $v_k(i)$ is spatially uncorrelated white noise, then $\hat{\mathbf{w}}$ is the BLUE estimator for $\mathbf{w}^o$. It is noted that this is an unbiased estimate, only if the observed $u_{k,i}$'s are the same as in (1.50), i.e., the rows of $\mathbf{U}_k$ are not corrupted by noise. In Chapters 7 and 8, we consider the case where there is indeed noise on the regressors $u_{k,i}$, and we propose two different methods to improve the estimation.

In adaptive scenarios, the latent vector $\mathbf{w}^o$ may change over time, and the number of observations collected by each sensor will also increase over time (i.e. M increases). It is then preferred to track the changes in $\mathbf{w}$ by incorporating the new samples in the estimation, and by removing the influence of older samples. This is the domain of adaptive filtering [86, 87], where $\mathbf{w}$ can be viewed as a linear adaptive filter with a signal u_k as an input, and d_k as an output (at node k). The rows of $\mathbf{U}_k$ then consist of subsequent samples of u_k (each row corresponds to a different time window), and the vector $\mathbf{d}_k$ contains corresponding output samples of d_k.

There are three well-known strategies to solve the aforementioned deterministic and/or adaptive distributed regression problem: consensus-based strategies, incremental strategies and diffusion strategies. In the sequel, we will briefly describe all three strategies, and apply them to LMS-type adaptive-filtering techniques.

Consensus-based strategies

The idea of consensus-based linear regression is to decouple the network-wide LS cost function (1.49) by letting each node compute a local estimate of $\mathbf{w}$, and then to explicitly add consensus constraints to obtain the same estimate in each node. To this end, the optimization problem (1.49) is transformed into the constrained optimization problem

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J} \sum_{k\in\mathcal{J}} \|\mathbf{d}_k - \mathbf{U}_k\mathbf{w}_k\|^2 \tag{1.51}$$

$$\text{s.t. } \mathbf{w}_k = \mathbf{w}_q,\ k \in \mathcal{J},\ q \in \mathcal{N}_k\ . \tag{1.52}$$

Due to the coupling of the optimization variables in each node, based on the consensus constraints (1.52), the optimal $\mathbf{w}_k$'s will all be equal to the optimal solution $\hat{\mathbf{w}}$ of (1.49). The cost function can now be decoupled, such that

[22] In this section, we use the notation of [86], i.e., observations of stochastic vector variables are not written in bold face.

each node can locally compute a term of (1.51). However, due to the consensus constraints, the problem is still not fully separable into local optimization problems. To obtain a fully distributed algorithm, the optimization problem is solved by means of 'dual decomposition' [67, 88]. Without going into detail (dual decomposition is also addressed in Chapter 7), it can be shown that (1.51)-(1.52) can be solved iteratively with the following updating steps, which are based on the method of multipliers[23] (MoM) [25]:

1. $\forall\, k \in \mathcal{J}, \forall\, q \in \mathcal{N}_k$, initialize the P-dimensional Lagrange multiplier vector $\boldsymbol{\lambda}_{kq}^0$ with a random real value.
2. $\forall\, k \in \mathcal{J}$, initialize the P-dimensional local estimator $\mathbf{w}_k^0$ with a random real value.
3. $i \leftarrow 0$.
4. Choose a stepsize $c > 0$.
5. Perform the following updates for all nodes $k \in \mathcal{J}$ simultaneously, until convergence:
   - $\forall\, q \in \mathcal{N}_k : \boldsymbol{\lambda}_{kq}^{i+1} = \boldsymbol{\lambda}_{kq}^{i} + \frac{c}{2}\left(\mathbf{w}_k^i - \mathbf{w}_q^i\right)$.
   - $\mathbf{w}_k^{i+1} = \arg\min_{\mathbf{w}_k} \left\|\mathbf{d}_k - \mathbf{U}_k\mathbf{w}_k^i\right\|^2 + \sum_{q\in\mathcal{N}_k}(\boldsymbol{\lambda}_{kq}^{i+1} - \boldsymbol{\lambda}_{qk}^{i+1})^T\mathbf{w}_k + c\sum_{q\in\mathcal{N}_k}\|\mathbf{w}_k - \frac{1}{2}(\mathbf{w}_k^i - \mathbf{w}_q^i)\|^2.$
   - $i \leftarrow i+1$

Although this procedure converges for any value of c, the latter should be tuned to obtain a practical convergence speed. It is noted that, besides the local estimates (the $\mathbf{w}_k$'s), the multipliers (the $\boldsymbol{\lambda}_{kq}$'s) also need to be communicated between neighboring nodes. This can be eliminated if the latter are initialized as $\boldsymbol{\lambda}_{kq}^0 = -\boldsymbol{\lambda}_{qk}^0$, since in this case $\boldsymbol{\lambda}_{kq}^i = -\boldsymbol{\lambda}_{qk}^i$, $\forall\, i \in \mathbb{N}$. However, in [25], it is shown that exchanging the multipliers makes the algorithm more robust to communication noise in the wireless links (this holds for all MoM-based distributed algorithms).

The distributed least-squares regression algorithm described above solves the fixed deterministic optimization problem (1.49). In each iteration, a local unconstrained quadratic optimization problem is solved at each node. To make the algorithm adaptive, e.g. to track changes in $\mathbf{w}^o$, it is sufficient to replace the matrix $\mathbf{U}_k$ and the data vector $\mathbf{d}_k$ with their updated versions in each iteration, including the most recently collected samples, i.e., they become dependent on the iteration index i. Assuming that the solutions of the local optimization problems do not change much over different iterations, it is also possible to rely on time-recursive algorithms. For example, in [25], a stochastic gradient descent approach is used, as in the well-known LMS algorithm [86, 87]. However, the local optimization problem is then not fully solved in each iteration, and there-

[23] This updating procedure is the result of an alternating-direction method of multipliers (AD-MoM) [67], which is a particular dual decomposition algorithm to solve problems like (1.51)-(1.52). More details can be found in [25].

fore convergence of the algorithm cannot be guaranteed anymore. Nevertheless, this approach seems to yield good estimation and tracking performance, and it can be analyzed in an adaptive filtering framework [25]. Consensus-based RLS-type algorithms have also been derived in [26].

A similar consensus-based technique is used in [89] for sparse linear regression, based on the least-absolute shrinkage and selection operator (LASSO), which is capable of performing both estimation and variable selection. In Chapter 7, we propose another[24] consensus-based strategy to solve a distributed total least squares (D-TLS) problem, which can cope with white noise[25] in both the $\mathbf{U}_k$'s and the $\mathbf{d}_k$'s. However, due to the non-convex norm constraint in the problem statement of D-TLS, some additional techniques are required before the dual decomposition can be performed.

Incremental strategies

Incremental strategies are distributed approximations of a gradient-descent algorithm, exploiting the separability of the gradient of the LS cost function (1.49), which is given by

$$\nabla J(\mathbf{w}) = 2\sum_{k=1}^{J} (\mathbf{R}_{U_k}\mathbf{w} - \mathbf{r}_{U_k d_k}) \tag{1.53}$$

with

$$\mathbf{R}_{U_k} = \mathbf{U}_k^T\mathbf{U}_k \tag{1.54}$$

$$\mathbf{r}_{U_k d_k} = \mathbf{U}_k^T\mathbf{d}_k \ . \tag{1.55}$$

A (centralized) gradient-descent algorithm would then apply the following updating procedure to find the solution $\hat{\mathbf{w}}$ (we omit the factor 2 in the gradient):

$$\mathbf{w}^{i+1} = \mathbf{w}^i - \mu\nabla J(\mathbf{w}^i) \tag{1.56}$$

$$= \mathbf{w}^i - \mu\sum_{k=1}^{J} \left(\mathbf{R}_{U_k}\mathbf{w}^i - \mathbf{r}_{U_k d_k}\right) \tag{1.57}$$

with a small enough stepsize $\mu > 0$ to achieve convergence. In the distributed case, the full gradient vector cannot be computed, since the $\mathbf{R}_{U_k}$'s and $\mathbf{r}_{U_k d_k}$'s are distributed over the different nodes. Assume (hypothetically) that each node has access to the current estimate $\mathbf{w}^i$, and define a so-called Hamiltonian cycle, i.e., a cyclic path through the network that visits each node once (see Fig. 1.13), then (1.57) can be computed in a distributed fashion:

[24] The derivation of the distributed total least squares algorithm in Chapter 7 is also based on dual decomposition, but it does not use the AD-MoM, but a dual-based subgradient algorithm (DBSA) instead.

[25] A traditional least-squares estimate, as in (1.49), is biased when the data matrix $\mathbf{U}_k$ is contaminated with white noise.

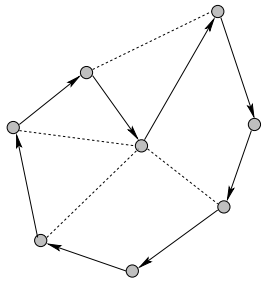


Figure 1.13: Information exchange in incremental strategies.

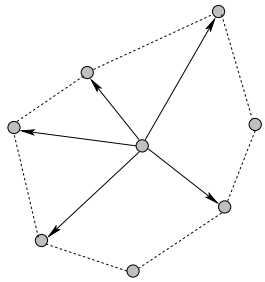


Figure 1.14: Information exchange in diffusion strategies.

- $\boldsymbol{\psi}_0 \leftarrow \mathbf{w}^i$
- $\boldsymbol{\psi}_k \leftarrow \boldsymbol{\psi}_{k-1} - \mu \left(\mathbf{R}_{U_k} \mathbf{w}^i - \mathbf{r}_{U_k d_k} \right),\ k = 1, \ldots, J$
- $\mathbf{w}^{i+1} \leftarrow \boldsymbol{\psi}_J$

where we assume that the nodes are labeled according to their respective order in the Hamiltonian cycle. In each iteration, node $k-1$ forwards its partially updated gradient $\boldsymbol{\psi}_{k-1}$ to the next node in the cycle. However, also $\mathbf{w}^i$ must be transmitted in each step, to provide every node with the current value of the estimate, and then the required communication bandwidth is effectively doubled. To avoid the latter, the gradient-descent algorithm can be approximated as follows:

- $\boldsymbol{\psi}_0 \leftarrow \mathbf{w}^i$
- $\boldsymbol{\psi}_k \leftarrow \boldsymbol{\psi}_{k-1} - \mu \left(\mathbf{R}_{U_k} \boldsymbol{\psi}_{k-1} - \mathbf{r}_{U_k d_k} \right),\ k = 1, \ldots, J$
- $\mathbf{w}^{i+1} \leftarrow \boldsymbol{\psi}_J$.

Notice that we just replaced $\mathbf{w}^i$ with the new intermediate estimate $\boldsymbol{\psi}_{k-1}$. This is known as an incremental technique, which has been studied extensively in literature, e.g., [90]. In [2, Ch. 22], it is shown that this incremental procedure converges as fast as the steepest-descent algorithm for vanishing stepsizes. Furthermore, the incremental procedure converges over a wider range of stepsizes, especially so when the number of nodes is large.

In an adaptive (stochastic) framework, the LS estimation becomes an LMMSE estimation, and the $\mathbf{R}_{U_k}$'s and $\mathbf{r}_{U_k d_k}$'s can then be replaced with second order statistics, i.e., $\mathbf{R}_{u_k} = E\{\mathbf{u}_k^T \mathbf{u}_k\}$ and $\mathbf{r}_{u_k d_k} = E\{\mathbf{u}_k^T d_k\}$, respectively. Instead of using these second-order statistics in the incremental procedure given above, we can replace them with instantaneous estimates, i.e. $\mathbf{R}_{u_k} \approx u_{k,i}^T u_{k,i}$ and $\mathbf{r}_{u_k d_k} \approx u_{k,i}^T d_k(i)$ where $u_{k,i}$ and $d_k(i)$ denote a sample of $\mathbf{u}_k$ and d_k, respectively, collected at iteration i. We then obtain the incremental LMS algorithm [13]:

The incremental LMS algorithm

1. Initialize $i \leftarrow 0$ and initialize $\mathbf{w}^0$ with a random P-dimensional vector.
2. $\boldsymbol{\psi}_0 \leftarrow \mathbf{w}^i$
3. $\boldsymbol{\psi}_k \leftarrow \boldsymbol{\psi}_{k-1} - \mu u_{k,i}^T \left(d_k(i) - u_{k,i}\boldsymbol{\psi}_{k-1}\right),\ k = 1, \ldots, J$
4. $\mathbf{w}^{i+1} \leftarrow \boldsymbol{\psi}_J$
5. $i \leftarrow i + 1$
6. Return to step 2.

A detailed mean-square and stability analysis of this algorithm is performed in [13], where it is shown that the cooperation yields an equalization effect on the variance of the estimators throughout the network, i.e., the variance of the estimators is the same for each node. A similar incremental strategy is described for the affine projection algorithm in [14].

Incremental algorithms usually yield good estimation performance, i.e., better than the diffusion-type algorithms that are described below. Furthermore, they have the advantage that each node only needs to communicate with one other node, which is efficient in terms of both the communication and computational load. There are however a couple of disadvantages, i.e., the fact that a Hamiltonian cycle needs to be defined, and that the network processing has to be faster than the measurement process, since a full communication cycle is required for each measurement. Furthermore, incremental strategies are not robust to node and link failure.

Diffusion Strategies

The drawbacks of incremental cooperation can be avoided at the price of a slightly worse estimation performance and larger communication and computational load. In a diffusion-based technique (Fig. 1.14), every node communicates with all of its neighbors, as dictated by the network. The need for a Hamiltonian cycle is therefore eliminated, yielding distributed algorithms that are robust to node and link failure. It is noted that, if the communication protocol allows local broadcasts rather than reserved links between node pairs, the required communication bandwidth is similar to the incremental mode.

To emphasize the close relationship between diffusion and consensus averaging (see Subsection 1.4.1), we explain diffusion slightly different then in [20, 22]. We start with combining the stochastic gradient-descent algorithm with consensus averaging. To this end, let us define the gradient of the local cost function at

node k:

$$\nabla J_k(\mathbf{w}_k) = \mathbf{R}_{U_k}\mathbf{w}_k - \mathbf{r}_{U_k d_k} \tag{1.58}$$

where J_k denotes one node-specific term of the network-wide LS cost function (1.49), i.e., $J(\mathbf{w}) = \sum_{k\in\mathcal{J}} J_k(\mathbf{w})$. Similarly, define the local gradient-descent step:

$$\boldsymbol{\psi}_k^i = \mathbf{w}_k^i - \overline{\mu}\nabla J_k(\mathbf{w}_k^i) \; . \tag{1.59}$$

If we assume that there is consensus on the current estimate at iteration i, i.e., $\mathbf{w}_k^i = \mathbf{w}^i$, $\forall\, k \in \mathcal{J}$, then the centralized gradient-descent algorithm (1.57) can be computed as the average of the local gradient-descent steps, i.e.,

$$\mathbf{w}^{i+1} = \frac{1}{J}\sum_{k\in\mathcal{J}} \boldsymbol{\psi}_k^i \tag{1.60}$$

where we assume that μ in (1.57) is equal to $\mu = \frac{\overline{\mu}}{J}$. Expression (1.60) can be iteratively computed by means of a CA algorithm, which would require a second iteration index j that runs on the level of the CA algorithm. We then obtain the following procedure:

1. Initialize $i \leftarrow 0$ and $\mathbf{w}_k^0 = \mathbf{w}$, $\forall\, k \in \mathcal{J}$, where $\mathbf{w}$ is a random P-dimensional vector.
2. At each node $k \in \mathcal{J}$ simultaneously, perform the local gradient-descent step
$$\boldsymbol{\psi}_k^0 = \mathbf{w}_k^i - \overline{\mu}\nabla J_k(\mathbf{w}_k^i) \; . \tag{1.61}$$
3. Set $j \leftarrow 0$ and perform the following CA steps for each node $k \in \mathcal{J}$ simultaneously, until convergence:
   - $\boldsymbol{\psi}_k^{j+1} = \sum_{q\in\mathcal{N}_k\cup\{k\}} \alpha_{kq}\boldsymbol{\psi}_q^j$
   - $j \leftarrow j+1$.
4. At each node $k \in \mathcal{J}$, set $\mathbf{w}_k^i = \boldsymbol{\psi}_k^\infty$, where $\boldsymbol{\psi}_k^\infty$ is the result from the previous step at node k.
5. $i \leftarrow i+1$.
6. Return to step 2.

If $\overline{\mu}$ is chosen small enough, then this procedure will converge to the optimal vector in each node, i.e., $\mathbf{w}_k^\infty = \hat{\mathbf{w}}$, $\forall\, k \in \mathcal{J}$. This directly follows from convergence of CA and gradient descent. However, for each gradient-descent iteration, a large number of intermediate estimates have to be exchanged between nodes to obtain consensus over the global gradient. Especially in adaptive scenarios, where $\mathbf{w}^o$ changes over time, this approach is too slow or too computationally intensive. The idea of adaptive diffusion is then to merge the two iteration indices i and j, and to relax the consensus assumption, yielding the following algorithm:

1. Initialize $i \leftarrow 0$ and $\mathbf{w}_k^0 = \mathbf{w}$, $\forall\, k \in \mathcal{J}$, where $\mathbf{w}$ is a random P-dimensional vector.

2. At each node $k \in \mathcal{J}$ simultaneously, compute

$$\boldsymbol{\psi}_k^i = \mathbf{w}_k^i - \overline{\mu}\nabla J_k(\mathbf{w}_k^i) \tag{1.62}$$

   and broadcast the result to the nodes in $\mathcal{N}_k$.
3. At each node $k \in \mathcal{J}$ simultaneously, compute

$$\mathbf{w}_k^{i+1} = \sum_{q\in\mathcal{N}_k\cup\{k\}} \alpha_{kq}\boldsymbol{\psi}_q^i \; . \tag{1.63}$$

4. $i \leftarrow i+1$.
5. Return to step 2.

By replacing the gradient $\nabla J_k(\mathbf{w}_k^i)$ with a stochastic gradient based on instantaneous second-order statistics, we obtain the diffusion LMS algorithm from [20, 22]:

The diffusion LMS algorithm

1. Initialize $i \leftarrow 0$ and $\mathbf{w}_k^0 = \mathbf{w}$, $\forall\, k \in \mathcal{J}$, where $\mathbf{w}$ is a random P-dimensional vector.
2. At each node $k \in \mathcal{J}$ simultaneously, compute

$$\boldsymbol{\psi}_k^i = \mathbf{w}_k^i - \overline{\mu}u_{k,i}^T\left(d_k(i) - u_{k,i}\mathbf{w}_k^i\right) \tag{1.64}$$

   and broadcast the result to the nodes in $\mathcal{N}_k$.
3. At each node $k \in \mathcal{J}$ simultaneously, compute

$$\mathbf{w}_k^{i+1} = \sum_{q\in\mathcal{N}_k\cup\{k\}} \alpha_{kq}\boldsymbol{\psi}_q^i \; . \tag{1.65}$$

4. $i \leftarrow i+1$.
5. Return to step 2.

The weighting coefficients α_{kq} at node k have to be non-negative and add up to one, i.e.

$$\sum_{q\in\mathcal{N}_k\cup\{k\}} \alpha_{kq} = 1, \;\; \forall\, k \in \mathcal{J} \; . \tag{1.66}$$

It can be shown that, if (1.66) holds and if $\overline{\mu}$ is small enough, the estimates $\{\mathbf{w}_k^i\}$ are stable in the mean and unbiased , i.e., $E\{\mathbf{w}_k^i - \mathbf{w}^o\} = 0$, if $i \to \infty$. Similarly to an incremental cooperation mode, diffusion also improves the stability of the algorithm, i.e., it converges over a wider range of stepsizes.

It should be noted that the above algorithm is a special case of the 'adapt-then-combine' (ATC) diffusion LMS algorithm, as introduced in [22], since node k first adapts its previous estimate $\mathbf{w}_k^i$ according to its new local observations, and then combines this with the intermediate estimates $\boldsymbol{\psi}_q^i$ of its neighbors to compute the final estimate. The exchanged intermediate estimates then also contain the most recent information of the nodes in $\mathcal{N}_k$. In the 'combine-then-adapt' (CTA) approach [20, 22] node k first combines its previous estimates with those of its neighbors, and then updates this combined intermediate estimate with a stochastic gradient descent step based on this combined estimate and the node's last local observation. Intuitively, since the most recent observations of its neighbors are not incorporated in the case of CTA, ATC should perform better[26] than CTA. In [22], this is indeed confirmed based on a theoretical steady-state analysis and by means of simulations.

The diffusion strategy is also applicable to RLS implementations instead of LMS [21]. However, both the LMS estimate and the RLS estimate are biased in the case when there is noise on the observations of the regressors $\{\mathbf{u}_k\}$. If this noise is white, the distributed total least squares algorithm, derived in Chapter 7, provides an unbiased estimate of $\mathbf{w}^o$. In Chapter 8, we consider the case where the noise is colored, and we extend the RLS algorithm with a bias compensation. Removing the bias generally results in a larger variance on the estimate, and therefore diffusion is applied to limit this variance increase. Furthermore, we show that the diffusion-based cooperation also decreases the residual bias due to errors in the estimates of the noise statistics.

1.5 Problem Statement and Challenges

Most chapters in this thesis contribute to a general target application, i.e., distributed acoustic noise reduction for speech enhancement in WASNs. Noise reduction is crucial in many speech recording applications, such as hearing aids, mobile phones, video conferencing, hands-free telephony, automatic speech recognition, etc. By using a WASN, many more microphone signals become available, which can greatly improve the noise reduction performance in these applications. In many cases, the desired signal should be available in multiple devices (multiple nodes), e.g., in binaural HAs or collaborating HAs of multiple users. Furthermore, it is often desired that the node-specific localization cues are preserved in each node, e.g., for spatial hearing or when the noise reduction is followed by a speaker localization algorithm. This means that the signal

[26] At first sight, ATC and CTA appear to be the same algorithm, since they both alternate between the adaptation step (1.64) and the combination step (1.65). However, there is a subtle but important difference: CTA uses $\boldsymbol{\psi}_k$ from (1.64) as the final estimate whereas ATC uses $\mathbf{w}_k$ from (1.65), which could both serve as an estimate of $\mathbf{w}^o$. However, the latter can be shown to be a better estimate.

estimates at each node are node-specific, i.e., each node estimates the desired speech component as locally observed by one of its own microphones.

We aim to perform distributed noise reduction for speech enhancement in a WASN consisting of multiple wirelessly connected nodes, each having a local microphone array and a local processing unit, such that a node-specific noise-reduced speech signal is available in each node. The WASN should operate in complex acoustic scenarios with multiple desired sources and multiple noise sources. This is a computationally intensive signal estimation task due to the high sampling rates, and therefore communication bandwidth and processing power are considered to be highly limited resources, especially in small devices such as HAs. We therefore aim to reduce the amount of data that is communicated between the nodes, and to limit the local computational complexity. For reasons of scalability, and since each node has to be provided with a node-specific noise-reduced signal, we avoid the use of a fusion center, i.e., the computations have to be fully distributed. Traditional compress-and-fuse techniques (see Subsection 1.3.1) cannot be used in this case, since they rely on one-way communication between the microphones and a fusion center. Furthermore, we envisage an ad hoc placement of the microphone nodes, i.e., we cannot rely on prior knowledge of the microphone positions, array geometry, or the position of the sound sources, and we assume that the environment is dynamic, i.e., microphone and positions can change during operation of the algorithm. Therefore, the noise reduction procedure must be blind and able to swiftly adapt to changes in the environment, which is generally not the case in compress-and-fuse techniques, since they mostly require prior knowledge on sensor signal cross-correlations.

As a general target, we aim to develop distributed noise reduction algorithms that achieve the same noise reduction performance as in a centralized approach where the nodes have access to all the microphone signals of the network. This means that we cannot rely on ad hoc estimators such as in Subsection 1.3.2. Instead, we need to properly parametrize the optimal centralized estimator, such that its components can be distributed over the different nodes of the network. Eventually, we should obtain a noise reduction algorithm that operates in simply connected networks, where nodes only share (compressed) microphone signals with nodes in their local neighborhood. Besides the fundamental and theoretical aspects, the resulting techniques should also work effectively in practical acoustic scenarios. Therefore, the algorithm needs to be robust against numerically ill-conditioned situations, and it should be able to cope with sudden link failures, which are common in W(A)SNs. Furthermore, it is to be expected that there exists a significant set of microphone nodes that do not contribute much to the noise reduction due to lack of acoustical coupling with relevant sound sources or due to lack of coherence with other microphone signals. In this case, we also aim to select the subset of nodes that contributes the most in the noise reduction task, such that the less useful nodes can be put

to sleep to save energy, and to avoid noise injection from acoustically uncoupled nodes.

Since LCMV-beamforming and MWF are blind multi-channel noise reduction techniques, they are well-suited for noise reduction with ad hoc microphone arrays such as in WASNs. However, the lack of prior knowledge on microphone and sound source positions requires a robust VAD algorithm. In complex environments with multiple speakers, it is often required to have a VAD that can distinguish between different speakers. To this end, we can again rely on the significant amount of spatial information that becomes available when using a WASN.

Finally, when the WASN is used for speech analysis, e.g., for automatic speech recognition (ASR) or speech coding, the network should be able to extract relevant speech parameters from the noise-reduced microphone signals. For example, a well-known speech analysis technique is linear predictive coding (LPC), which extracts the auto-regressive (AR) coefficients that are representative for the different speech sounds or phonemes[27]. However, this linear regression technique is very sensitive to noise on the speech signals, even so when the signals are preprocessed with a noise reduction algorithm. To this end, we aim to exploit the cooperation between nodes to improve such linear parameter estimation based on noisy signals.

1.6 Thesis Contributions

In this section, we provide an overview of the different chapters in this thesis, and we briefly describe their main points, and how they relate to the state of the art described in the previous sections.

Part II: Distributed Signal Estimation Techniques

Chapter 2: Fully Connected DANSE with Sequential Node-Updating

In this chapter, we introduce the distributed adaptive node-specific signal estimation (DANSE) algorithm for linear minimum mean squared error (MMSE) estimation in a fully connected broadcasting sensor network. This is basically an extension of the DB-MWF (see Subsection 1.3.3) to the case of multiple nodes and multiple desired sources, which requires a completely different convergence and optimality proof. The estimation is node-specific because each

[27] It should be noted that LPC analysis is not the state of the art anymore for ASR feature extraction. However, it is still an important speech analysis technique in many other applications, e.g. in speech coding.

node estimates a different mixture of the target sources as observed by its local reference sensor. The DANSE algorithm exploits the fact that the node-specific desired signals at the nodes share a common latent signal subspace. The algorithm can then significantly reduce the required communication bandwidth and still provide the same optimal linear MMSE estimators as in the centralized case. DANSE is somewhat related to compress-and-fuse techniques (see Subsection 1.3.1) where each node acts as a different fusion center. However, DANSE has the important advantage that everything can be computed adaptively, and the required statistics can be estimated from the compressed sensor signals. This avoids the need of a training phase or prior knowledge on the cross-correlation between all sensor pairs. Because of its adaptive nature, the algorithm is suited for real-time signal estimation in dynamic environments, such as speech enhancement with acoustic sensor networks (see also Chapter 4). In this chapter, we only consider the case where nodes update their fusion rules in a sequential round-robin fashion. The case where the nodes update simultaneously is addressed in the next chapter. The influence of SIU source coding on the acoustic-noise reduction performance of DANSE has been investigated in [72].

Chapter 3: Fully Connected DANSE with Simultaneous Node-Updating

In the original DANSE algorithm, the nodes update their parameters in a sequential round-robin fashion, which may yield a slow convergence of the estimators, especially so when the number of nodes in the network is large. In this chapter, we consider the fully connected DANSE algorithm for the case where nodes update simultaneously. This allows the algorithm to adapt more swiftly, but simulations show that convergence can no longer be guaranteed. We then provide an extension to the DANSE algorithm, in which we apply an additional relaxation in the updating process. The new algorithm is then proven to converge to the optimal estimators when nodes update simultaneously or asynchronously, be it that the computational load at each node increases in comparison with the algorithm with sequential updates. Finally, based on simulations it is demonstrated that a simplified version of the new algorithm, without any extra computational load, can also provide convergence to the optimal estimators. Similar results are obtained for the case where nodes update asynchronously, i.e., each node decides for itself when and how often it updates its parameters.

Chapter 4: Robust DANSE for Speech Enhancement

In this chapter, the results obtained in Chapters 2 and 3 are applied to acoustical noise reduction in fully connected wireless acoustic sensor networks. This can be considered as an extension of DB-MWF in BHAs to the case where extra external acoustic sensor nodes are wirelessly connected to the BHA. The

benefit of using external sensor nodes for noise reduction is demonstrated in a simulated acoustic scenario with multiple sound sources. Batch-mode simulations compare the noise reduction performance of a centralized MWF algorithm with DANSE. In the simulated scenario, DANSE is observed not to be able to achieve the same performance as the centralized MWF, although in theory both should generate the same set of filters. A modification to DANSE is then proposed to increase its robustness, yielding smaller discrepancy between the performance of DANSE and the centralized MWF. This algorithm is referred to as robust-DANSE or R-DANSE. Furthermore, the influence of several parameters such as the DFT size used for frequency domain processing and possible delays in the communication link between nodes is investigated. It is noted that all results in this chapter also apply to the SDW-MWF estimation procedure as described in Subsection 1.2.2, i.e., it can be shown that DANSE and R-DANSE also converge to the optimal centralized SDW-MWF filters for any value of the weighting factor [91]. This also holds for the T-DANSE algorithm as described in Chapter 5.

Chapter 5: DANSE in Networks with Tree Topology

In this chapter, we extend the fully connected DANSE algorithm to operate in simply connected networks. Different from the distributed ad hoc estimator described in Subsection 1.3.2, we aim to obtain a node-specific estimator in each node that provides the same output as a centralized estimator. We show that this is only possible if there is no feedback in the signal paths. This motivates the use of a tree topology, since the latter does not have any loops in the network graph, and hence removes feedback paths. We refer to the new algorithm as tree-DANSE (T-DANSE). If the node-specific desired signals share a common latent signal subspace, it is shown that T-DANSE converges to the same linear MMSE solutions as obtained with the centralized version of the algorithm. The computational load is then shared between the different nodes in the network, and nodes exchange only linear combinations of their sensor signal observations and data received from their neighbors. Despite the low connectivity of the network and the multi-hop signal paths, the algorithm is fully scalable in terms of communication bandwidth and computational power. Two different cases are considered concerning the communication protocol between the nodes: point-to-point transmission and local broadcasting. The former assumes that there is a reserved communication link between node pairs, whereas with the latter, nodes communicate the same data to all of their neighbors simultaneously.

Chapter 6: Linearly-Constrained DANSE

In this chapter, we extend the DANSE algorithm with node-specific linear constraints to generate a node-specific LCMV beamformer (see Subsection 1.2.3) at each node. This is referred to as linearly constrained DANSE (LC-DANSE).

The LC-DANSE algorithm is again able to significantly reduce the number of signals that are shared between nodes, but obtains the node-specific LCMV beamformers as if each node has access to all the signals in the network. The number of signals that are broadcast by each node needs to be equal to the number of relevant sources, i.e., the source signals for which linear constraints are applied in at least one of the nodes. We formally prove convergence and optimality of the LC-DANSE algorithm under this assumption. We also consider the case where nodes update simultaneously, and we demonstrate with simulations that applying relaxation is often required to obtain a converging algorithm for this case. We provide application-oriented simulation results that demonstrate the effectiveness of the algorithm for speech enhancement in a wireless acoustic sensor network.

Part III: Distributed Parameter Estimation Techniques

Chapter 7: Distributed Total Least Squares

In this chapter, we consider a distributed linear regression problem, as described in Subsection 1.4.2, where both the right-hand side and the input data matrix are assumed to be noisy. For this case, the common least-squares estimation will have a bias, which may be undesired in some applications. Total least squares (TLS) is a popular solution technique to solve such linear regression problems with noisy data matrices, and usually results in significantly better estimates. Furthermore, it can be shown that the TLS estimate is unbiased if the noise is white. We consider a TLS problem in an ad hoc wireless sensor network, where each node collects observations that yield a node-specific subset of linear equations. The goal is to compute the TLS solution of the full set of equations in a distributed fashion, without gathering all these equations in a fusion center. Our method is based on a consensus approach (see Subsection 1.4.2). However, dual decomposition techniques, which are traditionally used in consensus-based optimization, cannot be used due to the non-convex nature of the TLS problem. To facilitate the use of the dual based subgradient algorithm (DBSA), we transform the TLS problem to an equivalent convex semidefinite program (SDP), based on semidefinite relaxation (SDR). This allows us to derive a distributed TLS (D-TLS) algorithm, that satisfies the conditions for convergence of the DBSA, and obtains the same solution as the original (unrelaxed) TLS problem. Even though we make a detour through SDR and SDP theory, the resulting D-TLS algorithm relies on solving local TLS-like problems at each node, rather than computationally expensive SDP optimization techniques. The algorithm is flexible and fully distributed, i.e. it does not make any assumptions on the network topology and nodes only share data with their neighbors through local broadcasts. Due to the flexibility and the uniformity of the network, there is no single point of failure, which makes the algorithm robust to sensor failures. Monte-Carlo simulation results

are provided to demonstrate the effectiveness of the method.

Chapter 8: Diffusion Bias-Compensated RLS Estimation

In this chapter, we again study the distributed linear regression problem, where both the right hand side and the input data matrix are assumed to be noisy. This time, we aim to fully remove the bias, even if the noise is colored. Furthermore, we tackle the problem in an adaptive filtering context, i.e., the nodes have a common objective to estimate and track a latent vector parameter. For example, this is a common problem in auto-regressive (AR) modeling of speech that is corrupted by additive noise. Assuming that the noise covariance can be estimated (or is known a priori), we first propose a bias-compensated recursive least-squares algorithm. However, this bias compensation increases the variance of the local estimates, and errors in the noise covariance estimates may still result in a residual bias. We demonstrate that the variance and the residual bias can be significantly reduced by applying diffusion adaptation, i.e. letting nodes combine their local estimate with those of their neighbors, similar to diffusion LMS described in Subsection 1.4.2. We derive a necessary and sufficient condition for mean-square stability of the algorithm, under some mild assumptions. Furthermore, we derive closed-form expressions for its steady-state mean and mean-square performance. Simulation results are provided, which agree well with the theoretical results. We also consider some special cases where the mean-square stability improvement of diffusion BC-RLS over undiffused BC-RLS can be mathematically verified.

Part IV: Supporting Techniques for Signal Estimation

Chapter 9: Blind Separation of Non-Negative Source Signals

In this chapter, we derive an algorithm, referred to as multiplicative non-negative independent component analysis (M-NICA), that is able to unmix independent non-negative source signals from a set of observed instantaneous mixtures of the original source signals. It serves as an enabling algorithm for the multi-speaker voice activity detector that is described in Chapter 10. Since the M-NICA is based on a multiplicative update rule, it has the facilitating property that it does not depend on a user-defined learning rate, as opposed to existing gradient based updates such as in the non-negative PCA (NPCA) algorithm. We provide batch mode and sliding-window simulations for different types of signals, and compare the performance of M-NICA with NPCA. It is observed that M-NICA generally yields a better unmixing accuracy, but converges slower than NPCA. Especially when the amount of data samples is small, M-NICA significantly outperforms NPCA, which makes it more useful for adaptive sliding-window implementations, as required in voice activity detection in dynamic environments (see Chapter 10).

Chapter 10: Energy-Based Multi-Speaker VAD

In this chapter, the goal is to track the individual speech power of multiple simultaneous speakers as observed by different microphones of a spatially distributed microphone array with unknown microphone positions. By thresholding the power of each speaker, we can easily create a voice activity detector for each speaker independently. The latter can be used for speaker extraction, e.g., by using LCMV beamforming techniques [51]. By considering the short-term power of the microphone signals, the problem can be converted into a non-negative blind source separation (NBSS) problem. We use the M-NICA algorithm described in Chapter 9 to solve the problem in an adaptive fashion. Since this is an energy-based method, the data rate is low and clock synchronization is not that crucial, which is a desirable property in WASNs. We provide simulation results that demonstrate the effectiveness of the presented algorithm, and we show that M-NICA outperforms NPCA in this specific application.

Chapter 11: Link Failure Response and Sensor Subset Selection

In distributed signal estimation problems with WSNs, energy saving is very important. In many cases, only a subset of the nodes observe useful signals that significantly contribute to the signal estimation. The other nodes can then be put to sleep to save energy. In this chapter, we provide a greedy sensor subset selection (SSS) algorithm for an MMSE signal estimation problem in a WSN with a fusion center. However, the technique can also be used in DANSE-type algorithms, where each node serves as a local fusion center. Besides the SSS, we also consider link failure response, i.e., to provide a quick update of the signal estimator in case one of the links suddenly breaks down. SSS and link failure response problems are related since they require knowledge of the new optimal estimator when sensors are removed or added, and both are very important in low-delay signal estimation with high sampling frequency. We derive formulas to efficiently compute the optimal fall-back estimator in case of a link failure, and we derive formulas to efficiently monitor the utility of each sensor signal. Simulation results demonstrate that a significant amount of energy can be saved at the cost of a slight decrease in estimation performance.

1.7 Chapters and Publications Overview

The following publications are included in this thesis:

Part II: Distributed Signal Estimation Techniques

Chapter 2: Fully Connected DANSE with Sequential Node-Updating
A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal es-

timation in fully connected sensor networks – Part I: sequential node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5277 - 5291, 2010.

Chapter 3: Fully Connected DANSE with Simultaneous Node-Updating
A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – Part II: simultaneous & asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5292 - 5306, 2010.

Chapter 4: Robust DANSE for Speech Enhancement
A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435

Chapter 5: DANSE in Networks with Tree Topology
A. Bertrand and M. Moonen, "Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2196-2210, 2011.

Chapter 8: Linearly-Constrained DANSE
A. Bertrand and M. Moonen, "Distributed node-specific LCMV beamforming in wireless sensor networks," Submitted for publication, 2011.

Part III: Distributed Parameter Estimation Techniques

Chapter 7: Distributed Total Least Squares
A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2320 - 2330, 2011.

Chapter 8: Diffusion Bias-Compensated RLS Estimation
A. Bertrand, M. Moonen and A. H. Sayed, "Diffusion bias-compensated RLS estimation over adaptive networks," Submitted for publication, 2011.

Part IV: Supporting Techniques for Signal Estimation

Chapter 9: Blind Separation of Non-Negative Source Signals
A. Bertrand and M. Moonen, "Blind separation of non-negative source signals using multiplicative updates and subspace projection," *Signal Processing*, vol. 90, no. 10, pp. 2877 - 2890, 2010.

Chapter 10: Energy-Based Multi-Speaker VAD
A. Bertrand and M. Moonen, "Energy-based multi-speaker voice activity

detection with an ad hoc microphone array," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, Texas USA, March 2010 , pp. 85 - 88.

Chapter 11: Link Failure Response and Sensor Subset Selection
A. Bertrand and M. Moonen, "Efficient sensor subset selection and link failure response for linear MMSE signal estimation in wireless sensor networks," *Proc. European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, August 2010, pp. 1092 - 1096.

Bibliography

[1] "Special issue on compressive sensing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, 2008.

[2] S. Haykin and K. J. Ray Liu, *Handbook on Array Processing and Sensor Networks*. Hoboken, NJ: Wiley-IEEE Press, 2010.

[3] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2033–2036, 2001.

[4] A. Swami, Q. Zhao, Y.-W. Hong, and L. Tong, *Wireless Sensor Networks: Signal Processing and Communications*. West Sussex, England: Wiley, 2007.

[5] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, pp. 51–58, May 2000.

[6] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124 –138, 2011.

[7] K. Zhang, X. Li, P. Zhang, and H. Li, "Optimal linear estimation fusion - part VI: sensor data compression," *Proc. Sixth International Conference of Information Fusion*, pp. 221–228, 2003.

[8] Z.-Q. Luo, G. Giannakis, and S. Zhang, "Optimal linear decentralized estimation in a bandwidth constrained sensor network," *Proc. International Symposium on Information Theory (ISIT)*, pp. 1441–1445, Sept. 2005.

[9] I. Schizas, G. Giannakis, and Z.-Q. Luo, "Distributed estimation using reduced-dimensionality sensor observations," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4284–4299, Aug. 2007.

[10] Y. Zhu, E. Song, J. Zhou, and Z. You, "Optimal dimensionality reduction of sensor data in multisensor estimation fusion," *IEEE Transactions on Signal Processing*, vol. 53, no. 5, pp. 1631–1639, May 2005.

[11] J. Zhou, Y. Zhu, Z. You, and E. Song, "An efficient algorithm for optimal linear estimation fusion in distributed multisensor systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 36, no. 5, pp. 1000–1009, Sept. 2006.

[12] J. Fang and H. Li, "Optimal/near-optimal dimensionality reduction for distributed estimation in homogeneous and certain inhomogeneous scenarios," *IEEE Transactions on Signal Processing*, vol. 58, no. 8, pp. 4339 –4353, 2010.

[13] 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, Aug. 2007.

[14] L. Li, J. Chambers, C. G. Lopes, and A. H. Sayed, "Distributed estimation over an adaptive incremental network based on the affine projection algorithm," *IEEE Transactions on Signal Processing*, vol. 58, no. 1, pp. 151 –164, 2010.

[15] Y.-W. Hong and A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 5, pp. 1085 – 1099, May 2005.

[16] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. Strogatz, "Distributed synchronization in wireless networks," *IEEE Signal Processing Magazine*, vol. 25, no. 5, pp. 81 –97, 2008.

[17] S. Wehr, I. Kozintsev, R. Lienhart, and W. Kellermann, "Synchronization of acoustic sensors for distributed ad-hoc audio networks and its use for blind source separation," in *Proc. IEEE International Symposium on Multimedia Software Engineering (ISMSE)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 18–25.

[18] A. Speranzon, C. Fischione, K. Johansson, and A. Sangiovanni-Vincentelli, "A distributed minimum variance estimator for sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 609 –621, May 2008.

[19] B. Van Veen and K. Buckley, "Beamforming: a versatile approach to spatial filtering," *IEEE ASSP Magazine*, vol. 5, no. 2, pp. 4 –24, apr. 1988.

[20] 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, July 2008.

[21] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[22] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1035–1048, March 2010.

[23] A. H. Sayed and C. G. Lopes, "Distributed processing over adaptive networks," in *International Symposium on Signal Processing and Its Applications (ISSPA)*, 2007, pp. 1 –3.

[24] G. Mateos, I. Schizas, and G. Giannakis, "Closed-form MSE performance of the distributed LMS algorithm," in *13th IEEE Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop (DSP/SPE)*, 2009, pp. 66 –71.

[25] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 981030, 19 pages, 2009. doi:10.1155/2009/981030.

[26] ——, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, 2009.

[27] S. Kar and J. Moura, "Distributed linear parameter estimation in sensor networks: Convergence properties," in *Asilomar Conference on Signals, Systems and Computers*, Oct. 2008, pp. 1347 –1351.

[28] A. Dogandzic and B. Zhang, "Distributed estimation and detection for sensor networks using hidden Markov random field models," *IEEE Transactions on Signal Processing*, vol. 54, no. 8, pp. 3200 –3215, 2006.

[29] J. Fang and H. Li, "Distributed estimation of Gauss-Markov random fields with one-bit quantized data," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 449 –452, May 2010.

[30] P. Julian, A. Andreou, L. Riddle, S. Shamma, D. Goldberg, and G. Cauwenberghs, "A comparative study of sound localization algorithms for energy aware sensor network nodes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 4, pp. 640 – 648, 2004.

[31] X. Sheng and Y.-H. Hu, "Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 53, no. 1, pp. 44 – 53, 2005.

[32] M. Chen, Z. Liu, L.-W. He, P. Chou, and Z. Zhang, "Energy-based position estimation of microphones and speakers for ad hoc microphone arrays," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2007, pp. 22–25.

[33] P. Bergamo, S. Asgari, H. Wang, D. Maniezzo, L. Yip, R. Hudson, K. Yao, and D. Estrin, "Collaborative sensor networking towards real-time acoustic beamforming in free-space and limited reverberance," *IEEE Transactions on Mobile Computing*, vol. 3, pp. 211–224, Jul.-Sep. 2004.

[34] Y. Jia, Y. Luo, Y. Lin, and I. Kozintsev, "Distributed microphone arrays for digital home and office," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2006, pp. 1065–1068.

[35] J. Freudenberger, S. Stenzel, and B. Venditti, "Spectral combining for microphone diversity systems," *Proc. European signal processing conference (EUSIPCO), Glasgow - Scotland*, pp. 854–858, August 2009.

[36] A. Boothroyd, "Hearing aid accessories for adults: The remote FM microphone," *Ear and Hearing*, vol. 25, pp. 22–33, 2004.

[37] D. Fabry, H. Mälder, and E. Dijkstra, "Acceptance of the wireless microphone as a hearing aid accessory for adults," *Hearing Journal*, vol. 60, pp. 32–36, Nov. 2007.

[38] V. Berisha, H. Kwon, and S. Spanias, "Real-time implementation of a distributed voice activity detector," in *Fourth IEEE Workshop on Sensor Array and Multichannel Processing*, July 2006, pp. 659 –662.

[39] E. Robledo-Arnuncio, T. S. Wada, and B.-H. Juang, "On dealing with sampling rate mismatches in blind source separation and acoustic echo cancellation," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2007, pp. 34 –37.

[40] S. Doclo, T. van den Bogaert, M. Moonen, 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, Jan. 2009.

[41] E. Aarts and S. Marzano, *The New Everyday: Views on Ambient Intelligence*. 010 Publishers, 2003.

[42] S. Doclo, *Multi-microphone noise reduction and dereverberation techniques for speech applications*. PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), May 2003.

[43] 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, Oct. 1987.

[44] O. Hoshuyama, A. Sugiyama, and A. Hirano, "A robust adaptive beamformer for microphone arrays with a blocking matrix using constrained adaptive filters," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2677 –2684, Oct. 1999.

[45] P. Oak and W. Kellermann, "A calibration algorithm for robust generalized sidelobe cancelling beamformers," in *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC), Eindhoven, The Netherlands*, 2005.

[46] D. Van Compernolle, "Switching adaptive filters for enhancing noisy and reverberant speech from microphone array recordings," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Albuquerque, NM, Apr. 1990, pp. 833 –836 vol.2.

[47] M. Buck, T. Haulick, and H.-J. Pfleiderer, "Self-calibrating microphone arrays for speech signal acquisition: A systematic approach," *Signal Processing*, vol. 86, no. 6, pp. 1230 – 1238, 2006.

[48] 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, Sept. 2002.

[49] S. Gannot, D. Burshtein, and E. Weinstein, "Signal enhancement using beamforming and nonstationarity with applications to speech," *IEEE Transactions on Signal Processing*, vol. 49, no. 8, pp. 1614 –1626, Aug. 2001.

[50] S. Markovich Golan, S. Gannot, and I. Cohen, "Subspace tracking of multiple sources and its application to speakers extraction," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, Texas USA, Mar. 2010, pp. 201 –204.

[51] 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, pp. 1071–1086, August 2009.

[52] H. Buchner, R. Aichner, and W. Kellermann, "Blind source separation for convolutive mixtures exploiting nongaussianity, nonwhiteness, and nonstationarity," in *Proc. International Workshop on Acoustic Echo and Noise Control (IWAENC)*, Kyoto, Japan, Sept. 2003, pp. 223–226.

[53] S. Araki, S. Makino, R. Mukai, Y. Hinamoto, T. Nishikawa, and H. Saruwatari, "Equivalence between frequency domain blind source separation and frequency domain adaptive beamforming," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002.

[54] S. Tanyer and H. Ozer, "Voice activity detection in nonstationary noise," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 4, pp. 478 –482, July 2000.

[55] P. Ghosh, A. Tsiartas, and S. Narayanan, "Robust voice activity detection using long-term signal variability," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 3, pp. 600 –613, 2011.

[56] M. Fujimoto and K. Ishizuka, "Noise robust voice activity detection based on switching kalman filter," *IEICE - Transactions on Information and Systems*, vol. E91-D, pp. 467–477, March 2008.

[57] T. Klasen, T. Van den Bogaert, M. Moonen, and J. Wouters, "Binaural noise reduction algorithms for hearing aids that preserve interaural time delay cues," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1579–1585, April 2007.

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

[59] B. Cornelis, M. Moonen, and J. Wouters, "Performance analysis of multichannel Wiener filter based noise reduction in hearing aids under second order statistics estimation errors," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, 2011.

[60] M. Souden, J. Benesty, and S. Affes, "A study of the LCMV and MVDR noise reduction filters," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4925 –4935, 2010.

[61] E. Habets, J. Benesty, I. Cohen, S. Gannot, and J. Dmochowski, "New insights into the MVDR beamformer in room acoustics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 1, pp. 158 –170, 2010.

[62] P. B. S. Maraboina, D. Kolossa and R. Orglmeister, "Multi-speaker voice activity detection using ICA and beampattern analysis," in *Proc. European signal processing conference (EUSIPCO)*, Florence, Italy, 2006.

[63] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Prentice Hall, 1996.

[64] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.

[65] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs, CT: YBS Publishing, 1995.

[66] P. Yip and K. Rao, *The transform and data compression handbook*. Boca Raton, FL: CRC Press, 2001.

[67] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

[68] S. Srinivasan and A. C. Den Brinker, "Rate-constrained beamforming in binaural hearing aids," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 257197, 14 pages, 2009.

[69] S. Markovich Golan, S. Gannot, and I. Cohen, "A reduced bandwidth binaural MVDR beamformer," in *Proc. International Workshop on Acoustic Echo and Noise Control (IWAENC)*, Tel-Aviv, Israel, Aug. 2010.

[70] T. M. Cover and J. A. Thomas, *Elements of information theory*, 2nd ed. New York: Wiley-Interscience, 2006.

[71] S. Doclo, T. C. Lawin-Ore, and T. Rohdenburg, "Rate-constrained binaural MWF-based noise reduction algorithms," in *Proc. ITG Conference on Speech Communication*, Bochum, Germany, Oct. 2010.

[72] T. C. Lawin-Ore and S. Doclo, "Analysis of rate constraints for MWF-based noise reduction in acoustic sensor networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011.

[73] O. Roy and M. Vetterli, "Rate-constrained collaborative noise reduction for wireless hearing aids," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 645 –657, 2009.

[74] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471 – 480, July 1973.

[75] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1 – 10, Jan. 1976.

[76] S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, March 2003.

[77] J. Chou, D. Petrovic, and K. Ramachandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *Proc. Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, March 2003, pp. 1054–1062.

[78] M. Gastpar, P. Dragotti, and M. Vetterli, "The distributed Karhunen-Loève transform," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5177 –5196, 2006.

[79] B. Chen and P. C. Loizou, "A Laplacian-based MMSE estimator for speech enhancement," *Speech Communication*, vol. 49, pp. 134–143, February 2007.

[80] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004.

[81] D. Scherber and H. Papadopoulos, "Locally constructed algorithms for distributed computations in ad-hoc networks," in *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp. 11 – 19.

[82] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. International Symposium on Information Processing in Sensor Networks (ISPN)*, 2005, pp. 63 – 70.

[83] M. S. Talebi, M. Kefayati, B. H. Khalaj, and H. R. Rabiee, "Adaptive consensus averaging for information fusion over sensor," in *Proc. IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.

[84] P. Braca, S. Marano, and V. Matta, "Running consensus in wireless sensor networks," in *Proc. International Conference on Information Fusion*, July 2008, pp. 1 –6.

[85] 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.

[86] A. H. Sayed, *Adaptive Filters*. NJ: John Wiley & Sons, 2008.

[87] B. Widrow and S. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[88] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[89] G. Mateos, J. Bazerque, and G. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262 –5276, 2010.

[90] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM Journal on Optimization*, vol. 7, pp. 913–926, April 1997.

[91] A. Bertrand, J. Callebaut, and M. Moonen, "Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks," in *Proc. International Workshop on Acoustic Echo and Noise Control (IWAENC), Tel Aviv, Israel*, Aug. 2010.

Part II

Distributed Signal Estimation Techniques

Chapter 2

Fully Connected DANSE with Sequential Node Updating

Distributed adaptive node-specific signal estimation in fully connected sensor networks – Part I: sequential node updating

Alexander Bertrand and Marc Moonen

Published in *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5277 - 5291, Oct. 2010.

Contributions of first author

- literature study
- co-development of the DANSE$_K$ algorithm
- co-establishment of proof of convergence and optimality of DANSE$_K$
- design of simulation scenarios
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

We introduce a distributed adaptive algorithm for linear minimum mean squared error (MMSE) estimation of node-specific signals in a fully connected broadcasting sensor network where the nodes collect multi-channel sensor signal observations. We assume that the node-specific signals to be estimated share a common latent signal subspace with a dimension that is small compared to the number of available sensor channels at each node. In this case, the algorithm can significantly reduce the required communication bandwidth and still provide the same optimal linear MMSE estimators as the centralized case. Furthermore, the computational load at each node is smaller than in a centralized architecture in which all computations are performed in a single fusion center. We consider the case where nodes update their parameters in a sequential round robin fashion. Numerical simulations support the theoretical results. Because of its adaptive nature, the algorithm is suited for real-time signal estimation in dynamic environments, such as speech enhancement with acoustic sensor networks.

2.1 Introduction

In a sensor network [1] a general objective is to utilize all sensor signal observations available in the entire network to perform a certain task, such as the estimation of a parameter or signal. Gathering all observations in a fusion center to calculate an optimal estimate may however require a large communication bandwidth and computational power. This approach is often referred to as centralized fusion or estimation. An alternative is a distributed approach where each node has its own processing unit and the estimation relies on distributed processing and cooperation. This approach is preferred, especially so when it is scalable in terms of its communication bandwidth requirement and computational complexity.

In many sensor network estimation frameworks the sensor signal observations are used to estimate a common network-wide desired parameter or signal, denoted here by $\mathbf{d}$. This means that all nodes contribute to a common goal, i.e. the estimation of the globally defined variable $\mathbf{d}$, which is the same for all nodes (see for example [2–8]). This can be viewed as a special case of the more general problem, which is considered here, where each node in the network estimates a different node-specific desired signal, i.e. node k estimates the locally defined signal $\mathbf{d}_k$. This means that all nodes have a different local objective, which they pursue through cooperation with other nodes. We describe a distributed adaptive node-specific signal estimation (DANSE) algorithm that operates in an ideal fully connected network. The nodes broadcast compressed multi-channel sensor signal observations that can be captured by

all other nodes in the network, possibly with the help of relay nodes. The computational load is distributed over the different nodes in the network.

The DANSE algorithm is designed for the case where the node-specific desired signals share a common (unknown) latent signal subspace. If this signal space has a small dimension compared to the number of available sensor channels at each node, the DANSE algorithm exploits this common interest of the nodes to significantly compress the data to be broadcast, and yet converge to the optimal linear minimum mean squared error (MMSE) estimators as if all sensor signal observations were available at each node. Although the DANSE algorithm implicitly assumes a specific structure in the relationship between the desired signals of the different nodes, it is noted that the actual parameters of these latent dependencies are not assumed to be known, i.e. nodes do not know how their desired signal is related to the desired signals of other nodes. The model that is assumed in the DANSE algorithm naturally emerges in adaptive signal estimation problems in dynamic scenarios where the target signal statistics and the transfer functions to the sensors are not known and may change during operation of the algorithm. Therefore, the original target signal cannot be recovered, and so an option is then to let the nodes optimally estimate the signal as it is observed locally by the node's sensors. In this case, the desired signals of the different nodes are differently filtered versions of the same target signal, i.e. they share a common latent signal subspace.

Because of its adaptive nature, the DANSE algorithm is suited for real-time applications in dynamic environments. Typical applications are vibration monitoring, wireless acoustic sensor networks (for surveillance, video conferencing, domotics, audio recording...), and noise reduction in hearing aids with external sensor nodes and/or cooperation between multiple hearing aids [9, 10]. Node-specific estimation is particularly important in applications where a target signal needs to be estimated as it is observed at a specific local position. For instance, in acoustic surveillance, it is often required to be able to locate a sound source, so spatial information in the observations of different nodes must be retained in the estimation process. In cooperating hearing aids, it is important to estimate the signal as it impinges at the hearing aid itself, to preserve the auditory cues for directional hearing [11, 12].

The DANSE algorithm is based on linear compression of multi-channel sensor signal observations. Linear compression of sensor signal observations for data fusion has been the topic of earlier work, e.g. [5–8]. The presented techniques, however, assume prior knowledge of the intra- and inter-sensor (cross-)correlation structure in the entire network. This must be obtained by a priori training using all uncompressed sensor signal observations, or must be derived from a specific data model. Such assumptions make it difficult to apply the resulting algorithms in adaptive networks or dynamic environments where the statistics of the desired signals or sensor signals may change. The DANSE algorithm can adapt to these changes because nodes estimate and re-estimate

all required statistical quantities on the compressed data during operation. For this, we assume that each node can adaptively estimate the cross correlation between its local sensor signals and its desired signal. It is noted that the acquisition of these signal statistics is often difficult or impossible, since the target signal is assumed to be unknown. However, we will explain that in particular cases, it is possible to estimate the required statistics, e.g. when the target signal has an on-off behavior (such as speech signals), or when the target source periodically transmits a priori known training sequences. In cases where the local statistics cannot be estimated adaptively, the DANSE algorithm can still be used in a semi-adaptive context, i.e. scenarios with static noise statistics but with changing target signal statistics or vice versa, assuming that the static correlation structure is a priori known.

In [13], a batch-mode description of the DANSE algorithm was briefly introduced. In this paper, we provide more details, i.e. we include a convergence proof and introduce a truly adaptive version. In addition, we address implementation aspects, and provide extensive simulation results, both in batch mode and in a dynamic scenario. We only consider the case where nodes update their parameters in a sequential round robin fashion. The case where nodes update simultaneously or asynchronously is treated in a companion paper [14]. In [10], a pruned version of the DANSE algorithm has been used for microphone-array based speech enhancement in binaural hearing aids, where it was referred to as distributed multi-channel Wiener filtering. In this application, two hearing aids in a binaural configuration exchange a linear combination of their microphone signals to estimate the target sound that is recorded by their reference microphone. Convergence of the 2-node system has been proven for the special case where there is a single target speaker. The more general DANSE algorithm provided in this paper allows for a non-trivial extension to a scenario with multiple target speakers and a network with more than 2 nodes. Using extra acoustic sensor nodes that communicate with the hearing aids generally improves the noise reduction performance, since the acoustic sensors physically cover a larger area [9].

The paper is organized as follows. The problem formulation and notation are presented in Section 2.2. In Section 2.3, we first address the simple case in which the node-specific desired signals are scaled versions of each other and we prove convergence of the DANSE algorithm to the optimal linear MMSE estimators when nodes update their parameters sequentially. In Section 2.4, this algorithm is generalized to the case in which the node-specific desired signals share a common latent Q-dimensional signal subspace. In Section 2.5, we address some implementation details of DANSE and we study the complexity of the algorithm. Finally, Section 2.6 illustrates the convergence results with numerical simulations. Conclusions are given in Section 2.7.

2.2 Problem Formulation and Notation

2.2.1 Node-Specific Linear MMSE Estimation

We consider an ideal fully connected network with sensor nodes $\{1, \ldots, J\} = \mathcal{J}$, in which data broadcast by a node can be captured by all other $(J-1)$ nodes in the network through an ideal link. Node k collects observations of a complex[1] valued M_k-channel signal $\mathbf{y}_k[t]$, where $t \in \mathbb{N}$ is the discrete time index, and where $\mathbf{y}_k[t]$ is an M_k-dimensional column vector. Each channel $y_{kn}[t]$, $\forall\, n \in \{1, \ldots, M_k\}$, of the signal $\mathbf{y}_k[t]$ corresponds to a sensor signal to which node k has access. We assume that all signals are stationary and ergodic. In practice, the stationarity and ergodicity assumption can be relaxed to short-term stationarity and ergodicity, in which case the theory should be applied to finite signal segments that are assumed to be stationary and ergodic. For the sake of an easy exposition, we will omit the time index when referring to a signal, and we will only write the time index when referring to one specific observation, i.e. $\mathbf{y}_k[t]$ is the observation of the signal $\mathbf{y}_k$ at time t. We define $\mathbf{y}$ as the M-channel signal in which all $\mathbf{y}_k$ are stacked, where $M = \sum_{k=1}^{J} M_k$. This scenario is described in Fig. 2.1.

It is noted that this problem formulation also allows for hierarchical network architectures, in which the sensors are grouped in J clusters. The sensors of a specific cluster then transmit their observations to a nearby fusion center, i.e. a 'higher level' node. The J fusion centers then correspond to the J nodes in the above framework, and the collected observations in sensor cluster k correspond to the M_k-channel signals $\mathbf{y}_k$ as explained above. Fig. 2.2 shows such a scenario for a network with 3 fusion centers ($J = 3$).

We first consider the centralized estimation problem, i.e. we assume that each node has access to the observations of the entire M-channel signal $\mathbf{y}$. This corresponds to the case where nodes broadcast their uncompressed observations to all other nodes. In Sections 2.3 and 2.4, the general goal will be to compress the broadcast signals, while preserving the estimation performance of this centralized estimator. The objective for node k is to estimate a complex valued node-specific signal d_k, referred to as the desired signal, from the observations of $\mathbf{y}$. We consider the general case where d_k is not an observed signal, i.e. it is assumed to be unknown, as it is the case in signal enhancement (e.g. in speech enhancement, d_k is the speech component in a noisy microphone signal). Node k uses a linear estimator $\mathbf{w}_k$ to estimate d_k as $\overline{d}_k = \mathbf{w}_k^H \mathbf{y}$ where $\mathbf{w}_k$ is a complex valued M-dimensional vector, and where superscript H denotes the conjugate transpose operator. We assume that the M-channel signal $\mathbf{y}$ is correlated to the node-specific desired signals, but unlike [6, 8], we do not restrict ourselves to any data model generating the sensor signals, nor do we

[1] Throughout this paper, all signals are assumed to be complex valued to permit frequency-domain descriptions.

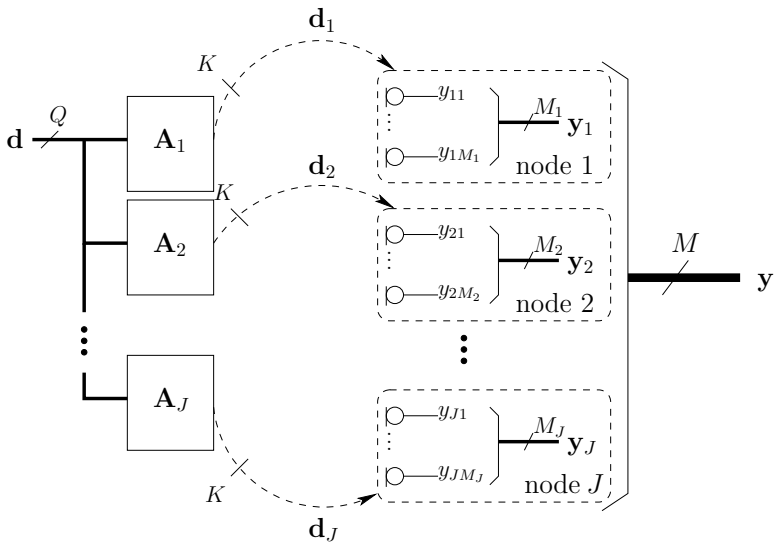


Figure 2.1: Description of the scenario. The network contains J sensor nodes, $k = 1 \dots J$, where node k collects M_k-channel sensor signal observations and estimates a node-specific desired signal $\mathbf{d}_k$, which is a mixture of the Q channels of a common latent signal $\mathbf{d}$.

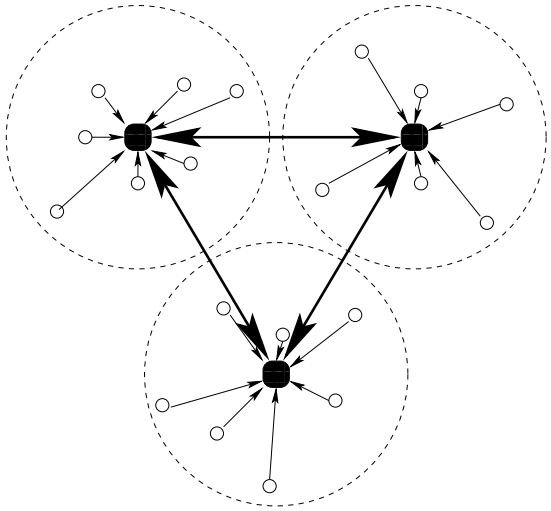


Figure 2.2: A hierarchical architecture with 3 fusion centers ($J = 3$), each one collecting sensor signals from nearby sensors.

make any assumptions on the probability distributions of the involved signals. We consider linear MMSE estimation based on a node-specific estimator $\hat{\mathbf{w}}_k$, i.e.

$$\hat{\mathbf{w}}_k = \arg\min_{\mathbf{w}_k} E\{|d_k - \mathbf{w}_k^H \mathbf{y}|^2\} \tag{2.1}$$

with $E\{.\}$ the expected value operator. Assuming that the correlation matrix $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$ has full rank[2], the unique solution of (2.1) is [15]:

$$\hat{\mathbf{w}}_k = \mathbf{R}_{yy}^{-1}\mathbf{r}_{yd_k} \tag{2.2}$$

with $\mathbf{r}_{yd_k} = E\{\mathbf{y}d_k^*\}$, where d_k^* denotes the complex conjugate of d_k. Based on the assumption that the signals are ergodic, $\mathbf{R}_{yy}$ and $\mathbf{r}_{yd_k}$ can be estimated by time averaging. The $\mathbf{R}_{yy}$ is directly estimated from the sensor signal observations. Since d_k is assumed to be unknown, the estimation of the correlation vector $\mathbf{r}_{yd_k}$ has to be done indirectly, based on specific strategies, e.g. by exploiting the on-off behavior of the target signal (e.g. for speech enhancement [9, 10]), by using training sequences, or by using partial prior knowledge when the estimation is performed in a semi-adaptive context. We will provide more details on these strategies in Section 2.5.1. In the sequel, we assume that $\mathbf{r}_{yd_k}$ can be estimated during operation of the algorithm.

In the above estimation procedure, temporal correlation appears to be ignored. However, differently delayed versions of one or more sensor signals at node k can be added to the channels of $\mathbf{y}_k$, to also exploit the temporal information in the signals. For example, assume that node k has access to 4 sensor signals. Then each of these signals is delayed with 1, up to $N-1$ sample delays, resulting in $N-1$ extra (delayed) channels. In this case, the dimension of $\mathbf{y}_k$ is $M_k = 4N$.

It is noted that our problem statement differs from [2–4], where each node collects different spatio-temporal observations of two correlated signals y and d. The objective is then to find the best common linear fit between these observations, with a single set of coefficients $\mathbf{w}$, which is assumed to be the same for each node. Since the coefficients in $\mathbf{w}$ are of interest, only the locally estimated $\mathbf{w}$'s must be shared between nodes, whereas the sensor observations themselves are only used locally to update the estimate of $\mathbf{w}$. Since all nodes are assumed to estimate the same set of coefficients, incremental or diffusive averaging strategies can be used.

2.2.2 Common Latent Signal Subspace

In our problem statement, each node k only collects observations of $\mathbf{y}_k$ which corresponds to a subset of the channels of the full signal $\mathbf{y}$. To find the optimal

[2]This assumption is mostly satisfied in practice because of a noise component at every sensor that is independent of other sensors, e.g. thermal noise. If not, pseudo-inverses should be used. A further comment on the rank-deficient case is made in Section 2.4.3.

MMSE solution (2.2), each node k therefore in principle has to broadcast its observations of $\mathbf{y}_k$ to all other nodes in the network, which requires a large communication bandwidth. One possibility to reduce the required bandwidth is to broadcast only a few linear combinations of the components of the $\mathbf{y}_k$ observations instead of all M_k components. Finding the optimal linear compression is often a non-trivial task, and in general this will not lead to the optimal solutions (2.2). In many practical cases, however, the d_k signals share a common latent signal subspace, and then this can be exploited in the compression. The most simple case is when all $d_k = d$, i.e. the desired signal is the same for all nodes. We will first handle the slightly more general case where all d_k are scaled versions of a common latent single-channel signal d. For this scenario, we will introduce the DANSE$_1$ algorithm, in which the data to be broadcast by each node k is compressed by a factor M_k. Despite this compression, the algorithm converges to the optimal node-specific solution (2.2) at every node as if no compression were used for the broadcasts.

This scenario can then be extended to the more general case where the desired signals share a common Q-dimensional signal subspace, i.e.

$$\forall\, k \in \mathcal{J}:\ d_k = \mathbf{a}_k^H \mathbf{d} \tag{2.3}$$

with $\mathbf{a}_k$ defining an unknown Q-dimensional complex vector, and $\mathbf{d}$ a latent complex valued Q-channel signal defining the Q-dimensional signal subspace that contains all d_k signals. This model applies to situations where the desired signal is generated by multiple latent processes simultaneously (e.g. measuring vibrations when there are multiple exciters, or recording a conversation between multiple speakers [9]). Since the statistics of the latent signals as well as the propagation properties to the different sensors are generally unknown, the signal estimation procedure can only use statistics that can be obtained from the local sensor signal observations. The desired signal of each node is then the linear mixture of the latent target signals as locally observed by a reference sensor.

In the sequel, we consider the general case where node k estimates a K-channel desired signal $\mathbf{d}_k$

$$\forall\, k \in \mathcal{J}:\ \mathbf{d}_k = \mathbf{A}_k \mathbf{d} \tag{2.4}$$

with $\mathbf{A}_k$ a $K \times Q$ complex valued matrix. This data model is depicted in Fig. 2.1. It is noted that the matrix $\mathbf{A}_k$ and the latent signal $\mathbf{d}$ are assumed to be unknown, i.e. nodes do not know how their node-specific desired signals $\mathbf{d}_k$ are related to each other. Since we also consider complex valued signals, (2.4) can correspond to a frequency domain description of a convolutive mixture in the time domain, as in [9, 10]. Expression (2.4) then defines a different estimation problem for each specific frequency. This yields frequency dependent estimators $\mathbf{w}_k(f)$, which translate to multi-tap filters in the time domain.

Notice that, if $K \geq Q$, the desired signal $\mathbf{d}_k$ spans the complete signal subspace

defined by the Q-channel signal $\mathbf{d}$ (provided that the $K \times Q$ matrix $\mathbf{A}_k$ has full rank). If this holds for each node in the network, we will show that the data to be broadcast by node k can be compressed by a factor $\frac{M_k}{Q}$. This means that node k only needs to broadcast Q linear combinations of the components of its observations of $\mathbf{y}_k$, while the optimal node-specific solution (2.2) is still obtained at all nodes. Notice that in practical applications, the actual signal(s) of interest can be a subset of the entries in $\mathbf{d}_k$, in which case the other entries should be seen as auxiliary channels to capture the latent Q-dimensional signal subspace that contains the $\mathbf{d}_k$'s. For instance, consider the case where nodes estimate the target signal as observed by their reference sensor, i.e. node k estimates the node-specific desired signal d_k as in (2.3). Node k then selects $K-1$ extra auxiliary reference sensors, and also estimates the target signal as it arrives on these sensors. The resulting K-channel desired signal $\mathbf{d}_k$ then spans the complete signal subspace if $K \geq Q$.

2.3 DANSE with Single-Channel Broadcast Signals (K=1)

The algorithm introduced in this paper is an iterative scheme referred to as distributed adaptive node-specific signal estimation (DANSE), since its objective is to estimate a node-specific signal at each node in a distributed fashion. In the general scheme, each node k broadcasts $\min\{K, M_k\}$-component compressed sensor signal observations. We will refer to this as DANSE$_K$, where the subscript K refers to the number of channels of the broadcast signals. For the sake of an easy exposition, we first introduce the DANSE algorithm for the simple case where $K = 1$ and we will show that DANSE$_1$ converges to the optimal filters if $Q = 1$, i.e. if the single-channel desired signals d_k are non-zero scaled versions of the same latent single-channel signal d. In Section 2.4 we generalize this to the more general DANSE$_K$ algorithm, and we will show that this algorithm converges to the optimal filters if $Q = K$ and if all $\mathbf{A}_k$ in (2.4) have rank K.

2.3.1 DANSE$_1$ Algorithm

The goal for each node k is to estimate the signal d_k with a linear estimator that uses all observations in the entire network, i.e. $\overline{d}_k = \mathbf{w}_k^H \mathbf{y}$. We aim to obtain the MMSE solutions (2.2), without the need for each node to broadcast all M_k components of the $\mathbf{y}_k$ observations. For this, we define a partitioning of the estimator $\mathbf{w}_k$ as $\mathbf{w}_k = [\mathbf{w}_{k1}^T \ \dots \ \mathbf{w}_{kJ}^T]^T$ with $\mathbf{w}_{kq}$ denoting the M_k-dimensional subvector of $\mathbf{w}_k$ that is applied to $\mathbf{y}_q$, and with superscript T denoting the

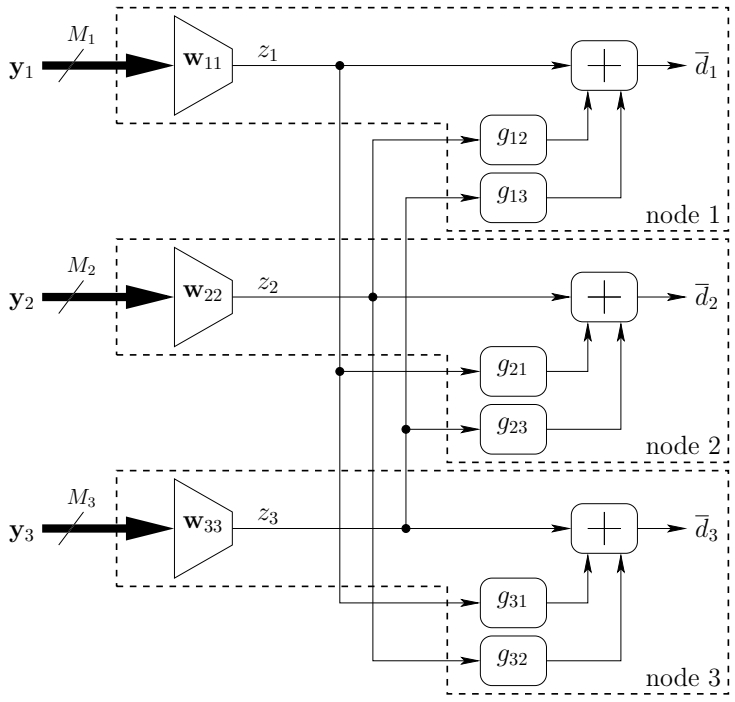


Figure 2.3: The DANSE$_1$ scheme with 3 nodes ($J = 3$). Each node k estimates a signal d_k using its own M_k-channel sensor signal observations, and 2 single-channel signals broadcast by the other two nodes.

transpose operator. In this way, (2.1) is equivalent to

$$\hat{\mathbf{w}}_k = \begin{bmatrix} \hat{\mathbf{w}}_{k1} \\ \vdots \\ \hat{\mathbf{w}}_{kJ} \end{bmatrix} = \underset{\{\mathbf{w}_{k1},\ldots,\mathbf{w}_{kJ}\}}{\arg\min} \; E\{|d_k - \sum_{q=1}^{J} \mathbf{w}_{kq}^H \mathbf{y}_q|^2\} \; . \tag{2.5}$$

Since node k only has access to the sensor signal observations of $\mathbf{y}_k$, it can only control a specific part of the estimator $\mathbf{w}_k$, namely $\mathbf{w}_{kk}$. In the DANSE$_1$ algorithm, each node k broadcasts the output of this partial estimator, i.e. observations of the compressed signal $z_k = \mathbf{w}_{kk}^H \mathbf{y}_k$. This reduces the data to be broadcast by a factor M_k. It is noted that $\mathbf{w}_{kk}$ acts both as a compressor and as a part of the estimator $\mathbf{w}_k$, i.e. the observations of the compressed signal z_k that is broadcast by node k is also used in the estimation of d_k at node k itself.

A node k now has access to $M_k + J - 1$ input channels, i.e. its own M_k sensor signals and $(J - 1)$ signals that it receives from the other nodes. Node k will compute the optimal linear combiner of these $M_k + J - 1$ input channels to

estimate d_k. The coefficient that is applied to the signal observations of z_q at node k is denoted by g_{kq}. A schematic illustration of this scheme (for J=3) is shown in Fig. 2.3. Notice that there is no decompression involved, i.e. node k does not expand the observations of the z_q signal, but only scales these with a scaling factor g_{kq}. As visualised in Fig. 2.3, the parametrization of the $\mathbf{w}_k$ now effectively applied at node k is therefore

$$\widetilde{\mathbf{w}}_k = \begin{bmatrix} g_{k1}\mathbf{w}_{11} \\ g_{k2}\mathbf{w}_{22} \\ \vdots \\ g_{kJ}\mathbf{w}_{JJ} \end{bmatrix} \tag{2.6}$$

i.e. each $\mathbf{w}_k$ is now defined by the set of $\mathbf{w}_{qq}$'s ($q \in \mathcal{J}$) together with a vector $\mathbf{g}_k = [g_{k1} \ \dots \ g_{kJ}]^T$, defining the scaling parameters. We use a tilde to indicate that the estimator is parametrized according to (2.6), which defines a solution space for $(\mathbf{w}_1, \dots, \mathbf{w}_J)$ with a specific structure. In this parametrization, node k can only manipulate the parameters $\mathbf{w}_{kk}$ and $\mathbf{g}_k$. In the sequel, we set $g_{kk} = 1$ to remove the ambiguity in $g_{kk}\mathbf{w}_{kk}$ (hence g_{kk} is omitted in Fig. 2.3). Notice that the solution space $(\widetilde{\mathbf{w}}_1, \dots, \widetilde{\mathbf{w}}_J)$ of DANSE$_1$ is $(M + J(J-1))$-dimensional, which is smaller[3] than the original MJ-dimensional solution space $(\mathbf{w}_1, \dots, \mathbf{w}_J)$ corresponding to the centralized algorithm, i.e. the solution space of the optimization problem (2.1). Still, the goal of the DANSE$_1$ algorithm is to iteratively update the parameters of (2.6) until $(\widetilde{\mathbf{w}}_1, \dots, \widetilde{\mathbf{w}}_J) = (\hat{\mathbf{w}}_1, \dots, \hat{\mathbf{w}}_J)$.

In the sequel, we will use the following notation and definitions. In general, we will use X^i to denote X at iteration i, where X can be a signal or a parameter. The J-channel signal $\mathbf{z}$ is defined as $\mathbf{z} = [z_1 \ \dots \ z_J]^T$. We define $\mathbf{g}_{k-q}$ as the vector $\mathbf{g}_k$ with entry g_{kq} omitted. Similarly, we define $\mathbf{z}_{-k}$ as the vector $\mathbf{z}$ with entry z_k omitted.

At every iteration i in the DANSE$_1$ algorithm, one specific node k will update its local parameters $\mathbf{w}_{kk}^i$ and $\mathbf{g}_{k-k}^i$, by solving its local node-specific MMSE problem with respect to its input signals, consisting of its own sensor signal observations $\mathbf{y}_k$ and the compressed signal observations of $\mathbf{z}_{-k}^i$, i.e. it solves

$$\begin{bmatrix} \mathbf{w}_{kk}^{i+1} \\ \mathbf{g}_{k-k}^{i+1} \end{bmatrix} = \underset{\mathbf{w}_{kk}, \mathbf{g}_{k-k}}{\arg\min} \ E\left\{ \left| d_k - \begin{bmatrix} \mathbf{w}_{kk}^H & \mathbf{g}_{k-k}^H \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix} \right|^2 \right\} . \tag{2.7}$$

Let $\widetilde{\mathbf{y}}_k^i$ denote the stacked version of the local input signals at node k, i.e.

$$\widetilde{\mathbf{y}}_k^i = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix} . \tag{2.8}$$

[3]It is assumed here that $J < M$, i.e. $M_k \geq 1$, $\forall\, k \in \mathcal{J}$, and there is at least one node k for which $M_k > 1$.

Then the solution of (2.7) is

$$\begin{bmatrix} \cdot & \mathbf{w}_{kk}^{i+1} & \cdot \\ & \mathbf{g}_{k-k}^{i+1} & \end{bmatrix} = \left(\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^{i}\right)^{-1} \mathbf{r}_{\tilde{y}_k d_k}^{i} \tag{2.9}$$

with

$$\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^{i} = E\{\tilde{\mathbf{y}}_k^i \tilde{\mathbf{y}}_k^{i\,H}\} \tag{2.10}$$

$$\mathbf{r}_{\tilde{y}_k d_k}^{i} = E\{\tilde{\mathbf{y}}_k^i d_k^*\}\ . \tag{2.11}$$

Since there is no decompression involved, the local estimation problems (2.7) have a smaller dimension than the original network-wide estimation problems (2.1), $\forall\, k \in \mathcal{J}$, i.e. the matrix $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^{i}$ is smaller than the $\mathbf{R}_{yy}$ matrix in (2.2).

We define a block size B which denotes the number of observations that the nodes collect in between two successive node updates, i.e. in between two increments of i. The DANSE$_1$ algorithm is described in Table 2.1 on the next page.

Remark I: Notice that the different iterations are spread out over time. Therefore, iterative characteristics of the algorithm do not have an impact on the amount of data that is transmitted, i.e. each sample is only broadcast once since the time index in (2.12) and (2.14) shifts together with the iteration index.

Remark II: In the above algorithm description, it is not mentioned how the correlation matrix $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^{i}$ and the correlation vector $\mathbf{r}_{\tilde{y}_k d_k}^{i}$ should be estimated. This estimation process depends on the application and the signals involved. In Section 2.5.1, we will suggest some possible strategies to estimate $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^{i}$ and $\mathbf{r}_{\tilde{y}_k d_k}^{i}$.

Remark III: It is noted that, when a node k updates its node-specific parameters $\mathbf{w}_{kk}^i$ and $\mathbf{g}_{k-k}^i$, the signal statistics of z_k change, i.e. z_k^i changes to z_k^{i+1}. Therefore, the next node to perform an update needs a sufficient number of observations of z_k to reliably estimate the correlation coefficients involving this signal. Therefore, the block-length B should be chosen large enough.

The DANSE$_1$ Algorithm

1. Initialize: $i \leftarrow 0$, $u \leftarrow 1$
   Initialize $\mathbf{w}_{kk}^0$ and $\mathbf{g}_{k-k}^0$ with random vectors, $\forall\, k \in \mathcal{J}$
2. Each node $k \in \mathcal{J}$ performs the following operation cycle:
   - Collect the sensor observations $\mathbf{y}_k[iB+n]$, $n = 0 \dots B-1$.
   - Compress these M_k-dimensional observations to

$$z_k^i[iB+n] = \mathbf{w}_{kk}^{i\,H}\mathbf{y}_k[iB+n],\ \ n = 0 \dots B-1\ . \tag{2.12}$$

   - Broadcast the compressed observations $z_k^i[iB+n]$, $n = 0 \dots B-1$, to the other nodes.
   - Collect the $(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \dots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   - Update the estimates of $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$ and $\mathbf{r}_{\tilde{y}_k d_k}^i$, by including the newly collected data.
   - Update the node-specific parameters:

$$\begin{bmatrix} \mathbf{w}_{kk}^{i+1} \\ \mathbf{g}_{k-k}^{i+1} \end{bmatrix} = \begin{cases} \left(\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i\right)^{-1}\mathbf{r}_{\tilde{y}_k d_k}^i & \text{if } k = u \\ \begin{bmatrix} \mathbf{w}_{k\underline{k}}^i \\ \mathbf{g}_{k-k}^i \end{bmatrix} & \text{if } k \neq u \end{cases} \tag{2.13}$$

   - Compute the estimate of $d_k[iB+n]$, $n = 0 \dots B-1$, as

$$\begin{aligned} \overline{d}_k[iB+n] =& \mathbf{w}_{kk}^{i+1\,H}\mathbf{y}_k[iB+n] \\ &+ \mathbf{g}_{k-k}^{i+1\,H}\mathbf{z}_{-k}^i[iB+n]\ . \end{aligned} \tag{2.14}$$

3. $i \leftarrow i+1$
4. $u \leftarrow (u \mod J)+1$
5. return to step 2

Table 2.1: The DANSE$_1$ algorithm

2.3.2 Convergence and Optimality of DANSE$_1$ if $Q = 1$ and Non-Zero Desired Signals

We now assume that all d_k are a non-zero scaled version of the same signal d, i.e. $d_k = \rho_k d$, with ρ_k a non-zero complex scalar but unknown to the individual nodes. Formula (2.2) shows that in this case, all $\hat{\mathbf{w}}_k$ are parallel, i.e.

$$\hat{\mathbf{w}}_k = \rho_{kq}\hat{\mathbf{w}}_q \;\; \forall\, k, q \in \mathcal{J} \tag{2.15}$$

with $\rho_{kq} = \frac{\rho_k^*}{\rho_q^*}$. Therefore, the set $(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_J)$ belongs to the solution space used by DANSE$_1$, as specified by (2.6), i.e. $(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_J) \in (\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_J)$.

In the theoretical convergence analysis infra, we assume that the correlation matrices $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ and the correlation vectors $\mathbf{r}^i_{\tilde{y}_k d_k}$, $\forall\, k \in \mathcal{J}$, are perfectly estimated, i.e. as if they are computed over an infinite observation window. Under this assumption, the following theorem guarantees convergence and optimality of the DANSE$_1$ algorithm.

Theorem 2.1 *If the sensor signal correlation matrix $\mathbf{R}_{yy}$ has full rank, and if $d_k = \rho_k d$, $\forall\, k \in \mathcal{J}$, with d a complex valued single-channel signal and $\rho_k \in \mathbb{C}\backslash\{0\}$, then the DANSE$_1$ algorithm converges for any initialization of its parameters to the MMSE solution (2.2) for all $k \in \mathcal{J}$.*

Before proving this theorem, we introduce some additional notation. The vector $\mathbf{w}$ (without subscript) denotes the stacked vector of all $\mathbf{w}_{kk}$ vectors, i.e.

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_{11} \\ \mathbf{w}_{22} \\ \vdots \\ \mathbf{w}_{JJ} \end{bmatrix}. \tag{2.16}$$

We also define the following MSE cost functions corresponding to node k:

$$J_k(\mathbf{w}_k) = E\{|d_k - \mathbf{w}_k^H \mathbf{y}|^2\} \tag{2.17}$$

$$\widetilde{J}_k(\mathbf{w}, \mathbf{g}_k) = J_k(\widetilde{\mathbf{w}}_k) \tag{2.18}$$

where $\widetilde{\mathbf{w}}_k$ is defined from $\mathbf{w}$ and $\mathbf{g}_k$ as in (2.6). Notice that $\mathbf{g}_k$ contains the entry g_{kk}, which is a fictitious variable that is never actually computed by the DANSE$_1$ algorithm. We define $F_k(\mathbf{w}^i)$ as the function that generates $\mathbf{w}_{kk}^{i+1}$ according to (2.9), i.e.

$$F_k(\mathbf{w}^i) = \mathbf{w}_{kk}^{i+1} = \left[\begin{array}{c:c} \mathbf{I}_{M_k} & \mathbf{O}_{M_k\times(J-1)} \end{array}\right] \left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}\right)^{-1} \mathbf{r}^i_{\tilde{y}_k d_k} \tag{2.19}$$

with $\mathbf{I}_P$ denoting a $P\times P$ identity matrix and $\mathbf{O}_{P\times Q}$ denoting an all-zero $P\times Q$ matrix. It is noted that the right-hand side of (2.19) depends on all entries of

the argument $\mathbf{w}^i$ through the signal $\mathbf{z}^i_{-k}$, which is not explicitly revealed in this expression.

The proof of Theorem 2.1 provided here differs from the proof in [10], where a scheme similar to DANSE$_1$ with $J = 2$ has been proved to converge to the optimal solution. Unlike the proof in [10], our proof allows for a generalization to the DANSE$_K$ case with $K > 1$, it allows $J > 2$, and provides more insight in the convergence properties of the algorithm. We first prove the convergence statement of Theorem 2.1, and then the optimality statement.

Proof : [Proof of convergence] We prove that the sequence $\left(\mathbf{w}^i\right)_{i\in\mathbb{N}}$ and the sequences $\left(\mathbf{g}^i_k\right)_{i\in\mathbb{N}}\ \forall\, k \in \mathcal{J}$ converge to a limit point $\mathbf{w}^\infty$ and $\mathbf{g}^\infty_k$ respectively. When node k performs an update of its variables $\mathbf{w}_{kk}$ and $\mathbf{g}_{k-k}$ at iteration i, these are replaced by the solution of the local MMSE problem (2.7), repeated here for convenience:

$$\min_{\mathbf{w}_{kk},\mathbf{g}_{k-k}} E\left\{\left|d_k - \begin{bmatrix} \mathbf{w}^H_{kk} & \mathbf{g}^H_{k-k} \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}^i_{-k} \end{bmatrix}\right|^2\right\} . \tag{2.20}$$

If node q were to optimize the variables $\mathbf{w}_{kk}$ and $\mathbf{g}_{q-k}$ with respect to its own node-specific estimation problem, it would solve the problem

$$\min_{\mathbf{w}_{kk},\mathbf{g}_{q-k}} E\left\{\left|d_q - \begin{bmatrix} \mathbf{w}^H_{kk} & \mathbf{g}^H_{q-k} \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}^i_{-k} \end{bmatrix}\right|^2\right\} . \tag{2.21}$$

Since $d^*_k = \rho_{kq} d^*_q$ with $\rho_{kq} = \frac{\rho^*_k}{\rho^*_q}$, the solution of (2.20) and (2.21) are identical up to a scalar ρ_{kq}. This means that an update of $\mathbf{w}_{kk}$ and $\mathbf{g}_{k-k}$ at node k, which is an optimization leading to a decrease of $\widetilde{J}_k$, will also lead to a decrease of $\widetilde{J}_q$ for any $q \in \mathcal{J}$ if node q were allowed to also perform a responding optimization of its $\mathbf{g}_q$. This shows that for any i (independent of the selection of the node u that actually performs an update at iteration i)

$$\forall\, k \in \mathcal{J} : \ \min_{\mathbf{g}_k} \widetilde{J}_k\left(\mathbf{w}^{i+1}, \mathbf{g}_k\right) \leq \min_{\mathbf{g}_k} \widetilde{J}_k\left(\mathbf{w}^i, \mathbf{g}_k\right) . \tag{2.22}$$

Since all $\widetilde{J}_k$ have a lower bound, each sequence $\left(\min_{\mathbf{g}_k} \widetilde{J}_k\left(\mathbf{w}^i, \mathbf{g}_k\right)\right)_{i\in\mathbb{N}}$ converges to a limit L_k, i.e.

$$\forall\, k \in \mathcal{J} :$$

$$i \to \infty : \ \min_{\mathbf{g}_k} \widetilde{J}_k\left(\mathbf{w}^{i+1}, \mathbf{g}_k\right) = \min_{\mathbf{g}_k} \widetilde{J}_k\left(\mathbf{w}^i, \mathbf{g}_k\right) = L_k . \tag{2.23}$$

If we again assume that node k performs an update at iteration i, then because of the strict convexity of the cost function in (2.20), the following expression holds:

$$\min_{\mathbf{g}_k} \widetilde{J}_k\left(\mathbf{w}^{i+1}, \mathbf{g}_k\right) = \min_{\mathbf{g}_k} \widetilde{J}_k\left(\mathbf{w}^i, \mathbf{g}_k\right) \Leftrightarrow \mathbf{w}^{i+1}_{kk} = \beta_k \mathbf{w}^i_{kk} \tag{2.24}$$

with

$$\beta_k = \arg\min_{g_{kk}} \left(\min_{\mathbf{g}_{k-k}} \widetilde{J}_k \left(\mathbf{w}^i, \mathbf{g}_k \right) \right) . \tag{2.25}$$

This shows that, after convergence of the sequences

$$\left(\min_{\mathbf{g}_k} \widetilde{J}_k \left(\mathbf{w}^i, \mathbf{g}_k \right) \right)_{i \in \mathbb{N}}, \forall\, k \in \mathcal{J}$$

any update of a $\mathbf{w}_{kk}$ must correspond to a scaling. Notice however that

$$F_k \left(\begin{bmatrix} \mathbf{w}_{11}^i \\ \vdots \\ \mathbf{w}_{JJ}^i \end{bmatrix} \right) = F_k \left(\begin{bmatrix} \beta_1 \mathbf{w}_{11}^i \\ \vdots \\ \beta_J \mathbf{w}_{JJ}^i \end{bmatrix} \right) \tag{2.26}$$

i.e. a scaling of a $\mathbf{w}_{qq}$ in node q does not change the update of $\mathbf{w}_{kk}$ in node k, since the scaling is implicitly compensated in F_k by the parameter g_{kq}. This proves convergence of the sequence $\left(\mathbf{w}^i\right)_{i \in \mathbb{N}}$ to a limit point $\mathbf{w}^\infty$ and therefore also the sequences $\left(\mathbf{g}_k^i\right)_{i \in \mathbb{N}}$ must converge to a limit point $\mathbf{g}_k^\infty$, $\forall\, k \in \mathcal{J}$. Notice that after convergence, based on what was stated earlier

$$\forall\, k, q \in \mathcal{J} : \ \widetilde{\mathbf{w}}_{kq}^\infty = \rho_{kq} \mathbf{w}_{qq}^\infty \tag{2.27}$$

or equivalently

$$\forall\, k, q \in \mathcal{J} : \ g_{kq}^\infty = \rho_{kq} . \tag{2.28}$$

□

From the proof of convergence, one can also conclude that convergence of the cost functions J_k will be monotonic, when sampled at the iteration steps in which node k updates its parameters. Indeed, whenever node q optimizes its own local MMSE problem, it also optimizes the corresponding MMSE problem in node k, at least when the latter is allowed to perform a responding update of its parameter g_{kq}. This shows that the DANSE$_1$ algorithm is at least as fast as a centralized equivalent that would use an alternating optimization (AO) technique [16], which is often referred to as the nonlinear Gauss-Seidel algorithm [17], with partitioning following directly from the parameters J and M_k for each node.

Proof : [Proof of optimality] We now prove that $\widetilde{\mathbf{w}}_k^\infty$ is the solution of (2.1) for every node k, which is equivalent to proving that the gradient of J_k is zero when evaluated at equilibrium, i.e.

$$\forall\, k \in \mathcal{J} : \ \nabla J_k(\widetilde{\mathbf{w}}_k^\infty) = 0 . \tag{2.29}$$

Because the solution of (2.20) sets the partial gradient of $\widetilde{J}_k$ with respect to $\mathbf{w}_{kk}$ to zero, we find that

$$\forall\, k \in \mathcal{J} : \ \nabla_{\mathbf{w}_{kk}} \widetilde{J}_k(\mathbf{w}^\infty, \mathbf{g}_k^\infty) = \nabla_{\mathbf{w}_{kk}} J_k(\widetilde{\mathbf{w}}_k^\infty) = 0 . \tag{2.30}$$

Since $d_k^* = \rho_{kq} d_q^*$, we can show that

$$\forall\, k, q \in \mathcal{J}:\ \nabla_{\mathbf{w}_{qq}} J_q(\mathbf{w}_q) = 0 \Leftrightarrow \nabla_{\mathbf{w}_{kq}} J_k(\rho_{kq}\mathbf{w}_q) = 0\ . \tag{2.31}$$

Combining (2.30) and (2.31) yields

$$\forall\, k, q \in \mathcal{J} : \nabla_{\mathbf{w}_{kq}} J_k(\rho_{kq}\widetilde{\mathbf{w}}_q^{\infty}) = 0\ . \tag{2.32}$$

Notice that (2.27) is equivalent with

$$\forall\, k, q \in \mathcal{J}:\ \widetilde{\mathbf{w}}_k^{\infty} = \rho_{kq}\widetilde{\mathbf{w}}_q^{\infty}\ . \tag{2.33}$$

Substituting (2.33) in (2.32) yields

$$\forall\, k, q \in \mathcal{J} : \nabla_{\mathbf{w}_{kq}} J_k(\widetilde{\mathbf{w}}_k^{\infty}) = 0 \tag{2.34}$$

which is equivalent to (2.29). This proves the theorem. □

2.4 DANSE with K-Channel Broadcast Signals

2.4.1 DANSE$_K$ Algorithm

In the DANSE$_K$ algorithm, each node broadcasts $\min\{K, M_k\}$-component compressed sensor signal observations to the other nodes. This compresses the data to be sent by node k by a factor of $\max\{\frac{M_k}{K}, 1\}$. We assume that each node k estimates a K-channel desired signal $\mathbf{d}_k = [d_k(1)\ \dots\ d_k(K)]^T$. Assuming that the desired signals $\mathbf{d}_k$ share a common Q-dimensional latent signal subspace, we will show in Section 2.4.2 that DANSE$_K$ achieves the optimal estimators if K is chosen equal to Q. Notice that the actual signal(s) of interest can be a subset of the vector $\mathbf{d}_k$, and the other entries should then be seen as auxiliary channels to fully capture the latent signal subspace, as explained in Section 2.2.2. Generally, these auxiliary channels are obtained by choosing $K-1$ extra reference sensors at node k.

Again, we use a linear estimator $\mathbf{W}_k$ to estimate $\mathbf{d}_k$ as $\overline{\mathbf{d}}_k = \mathbf{W}_k^H\mathbf{y}$, where $\mathbf{W}_k = [\mathbf{w}_k(1)\ ...\ \mathbf{w}_k(K)]$. The objective for node k is to find the linear MMSE estimator

$$\hat{\mathbf{W}}_k = \underset{\mathbf{W_k}}{\arg\min}\, E\left\{\|\mathbf{d}_k - \mathbf{W}_k^H\mathbf{y}\|^2\right\}\ . \tag{2.35}$$

The solution of (2.35) is

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^{-1}\mathbf{R}_{yd_k} \tag{2.36}$$

with $\mathbf{R}_{yd_k} = E\left\{\mathbf{y}\mathbf{d}_k^H\right\}$. Again, we define a partitioning of the estimator $\mathbf{W}_k$ as $\mathbf{W}_k = \left[\mathbf{W}_{k1}^T\ \dots\ \mathbf{W}_{kJ}^T\right]^T$ with $\mathbf{W}_{kq}$ denoting the $M_k \times K$ submatrix of $\mathbf{W}_k$ that is applied to $\mathbf{y}_q$. We wish to obtain (2.36) without the need for each node

to broadcast all M_k components of the $\mathbf{y}_k$ observations. Instead each node k will broadcast observations of the K-channel compressed signal $\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k$. Since the K channels of $\mathbf{z}_k$ will be highly correlated, further joint compression is possible, but we will not take this into consideration throughout this paper.

A node k can transform the observations of $\mathbf{z}_q$ that it receives from node q by a $K \times K$ transformation matrix $\mathbf{G}_{kq}$. Again, it is noted that $\mathbf{G}_{kq}$ does not decompress the observations of the signal $\mathbf{z}_q$, but makes K new linear combinations of their components. The parametrization of the $\mathbf{W}_k$ effectively applied at node k is then

$$\widetilde{\mathbf{W}}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{kJ} \end{bmatrix} \tag{2.37}$$

which is a generalization of (2.6). Here, node k can only optimize the parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_k = [\mathbf{G}_{k1}^T \ \dots \ \mathbf{G}_{kJ}^T]^T$. We set $\mathbf{G}_{kk} = \mathbf{I}_K$ with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix.

The (KJ)-channel signal $\mathbf{z} = \left[\mathbf{z}_1^T \dots \mathbf{z}_J^T\right]^T$ is a stacked version of all the broadcast signals. Similarly to the notation in Section 2.3, we define the signal $\mathbf{z}_{-k}$ as the signal $\mathbf{z}$ with $\mathbf{z}_k$ omitted, and we define $\mathbf{G}_{k-q}$ as the matrix $\mathbf{G}_k$ with the submatrix $\mathbf{G}_{kq}$ omitted. The MMSE problem that is solved at node k, at iteration i, is now

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \underset{\mathbf{W}_{kk},\mathbf{G}_{k-k}}{\arg\min} \ E\left\{ \left\| \mathbf{d}_k - \begin{bmatrix} \mathbf{W}_{kk}^H & \mathbf{G}_{k-k}^H \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix} \right\|^2 \right\} . \tag{2.38}$$

The solution of (2.38) is

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \left(\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i\right)^{-1} \mathbf{R}_{\tilde{y}_k d_k}^i \tag{2.39}$$

with $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$ defined as in (2.10) and with

$$\mathbf{R}_{\tilde{y}_k d_k}^i = E\{\widetilde{\mathbf{y}}_k^i \mathbf{d}_k^H\} . \tag{2.40}$$

The DANSE$_K$ algorithm is described in Table 2.2 on the next page. It is a straightforward generalization of the DANSE$_1$ algorithm as explained in Section 2.3.1, where all vector-variables are replaced by their matrix equivalent. Similarly, expressions (2.16)-(2.19) can be straightforwardly generalized to their matrix equivalent.

The DANSE$_K$ Algorithm

1. Initialize: $i \leftarrow 0$, $u \leftarrow 1$
   Initialize $\mathbf{W}_{kk}^0$ and $\mathbf{G}_{k-k}^0$ with random matrices, $\forall\, k \in \mathcal{J}$
2. Each node $k \in \mathcal{J}$ performs the following operation cycle:
   - Collect the sensor observations $\mathbf{y}_k[iB+n]$, $n = 0 \dots B-1$.
   - Compress these M_k-dimensional observations to K-dimensional vectors

$$\mathbf{z}_k^i[iB+n] = \mathbf{W}_{kk}^{i\,H}\mathbf{y}_k[iB+n],\ \ n = 0 \dots B-1 \tag{2.41}$$

   - Broadcast the compressed observations $\mathbf{z}_k^i[iB+n]$, $n = 0 \dots B-1$, to the other nodes.
   - Collect the $K(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \dots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   - Update the estimates of $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$ and $\mathbf{R}_{\tilde{y}_k d_k}^i$, by including the newly collected data.
   - Update the node-specific parameters:

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \begin{cases} \left(\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i\right)^{-1}\mathbf{R}_{\tilde{y}_k d_k}^i & \text{if } k = u \\ \begin{bmatrix} \mathbf{W}_{kk}^i \\ \mathbf{G}_{k-k}^i \end{bmatrix} & \text{if } k \neq u \end{cases} \tag{2.42}$$

   - Compute the estimate of $\mathbf{d}_k[iB+n]$, $n = 0 \dots B-1$, as

$$\begin{aligned} \overline{\mathbf{d}}_k[iB+n] = &\mathbf{W}_{kk}^{i+1\,H}\mathbf{y}_k[iB+n] \\ &+ \mathbf{G}_{k-k}^{i+1\,H}\mathbf{z}_{-k}^i[iB+n]\ . \end{aligned} \tag{2.43}$$

3. $i \leftarrow i+1$
4. $u \leftarrow (u \mod J)+1$
5. return to step 2

Table 2.2: The DANSE$_K$ algorithm

2.4.2 Convergence and Optimality of DANSE$_K$ if $Q = K$ and $\mathbf{A}_k$ Full Rank

We now assume that $\mathbf{d}_k = \mathbf{A}_k\mathbf{d}$, $\forall\, k \in \mathcal{J}$, with $\mathbf{A}_k$ a $K \times K$ matrix of rank K and $\mathbf{d}$ a complex valued K-channel signal. This means that all desired signals $\mathbf{d}_k$ share the same K-dimensional latent signal subspace (i.e. $Q = K$). Formula (2.36) shows that in this case all $\hat{\mathbf{W}}_k$ have the same column space, i.e.

$$\forall\, k, q \in \mathcal{J} : \hat{\mathbf{W}}_k = \hat{\mathbf{W}}_q\mathbf{A}_{kq} \tag{2.44}$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H}\mathbf{A}_k^H$. Therefore, the set $\left(\hat{\mathbf{W}}_1, \ldots, \hat{\mathbf{W}}_J\right)$ belongs to the solution space used by DANSE$_K$, as specified by (2.37), i.e. $\left(\hat{\mathbf{W}}_1, \ldots, \hat{\mathbf{W}}_J\right) \in \left(\widetilde{\mathbf{W}}_1, \ldots, \widetilde{\mathbf{W}}_J\right)$. The following theorem generalizes Theorem 2.1:

Theorem 2.2 *If the sensor signal correlation matrix $\mathbf{R}_{yy}$ has full rank, and if $\mathbf{d}_k = \mathbf{A}_k\mathbf{d}$, $\forall\, k \in \mathcal{J}$, with $\mathbf{d}$ a complex valued K-channel signal and $\mathbf{A}_k$ a $K \times K$ matrix of rank K, then the DANSE$_K$ algorithm converges for any initialization of its parameters to the MMSE solution (2.36) for all $k \in \mathcal{J}$.*

Proof : The proof of Theorem 2.1 can straightforwardly be generalized to prove Theorem 2.2, by replacing every $\mathbf{w}$ and $\mathbf{g}$ by its matrix version $\mathbf{W}$ and $\mathbf{G}$. □

In practice, the matrices $\mathbf{A}_k$ should be well-conditioned to obtain the optimal estimators, which is reflected in Theorem 2.2 by the condition that $\mathbf{A}_k$ has full rank. If the K-channel desired signal $\mathbf{d}_k$ is defined as the target signal in K reference sensors at node k, this matrix can be ill-conditioned if the reference sensors are close to each other. This problem is investigated in [9], where the DANSE algorithm is used for noise reduction in acoustic sensor networks, and a solution is proposed to tackle this problem.

2.4.3 DANSE Under Rank Deficiency

Until now, we have avoided the case where $\mathbf{R}_{yy}$ does not have full rank or when the parameter K is overestimated, i.e. $K > Q$. Both cases can result in broadcast data for which the correlation matrix is rank deficient[4]. In this case, (2.38) becomes ill-posed since singular correlation matrices are involved. The DANSE$_K$ algorithm can cope with these situations by adding a minimum-norm constraint to the local MMSE problems (2.38), i.e. using the pseudo-inverse

[4]In the case where $K > Q$, (2.44) has multiple solutions for $\mathbf{A}_{kq}$ since rank$(\hat{\mathbf{W}}_k) = Q$, $\forall\, k \in \mathcal{J}$. Therefore, the correlation matrix of the broadcast signal $\mathbf{z}_k^i$ becomes singular, once the $M_k \times K$ submatrix $\mathbf{W}_{kk}^i$ reaches this rank deficiency.

instead of a matrix inverse in the computation of the solution of (2.38) [15]. Extensive simulations have shown that with this modification, the DANSE$_K$ algorithm still converges to an MMSE solution for rank deficient estimation problems (see Section 2.6).

However, if the matrix $\mathbf{R}_{yy}$ does not have full rank, the solution of (2.1) is not unique. Simulations have shown that the solutions $\widetilde{\mathbf{W}}_k^\infty$ obtained by the DANSE$_K$ algorithm, although leading to a minimal MSE cost at node k, are generally different from the solutions provided by the centralized minimum norm version, i.e.

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^\dagger \mathbf{R}_{yd_k} \tag{2.45}$$

where superscript † denotes the pseudo-inverse.

2.5 DANSE$_K$ Implementation Aspects

2.5.1 Estimation of the Signal Statistics

In the theoretical analysis of the DANSE$_K$ algorithm, it is assumed that the second order signal statistics, which are needed to solve the MMSE problem (2.38) are perfectly known. However, in a practical application, the correlation matrices $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ and $\mathbf{R}^i_{\tilde{y}_k d_k}$ have to be estimated, based on the collected signal observations. In this section, we will describe some strategies to estimate these quantities.

Estimation of signal correlation matrices is typically done by time averaging. This means that some assumptions are made on short-term ergodicity and stationarity of the signals involved. However, this stationarity assumption is not necessarily strict. Even when the signals involved are non-stationary (such as in speech processing), the DANSE$_K$ algorithm can provide good estimators. By using long-term correlation matrices, the influence of rapidly changing temporal statistics is smoothed out, yielding estimators that mainly exploit the spatial coherence between the sensors. Since spatial coherence typically changes slowly, the DANSE$_K$ algorithm is able to provide good estimators, even when the signals themselves are highly non-stationary (this is e.g. demonstrated by the multi-channel speech enhancement experiments in [9]).

We let $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t]$ denote the estimate of $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ at time t. Signal correlation matrices are often estimated in practice by means of a forgetting factor $0 < \lambda < 1$, i.e.

$$\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t] = \lambda \mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t-1] + (1-\lambda)\widetilde{\mathbf{y}}^i_k[t]\widetilde{\mathbf{y}}^i_k[t]^H \ . \tag{2.46}$$

Notice that in the DANSE$_K$ algorithm, the statistics change every time a node updates its parameters. Therefore, (2.46) is not suited to compute $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t]$ and $\mathbf{R}^i_{\tilde{y}_k d_k}[t]$, since it uses an infinite time window. A better alternative is a

simple time averaging in a finite observation window, i.e.

$$\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t] = \frac{1}{L}\sum_{n=t-L+1}^{t} \widetilde{\mathbf{y}}^i_k[n]\widetilde{\mathbf{y}}^i_k[n]^H \tag{2.47}$$

where L is the length of the observation window. The procedure (2.46) puts more emphasis on the most recent samples, whereas (2.47) applies an equal weight to all past samples in the observation window. The procedure (2.47) can be implemented recursively by means of an updating and a downdating term, i.e.

$$\begin{aligned}\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t] =& \mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t-1] + \frac{1}{L}\widetilde{\mathbf{y}}^i_k[t]\widetilde{\mathbf{y}}^i_k[t]^H \\ &- \frac{1}{L}\widetilde{\mathbf{y}}^i_k[t-L+1]\widetilde{\mathbf{y}}^i_k[t-L+1]^H \ .\end{aligned} \tag{2.48}$$

Notice that the window length L introduces a trade-off between tracking performance and estimation performance. Indeed, to have a fast tracking, the statistics must be estimated from short signal segments, yielding larger estimation errors in the correlation matrices that are used to compute the estimators at the different nodes. However, as will be demonstrated in Section 2.6.2, the DANSE$_K$ algorithm is more robust to these errors, compared to the equivalent centralized algorithm, due to the fact that DANSE$_K$ uses correlation matrices with smaller dimensions than the network-wide estimation problem.

The estimation of $\mathbf{R}^i_{\tilde{y}_k d_k}$ is less straightforward since the signal $\mathbf{d}_k$ cannot be observed directly. However, depending on the application and the signals involved, some strategies can be developed to estimate $\mathbf{R}^i_{\tilde{y}_k d_k}$, as explained in the following two examples.

If the transmitting sources are controlled by the application itself, as it is the case in a communications scheme, the source signals that define the different channels in $\mathbf{d}$ can be manipulated directly. At periodic intervals, a deterministic training sequence can be broadcast by the transmitters. If the nodes have knowledge about these training sequences, they can use this to compute $\mathbf{R}^i_{\tilde{y}_k d_k}[t]$ in a similar way as in (2.48), during the broadcast of these training sequences. After the broadcast, the estimate is fixed until new training sequences are broadcast.

A different strategy can be applied if the desired signal $\mathbf{d}_k$ has an on-off behavior[5]. Assume that the sensor signals in $\mathbf{y}$ consist of a desired component $\mathbf{x}$ and an additive noise component $\mathbf{n}$, i.e. $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{x}$ has an on-off behavior, and where then $\mathbf{d}_k = [x_{k1} \ \ldots \ x_{kK}]^T$. In many practical applications, it can

[5]This is often used in speech enhancement applications, since a speech signal typically contains a lot of silent pauses in between words or sentences.

also be assumed that $\mathbf{x}$ and $\mathbf{n}$ are independent, and therefore[6]

$$E\{\mathbf{x}\mathbf{x}^H\} = E\{\mathbf{y}\mathbf{y}^H\} - E\{\mathbf{n}\mathbf{n}^H\} \,. \tag{2.49}$$

If there is a detection mechanism available that detects whether the signal $\mathbf{x}$ is present or not, one can estimate $E\{\mathbf{n}\mathbf{n}^H\}$ in time segments where only noise is observed ("noise-only segments"). Since the noise is uncorrelated to the desired component $\mathbf{d}_k$, we find that

$$\mathbf{R}^i_{\tilde{y}_k d_k} = E\{\widetilde{\mathbf{x}}^i_k \widetilde{\mathbf{x}}^{i\,H}_k\}\mathbf{E}_K \tag{2.50}$$

with

$$\mathbf{E}_K = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{O}_{(M_k+(J-2)K)\times K} \end{bmatrix} \tag{2.51}$$

where $\widetilde{\mathbf{x}}^i_k$ is the desired component in the signal $\widetilde{\mathbf{y}}^i_k$. The selection matrix $\mathbf{E}_K$ is used to select the first K columns corresponding to $\mathbf{d}_k = [x_{k1} \dots x_{kK}]^T$. Define the noise correlation matrix

$$\mathbf{R}^i_{\tilde{n}_k \tilde{n}_k} = E\{\widetilde{\mathbf{n}}^i_k \widetilde{\mathbf{n}}^{i\,H}_k\} \tag{2.52}$$

where $\widetilde{\mathbf{n}}^i_k$ denotes the noise component in the signal $\widetilde{\mathbf{y}}^i_k$. With (2.50), and similarly to (2.49), we readily find that

$$\mathbf{R}^i_{\tilde{y}_k d_k} = \left(\mathbf{R}^i_{\tilde{y}_k \tilde{y}_k} - \mathbf{R}^i_{\tilde{n}_k \tilde{n}_k}\right) \mathbf{E}_K \,. \tag{2.53}$$

Using (2.53), one can compute $\mathbf{R}^i_{\tilde{y}_k d_k}[t]$ as the difference between $\mathbf{R}^i_{\tilde{y}_k \tilde{y}_k}[t]$ and $\mathbf{R}^i_{\tilde{n}_k \tilde{n}_k}[t]$, where the latter is computed as in (2.48), during noise-only periods.

Notice that, even if the target signal does not have this on-off behaviour, the above strategy can be used in a semi-adaptive context, i.e. where the target signal statistics may change but the noise statistics are static and a priori known (or vice versa). Indeed, if $E\{\mathbf{n}\mathbf{n}^H\}$ is known, then (2.53) can be used to compute the required statistics. Notice that $\mathbf{R}^i_{\tilde{n}_k \tilde{n}_k}$ in (2.53) is a compressed version of $E\{\mathbf{n}\mathbf{n}^H\}$, i.e. it depends on the current parameters in $\mathbf{W}^i$. Therefore, each node k has to broadcast the entries of $\mathbf{W}^i_{kk}$, which are needed in the other nodes to compress the corresponding submatrices in $E\{\mathbf{n}\mathbf{n}^H\}$. Since these values change only once for each JB observations that are collected by the sensors, the resulting increase in bandwidth is negligible compared to the transmission of the samples of $\mathbf{z}^i_k$.

2.5.2 Computational Complexity

The estimation of the correlation matrices $\mathbf{R}^i_{\tilde{y}_k \tilde{y}_k}$ and $\mathbf{R}^i_{\tilde{y}_k d_k}$, and the inversion of the former, are the most computationally expensive steps of the DANSE$_K$

[6] For the sake of an easy exposition, we assume that the signals $\mathbf{x}$ and $\mathbf{n}$ have zero mean.

algorithm. From (2.48) it follows that an update of $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t]$ at node k, has a computational complexity of

$$O\left((M_k + K(J-1))^2\right) \tag{2.54}$$

i.e. it is quadratic in the number of nodes J, the number of channels K in the broadcast signals, and the number of channels M_k of the signal $\mathbf{y}_k$. If node k updates its parameters $\mathbf{W}^i_{kk}$ and $\mathbf{G}^i_{k-k}$ according to (2.39), it performs a matrix inversion, which is computationally more expensive than (2.54). However, instead of computing this inversion, node k can directly update the inverse of $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t]$ at each time t by means of the matrix inversion lemma [15], i.e.

$$\begin{aligned}\left(\mathbf{R}^{i\,\text{temp}}_{\tilde{y}_k\tilde{y}_k}\right)^{-1} &= \left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t-1]\right)^{-1} \\ &- \frac{\left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t-1]\right)^{-1}\widetilde{\mathbf{y}}^i_k[t]\widetilde{\mathbf{y}}^i_k[t]^H\left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t-1]\right)^{-1}}{1+\widetilde{\mathbf{y}}^i_k[t]^H\left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t-1]\right)^{-1}\widetilde{\mathbf{y}}^i_k[t]}\end{aligned} \tag{2.55}$$

$$\begin{aligned}\left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}[t]\right)^{-1} &= \left(\mathbf{R}^{i\,\text{temp}}_{\tilde{y}_k\tilde{y}_k}\right)^{-1} \\ &+ \frac{\left(\mathbf{R}^{i\,\text{temp}}_{\tilde{y}_k\tilde{y}_k}\right)^{-1}\widetilde{\mathbf{y}}^i_k[t-L+1]\widetilde{\mathbf{y}}^i_k[t-L+1]^H\left(\mathbf{R}^{i\,\text{temp}}_{\tilde{y}_k\tilde{y}_k}\right)^{-1}}{1-\widetilde{\mathbf{y}}^i_k[t-L+1]^H\left(\mathbf{R}^{i\,\text{temp}}_{\tilde{y}_k\tilde{y}_k}\right)^{-1}\widetilde{\mathbf{y}}^i_k[t-L+1]}.\end{aligned} \tag{2.56}$$

This update also has computational complexity (2.54), and therefore this is the overall complexity for a single node in the DANSE$_K$ algorithm.

2.6 Numerical Simulations

In this section, we provide simulation results to demonstrate the behavior of the DANSE$_K$ algorithm. In Section 2.6.1, we perform batch mode simulations where the required statistics are computed over the full length signals, and where the $\mathbf{d}_k$'s are available[7] to compute $\mathbf{R}^i_{\tilde{y}_k d_k}$. In the batch version of DANSE$_K$, all iterations are performed on the same set of signal observations. In Section 2.6.2, a more practical scenario with moving sources is considered. The DANSE$_K$ algorithm adapts to the changes in the scenario, and each set of observations is only broadcast once, i.e. subsequent iterations are performed over different observation sets. Furthermore, a practical estimation of the correlation matrices is used, where the $\mathbf{d}_k$'s are assumed to be unavailable.

[7] This is similar to using a priori known training sequences.

2.6.1 Batch Mode Simulations

In this section, we simulate the DANSE$_K$ algorithm in batch mode. This means that all iterations are performed on the full signal length. The network consists of 4 nodes ($J = 4$), each having 10 sensors (M=40). The dimension of the latent signal subspace defined by $\mathbf{d}$ is $Q = 3$. All 3 channels of $\mathbf{d}$ are uniformly distributed random processes on the interval $[-0.5, 0.5]$ from which $N = 10000$ samples are generated. The coefficients in $\mathbf{A}_k$ are generated by a uniform random process on the unit interval. The sensor signals in $\mathbf{y}$ consist of the different random mixtures of the latent Q-channel signal $\mathbf{d}$ to which zero-mean white noise is added with half the power of the channels of $\mathbf{d}$. The initial values of all $\mathbf{W}_{kk}$ and $\mathbf{G}_k$ are taken from a uniform random distribution on the unit interval.

The batch mode performance of the DANSE$_1$ algorithm as well as the DANSE$_3$ algorithm is simulated for this particular scenario. All evaluations of the MSE cost functions J_k are performed on the equivalent least-squares (LS) cost functions, i.e.

$$\sum_{t=0}^{N} \|\mathbf{d}_k[t] - \mathbf{W}_k^H \mathbf{y}[t]\|^2 \,. \tag{2.57}$$

Also, the correlation matrices are replaced by their least squares equivalent, i.e. $E\{\mathbf{y}\mathbf{y}^H\}$ is replaced by $\mathbf{Y}\mathbf{Y}^H$ where $\mathbf{Y}$ denotes the $M \times N$ sample matrix that contains samples of the variable $\mathbf{y}$ in its columns.

The results are illustrated in Fig. 2.4, showing the LS cost of node 1 versus the iteration index i. Node 1 is the first node that performs an update. It is observed that the DANSE$_3$ algorithm converges to the optimal linear LS solution, whereas the DANSE$_1$ algorithm does not since $K < Q$ in this case. Downsampling the curve corresponding to DANSE$_3$ by a factor $J = 4$, keeping only the iterations in which node 1 updates its parameters, results in a monotonically decreasing cost. This is because of expression (2.22), showing that the cost indeed monotonically decreases whenever a node optimizes its $\mathbf{G}$ parameters. If the curve corresponding to DANSE$_1$ is downsampled with the same factor, we do not obtain a monotonically decreasing cost, since expression (2.22) is not valid anymore for this case.

In Fig. 2.5, we vary the number of nodes J, keeping all other parameters unchanged. All nodes again have 10 sensors. Not surprisingly, the convergence time of DANSE$_3$ increases linearly with J since the effective number of updates per time unit in node 1 is reduced. As soon as each node has updated its parameters three times, the cost is almost at its minimum at each node.

In Fig. 2.6(a), we increase the value of K while keeping $Q = 3$. Notice that this corresponds to the case where K is overestimated and hence communication bandwidth is used inefficiently. The estimation problem becomes rank

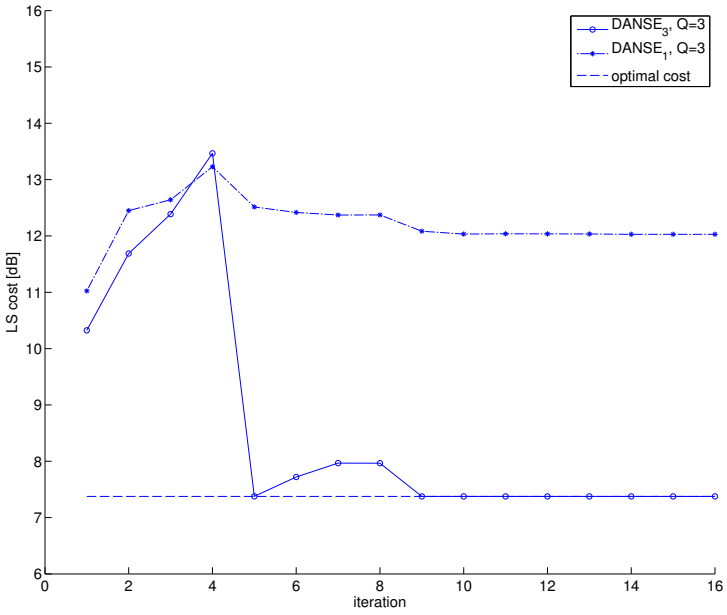


Figure 2.4: LS error of node 1 versus iteration i for four different scenarios in a network with $J = 4$ nodes. Each node has 10 sensors.

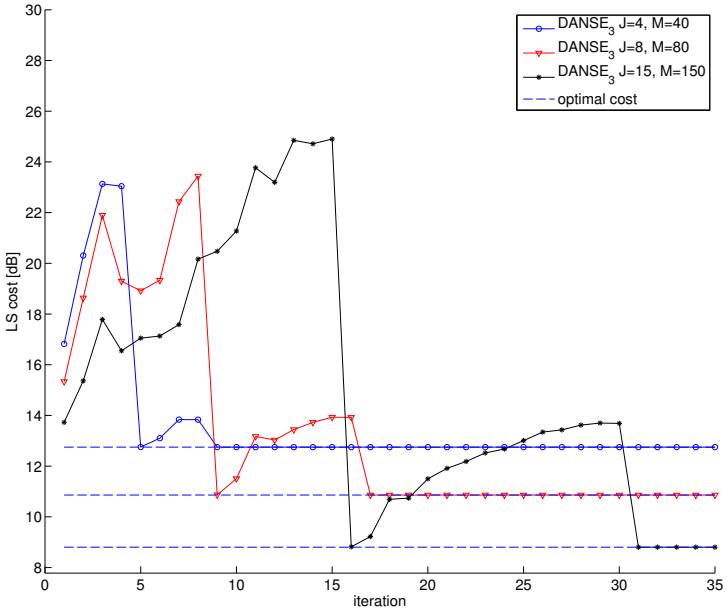
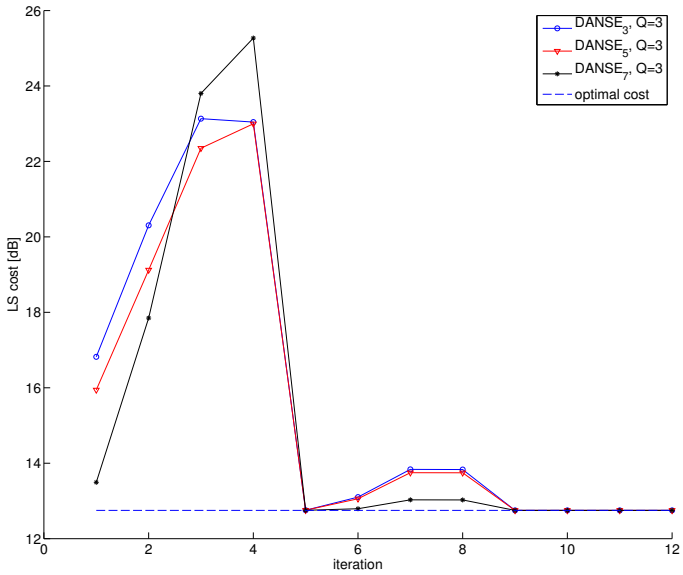
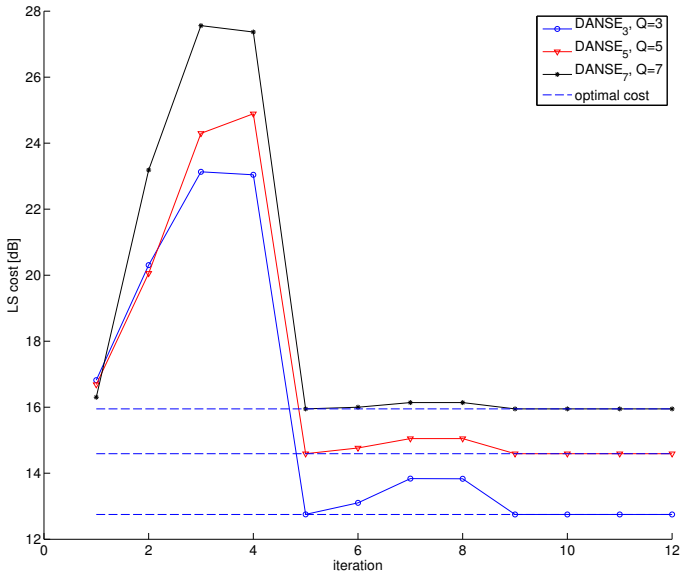


Figure 2.5: LS error of node 1 versus iteration i for networks with $J = 4$, $J = 8$ and $J = 15$ nodes respectively. Each node has 10 sensors.



(a) Different values of K, keeping $Q = 3$



(b) Different values of $K = Q$

Figure 2.6: LS error of node 1 versus iteration i in a network with $J = 4$ nodes. Each node has 10 sensors.

deficient in this case, and so the algorithm should be modified by replacing matrix inversions by pseudo-inversions (see Section 2.4.3). The algorithm still converges, and the optimal LS cost is again reached after two iterations per node when K is overestimated. In Fig. 2.6(b), we increase the value of K together with Q, keeping $Q = K$. This is again observed to have a negligible effect on convergence time.

As a general conclusion, we can state that for all settings of the parameters K, Q, J, the DANSE$_K$ algorithm approximately achieves convergence as soon as each node has updated its parameters three times.

Simulation results with speech signals are provided in a follow-up paper [9]. In this paper, a distributed speech enhancement algorithm based on DANSE$_K$ and its variations, is tested in a simulated acoustic sensor network scenario.

2.6.2 Adaptive Implementation

In this section, we show simulation results of a practical implementation of the DANSE$_K$ algorithm in a scenario with moving sources. The main difference with the batch mode simulations is that subsequent iterations are now performed on different signal segments, i.e. the same data is never used twice. This yields larger estimation errors, since shorter signal segments are used to estimate the statistics of the input signals. Furthermore, we will use a practical estimation procedure to estimate the correlation matrices $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ and $\mathbf{R}^i_{\tilde{y}_k d_k}$, yielding larger estimation errors.

The scenario is depicted in Fig. 2.7. The network contains $J = 4$ nodes ($\diamond$). Each node has a reference sensor at the node itself, and can collect observations of 5 additional sensors ($\circ$) that are uniformly distributed within a 1.6 m radius around the node. Eight localized white Gaussian noise sources ($\bigtriangledown$) are present. Two target sources ($\square$) move back and forth over the indicated straight lines at a speed of 1 m/s, and halt for 2 seconds at the end points of these lines. The first source (moving on the vertical line) transmits a low-pass filtered white noise signal with a cut-off frequency of 1600 Hz. The other source transmits a band-pass filtered white noise signal in the frequency range 1600 - 3200 Hz. Both target sources have an on-off behavior with a period of 0.2 seconds and both are active 66% of the time. It is assumed that at each time t, all nodes can detect whether the sources are active or not. The time between two consecutive updates is 0.4 s, which corresponds to 2 on-off cycles of the target sources. This means that, every 0.4 seconds, the iteration index i changes to $i+1$. The sensors observe their signals at a sampling frequency of $f_s = 16$ kHz.

The target source signals have half the power of the noise sources. In addition to the spatially correlated noise, independent white Gaussian sensor noise is added to each sensor signal. This noise component is 10% of the power of the localized

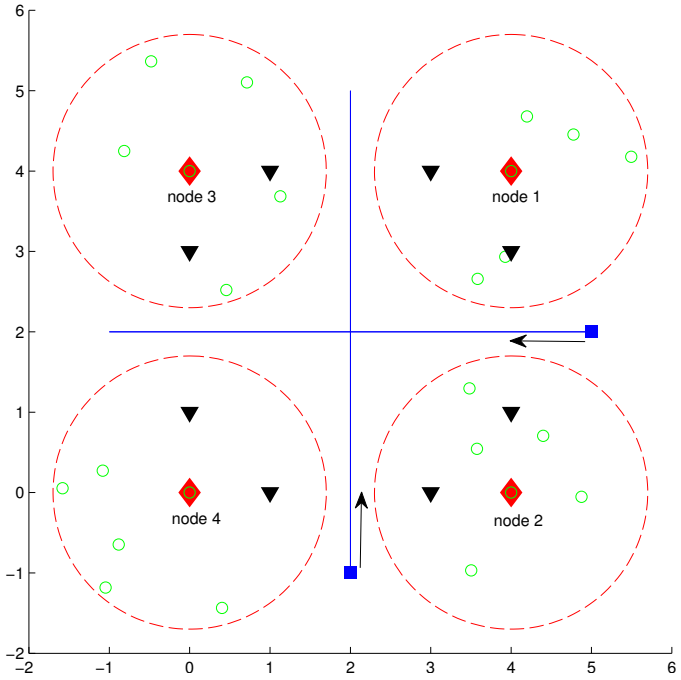


Figure 2.7: Description of the simulated scenario. The network contains 4 nodes ($\diamond$), each node collecting observations in a cluster of 6 sensors ($\circ$). One sensor of each cluster is positioned at the node itself. Two target sources ($\square$) are moving over the indicated straight lines. Eight noise sources are present ($\triangledown$).

noise signals. The individual signals originating from the target sources and the noise sources that are collected by a specific sensor are attenuated in power and summed. The attenuation factor of the signal power is $\frac{1}{r}$, where r denotes the distance between the source and the sensor. We assume that there is no time delay in the transmission path between the sources and the sensors[8]. Each node collects 6 sensor signal observations, and uses 5 differently delayed versions of each of these signals in its estimation process to exploit the temporal correlation in the target source signals. This means that $M_k = 30$.

We let y_{k1} denote the signal that is collected at the reference sensor of node k. It consists of an unknown mixture d_{k1} of the two target source signals, and a

[8]Since the time delays are the same for all sensors, the spatial information is purely energy based in this case. Therefore, the nodes cannot perform any beamforming towards specific locations by exploiting different delay paths between sources and sensors.

noise component n_{k1}, i.e.

$$y_{k1} = d_{k1} + n_{k1} = \mathbf{a}_{k1}^T \mathbf{d} + n_{k1} \tag{2.58}$$

where $\mathbf{d}$ is the 2-channel signal containing the two target source signals, and where $\mathbf{a}_{k1}$ denotes an unknown mixture vector. The goal for node k is to estimate the signal d_{k1}, i.e. the target source component in its reference sensor. Since $Q = 2$, the DANSE$_2$ algorithm is used, and therefore an auxiliary desired channel is used to obtain a 2-channel desired signal $\mathbf{d}_k$ at every sensor. The auxiliary channel of $\mathbf{d}_k$ consists of the target source component d_{k2} in the signal y_{k2} that is collected by another sensor of node k. This component consists of another unknown mixture of the target sources, so that the conditions of Theorem 2.2 are satisfied.

The correlation matrix $\mathbf{R}^i_{\tilde{y}_k d_k}[t]$ is computed according to (2.53). The estimates $\mathbf{R}^i_{\tilde{y}_k \tilde{y}_k}[t]$ and $\mathbf{R}^i_{\tilde{n}_k \tilde{n}_k}[t]$ are computed similarly to (2.48) with a window length of $L_1 = 4200$ and $L_2 = 2200$ respectively, which matches the time between two consecutive updates.

We will use the signal-to-error ratio (SER) as a measure to assess the performance of the estimators. The instantaneous SER for node k at time t and iteration i is computed over 3200 samples, and is defined as

$$\text{SER}_k^i[t] = \frac{\sum_{n=t+1}^{t+3200} |d_{k1}[n]|^2}{\sum_{n=t+1}^{t+3200} |d_{k1}[n] - \widetilde{\mathbf{w}}_k^i(1)^H \mathbf{y}[n]|^2} \tag{2.59}$$

where $\widetilde{\mathbf{w}}_k^i(1)$ denotes the first column of the estimator $\widetilde{\mathbf{W}}_k^i$, as defined in (2.37). Notice that this is the estimator that is of actual interest, since it estimates the desired component d_{k1} in the reference sensor. The other column of $\widetilde{\mathbf{W}}_k^i$ is viewed as an auxiliary estimator that is used for the generation of the second channel of the broadcast signal $\mathbf{z}_k^i$.

Fig. 2.8 shows the SER of the four nodes at different time instants. Dashed vertical lines are plotted to indicate the points in time where both sources start moving, and full vertical lines indicate when they stop moving. The sources stand still in the time intervals [0-4] s, [10-12] s and [18-20] s. The performance is compared to the centralized version, in which all sensor signals are centralized in a single fusion center that computes the optimal estimators according to (2.2).

In the first 4 seconds, both sources stand still. The DANSE$_2$ algorithm needs some time to reach a good estimator at each node (about 2 seconds), whereas the centralized algorithm converges much faster. This is because the DANSE$_2$ algorithm updates its nodes one at a time, with 0.4 seconds in between two consecutive updates. The centralized algorithm on the other hand, can update its estimators every time a new sample is collected. After a number of iterations however, the DANSE$_2$ algorithm converges to the optimal estimators.

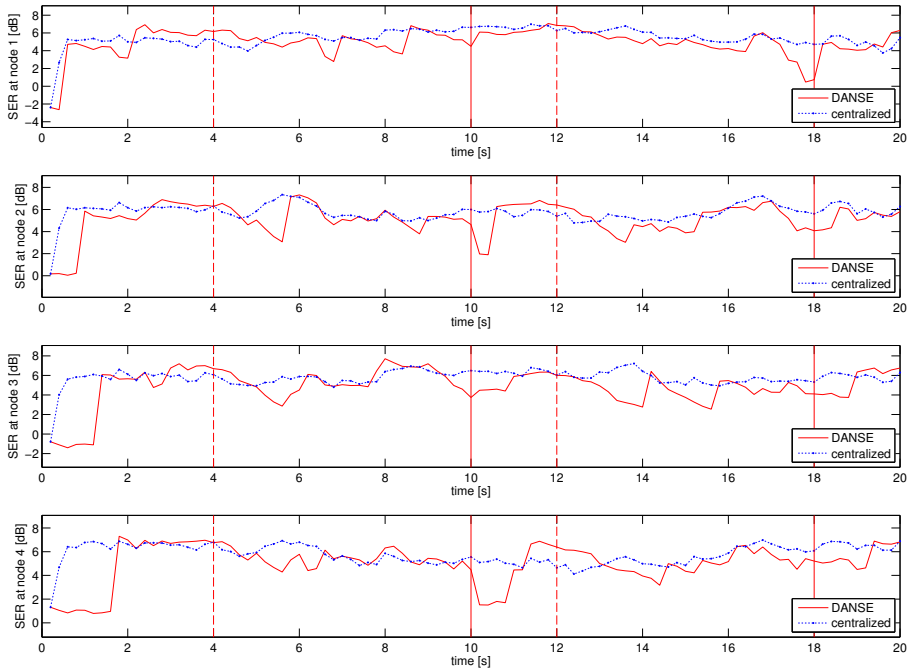


Figure 2.8: SER vs. time at the 4 nodes depicted in Fig. 2.7. The centralized version is added as a reference. Window lengths are $L_1 = 4200$ and $L_2 = 2200$.

Not surprisingly, it is observed that the centralized algorithm has better tracking capabilities than the DANSE$_2$ algorithm. This is again a consequence of the fact that the centralized version computes a new estimator each time a new sample is collected, yielding a much faster convergence. However, the DANSE$_2$ algorithm is able to react to changes in the scenario and always regains optimality after a number of iterations.

Notice that, once the DANSE$_2$ algorithm has converged, it outperforms the centralized algorithm. This can be explained by the fact that the DANSE$_2$ algorithm uses correlation matrices with smaller dimension compared to the correlation matrices that are used by the centralized algorithm. Small matrices are generally better conditioned and have a smaller estimation error than larger matrices. This performance increase of DANSE$_K$ compared to its centralized version is observed to become more significant when the number of sensors M increases, yielding larger matrices, or when the window length L decreases, yielding larger estimation errors in the correlation matrices. Fig. 2.9 shows the performance of DANSE$_2$ and its centralized version, now with window lengths $L_1 = 2100$ and $L_2 = 1100$, i.e. roughly half the sizes of the first experiment. It is observed that the estimation performance of the centralized

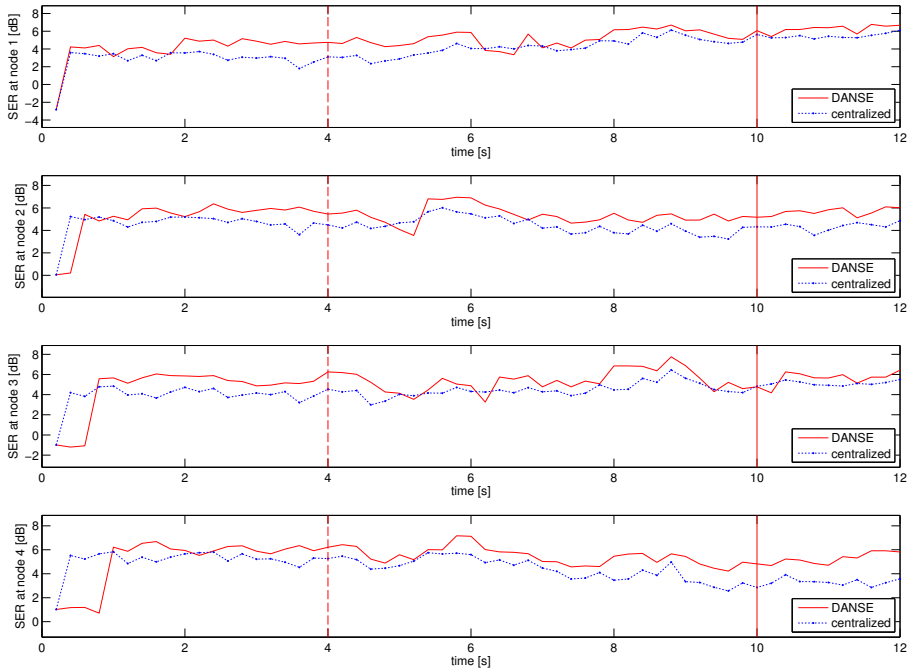


Figure 2.9: SER vs. time at the 4 nodes depicted in Fig. 2.7. The centralized version is added as a reference. Window lengths are $L_1 = 2100$ and $L_2 = 1100$.

algorithm significantly decreases compared to the first experiment, whereas the DANSE$_2$ algorithm is less influenced by the short window length. This observation demonstrates that DANSE$_K$ is more robust to estimation errors in the correlation matrices compared to its centralized equivalent. Notice that DANSE$_K$ converges much faster in the second experiment, since the time between two consecutive updates is now 0.2 seconds instead of 0.4 seconds, due to the shorter window lengths. As already mentioned in Section 2.5, this faster tracking comes with the drawback that the estimation performance decreases due to larger errors in the estimation of the correlation matrices.

In [14], a modified DANSE$_K$ algorithm is studied, where an improved tracking performance is obtained, by letting nodes update simultaneously.

2.7 Conclusion

In this paper, we have introduced a distributed adaptive algorithm (DANSE$_K$) for linear MMSE estimation of node-specific signals in a fully connected broad-

casting sensor network, where each sensor node collects multi-channel sensor signal observations. The algorithm significantly compresses the data to be broadcast, and the computational load is shared amongst the nodes. It is shown that, if the node-specific desired signals share a common low-dimensional latent signal subspace, DANSE$_K$ converges and provides the optimal linear MMSE estimator for every node-specific estimation problem, as if all nodes have access to all the sensor signals in the network. Simulations demonstrate that the algorithm achieves the same performance as a centralized algorithm. A practical adaptive implementation of the algorithm is described and simulated, demonstrating the tracking capabilities of the algorithm in a dynamic scenario. It is observed that the DANSE$_K$ algorithm is more robust to estimation errors in the correlation matrices, compared to its centralized equivalent. In this paper, we have only considered the case where nodes update their parameters in a sequential round robin fashion. A modified DANSE$_K$ algorithm is studied in a companion paper [14], where an improved tracking performance is obtained, by letting nodes update simultaneously.

Acknowledgements

The authors would like to thank B. Cornelis and the anonymous reviewers for their valuable comments after proof-reading this paper.

Bibliography

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. 2033–2036 vol.4, 2001.

[2] 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, Aug. 2007.

[3] ——, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, July 2008.

[4] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[5] I. Schizas, G. Giannakis, and Z.-Q. Luo, "Distributed estimation using reduced-dimensionality sensor observations," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4284–4299, Aug. 2007.

[6] Z.-Q. Luo, G. Giannakis, and S. Zhang, "Optimal linear decentralized estimation in a bandwidth constrained sensor network," *Proc. International Symposium on Information Theory (ISIT)*, pp. 1441–1445, Sept. 2005.

[7] K. Zhang, X. Li, P. Zhang, and H. Li, "Optimal linear estimation fusion - part VI: sensor data compression," *Proc. Sixth International Conference of Information Fusion*, vol. 1, pp. 221–228, 2003.

[8] Y. Zhu, E. Song, J. Zhou, and Z. You, "Optimal dimensionality reduction of sensor data in multisensor estimation fusion," *IEEE Transactions on Signal Processing*, vol. 53, no. 5, pp. 1631–1639, May 2005.

[9] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435.

[10] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, pp. 38–51, Jan. 2009.

[11] T. Klasen, T. Van den Bogaert, M. Moonen, and J. Wouters, "Binaural noise reduction algorithms for hearing aids that preserve interaural time delay cues," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1579–1585, April 2007.

[12] S. Doclo, T. 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," *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC), Paris, France*, Sep. 2006.

[13] A. Bertrand and M. Moonen, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2053–2056, April 2009.

[14] ——, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part II: Simultaneous and asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5292 –5306, Oct. 2010.

[15] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.

[16] J. C. Bezdek and R. J. Hathaway, "Some notes on alternating optimization," in *Advances in Soft Computing*. Springer Berlin / Heidelberg, 2002, pp. 187–195.

[17] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

Chapter 3

Fully Connected DANSE with Simultaneous Node Updating

Distributed adaptive node-specific signal estimation in fully connected sensor networks – Part II: simultaneous & asynchronous node updating

Alexander Bertrand and Marc Moonen

Published in *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5292 - 5306, Oct. 2010.

Contributions of first author

- literature study
- co-development of the S-DANSE$_K$, rS-DANSE$_K$ and rS-DANSE$_K^+$ algorithms, and their asynchronous variants
- co-establishment of proof of convergence and optimality of rS-DANSE$_K^+$ and rA-DANSE$_K^+$
- design of simulation scenarios
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

In this paper, we revisit an earlier introduced distributed adaptive node-specific signal estimation (DANSE) algorithm that operates in fully connected sensor networks. In the original algorithm, the nodes update their parameters in a sequential round-robin fashion, which may yield a slow convergence of the estimators, especially so when the number of nodes in the network is large. When all nodes update simultaneously, the algorithm adapts more swiftly, but convergence can no longer be guaranteed. Simulations show that the algorithm then often gets locked in a suboptimal limit cycle. We first provide an extension to the DANSE algorithm, in which we apply an additional relaxation in the updating process. The new algorithm is then proven to converge to the optimal estimators when nodes update simultaneously or asynchronously, be it that the computational load at each node increases in comparison with the algorithm with sequential updates. Finally, based on simulations it is demonstrated that a simplified version of the new algorithm, without any extra computational load, can also provide convergence to the optimal estimators.

3.1 Introduction

A wireless sensor network [1] consists of multiple sensor nodes that are connected with each other through a wireless link, and where each sensor node has its own processing unit. It allows to collect spatially diversed observations of a certain physical process, and to process these observations in a distributed fashion. A general objective is to utilize all sensor signal observations available in the entire network to perform a certain task, such as the estimation of a parameter or signal (see for example [2–10]).

In [9, 10], we have introduced a distributed adaptive node-specific signal estimation (DANSE) algorithm that operates in a fully connected sensor network. The term 'node-specific' refers to the fact that each node estimates a different desired signal. The nodes in the network broadcast compressed versions of their sensor signal observations, yet the algorithm converges to the optimal node-specific linear MMSE estimators, as if all sensor signal observations were available at each node, assuming that the desired signals of the different nodes share a common latent signal subspace with small dimension.

In the DANSE algorithm, as introduced in [9, 10], nodes update in a sequential round-robin fashion. The algorithm typically converges in a small number of iterations (about two iterations per node). However, due to the sequential updating scheme, only one node at a time can estimate the statistics of its input signals and perform an update of its parameters. Since every such parameter update at a specific node changes the statistics of the node's broadcast signal,

it takes some time before the next node can collect enough data to compute a reliable estimate of the modified signal statistics and then update its parameters. As a result, even though the DANSE algorithm converges in a small number of iterations, it may converge slowly in time, especially so when the number of nodes is large.

If alternatively, nodes would perform their updates simultaneously, the algorithm can adapt more swiftly, and all nodes can then estimate the signal statistics in parallel. However, convergence can no longer be guaranteed in this case, as will be shown by simulations. We will therefore extend the DANSE algorithm with a relaxation operation. We prove that this new algorithm converges when nodes update simultaneously or asynchronously. With the latter, we refer to the case where each node decides independently when and how often it updates its parameters, possibly simultaneously with other nodes. This avoids the need for a network-wide updating protocol that coordinates the updates between the different nodes.

We will also present a simplified version of the new algorithm, which reduces the computational load at each node. Although a theoretical convergence proof is not available for this simplified version of the algorithm, in simulations it is indeed observed to converge to the same optimal estimators. The tracking capabilities of the presented algorithm are tested in a simulated dynamic scenario, showing that simultaneous node updating significantly improves the adaptation speed of the DANSE algorithm, while maintaining its optimality.

The paper is organized as follows. The problem formulation and notation are given in Section 3.2. In Section 3.3, we briefly review the DANSE algorithm of [9, 10]. In Section 3.4, we extend this algorithm with a relaxation operation to guarantee convergence and optimality when nodes update simultaneously or asynchronously, allowing for parallelization and uncoordinated computation. In Section 3.5 the convergence results are illustrated with numerical simulations in batch mode, and the tracking capabilities are demonstrated with an adaptive implementation of the algorithm. Conclusions are given in Section 3.6.

3.2 Problem Formulation and Notation

We consider an ideal fully connected network with sensor nodes $\{1, \ldots, J\} = \mathcal{J}$, in which data broadcast by a node can be captured by all other $(J-1)$ nodes in the network through an ideal link. Sensor node k collects observations of a complex[1] valued M_k-channel signal $\mathbf{y}_k[t]$, where $t \in \mathbb{N}$ is the discrete time index, and where $\mathbf{y}_k[t]$ is an M_k-dimensional column vector. Each channel

[1] Throughout this paper, all signals are assumed to be complex valued to permit frequency domain descriptions.

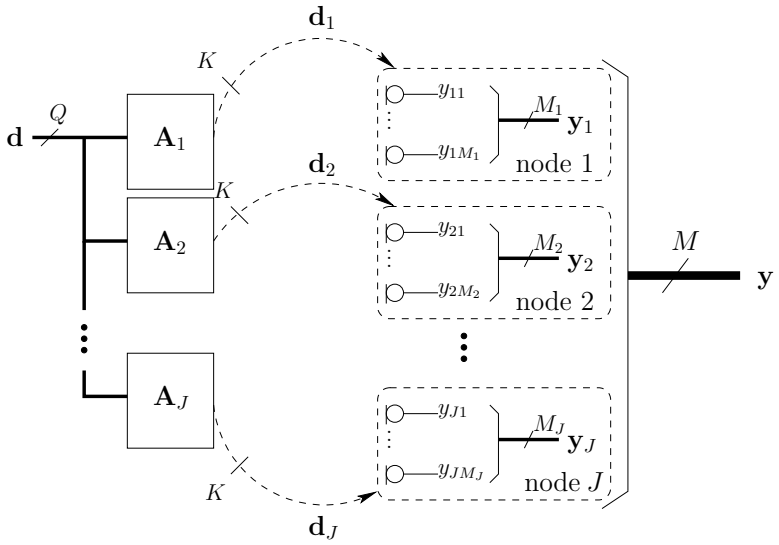


Figure 3.1: Description of the scenario. The network contains J sensor nodes, $k = 1 \dots J$, where node k collects M_k-channel sensor signal observations and estimates a node-specific desired signal $\mathbf{d}_k$, which is a mixture of the K channels of a common latent signal $\mathbf{d}$.

$y_{kn}[t]$, $\forall\, n \in \{1, \dots, M_k\}$, of the signal $\mathbf{y}_k[t]$ corresponds to a sensor signal to which node k has access. We assume that all signals are stationary[2] and ergodic. For the sake of an easy exposition, we will omit the time index when referring to a signal, and we will only write the time index when referring to one specific observation, i.e. $\mathbf{y}_k[t]$ is the observation of the signal $\mathbf{y}_k$ at time t. We define $\mathbf{y}$ as the M-channel signal in which all $\mathbf{y}_k$ are stacked, where $M = \sum_{k=1}^{J} M_k$. This scenario is depicted in Fig. 3.1.

We first consider the centralized estimation problem, i.e. we assume that each node has access to the observations of the entire M-channel signal $\mathbf{y}$. This corresponds to the case where nodes broadcast their uncompressed observations to all other nodes. In Sections 3.3 and 3.4, the general goal will be to compress the broadcast signals, while preserving the estimation performance of this centralized estimator. The objective for each node k is to optimally estimate a node-specific desired K-channel signal $\mathbf{d}_k$ that is correlated to $\mathbf{y}$. We consider the general case where $\mathbf{d}_k$ is not an observed signal, as it is the case in signal enhancement [11, 12]. As shown in Fig. 3.1, we assume that the node-specific desired signals $\mathbf{d}_k$ share a common Q-dimensional latent signal

[2]In practice, the stationarity and ergodicity assumption can be relaxed to short-term stationarity and ergodicity, in which case the theory should be applied to finite signal segments that are assumed to be stationary and ergodic.

subspace, defined by an unknown Q-channel latent signal $\mathbf{d}$, i.e.

$$\mathbf{d}_k = \mathbf{A}_k \mathbf{d}, \ \forall\, k \in \mathcal{J} \tag{3.1}$$

with $\mathbf{A}_k$ a full rank $K \times Q$ matrix with unknown coefficients. Without loss of generality, we assume that K is chosen equal to Q in the sequel. In many practical cases, only a subset of the channels of $\mathbf{d}_k$ may be of actual interest, in which case the other channels should be viewed as auxiliary channels to capture the entire Q-dimensional signal space spanned by all the desired signals of interest.

Node k uses a linear estimator $\mathbf{W}_k$ to estimate $\mathbf{d}_k$ as

$$\overline{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y} \tag{3.2}$$

where $\mathbf{W}_k$ is a complex $M \times K$ matrix, and where superscript H denotes the conjugate transpose operator. We consider linear MMSE estimation based on a node-specific estimator $\hat{\mathbf{W}}_k$, i.e.

$$\hat{\mathbf{W}}_k = \arg\min_{\mathbf{W}_k} E\left\{\|\mathbf{d}_k - \mathbf{W}_k^H \mathbf{y}\|^2\right\} \ , \tag{3.3}$$

where $E\{.\}$ denotes the expected value operator. The objective is to solve all J node-specific MMSE problems (3.3), i.e. one for each node. Assuming that the correlation matrix $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$ has full rank, the solution of (3.3) is

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^{-1} \mathbf{R}_{yd_k} \tag{3.4}$$

with $\mathbf{R}_{yd_k} = E\{\mathbf{y}\mathbf{d}_k^H\}$ [13]. Based on the assumed ergodicity, $\mathbf{R}_{yy}$ and $\mathbf{R}_{yd_k}$ can be estimated by time averaging. The $\mathbf{R}_{yy}$ is directly estimated from the sensor signal observations. A possible way to estimate $\mathbf{R}_{yd_k}$, where $\mathbf{d}_k$ is not an observed signal, is to use periodic training sequences, or by exploiting the on-off behavior of the desired signal. The latter is often used in speech enhancement applications since speech signals contain pauses in between words and sentences [11, 12, 14]. In the sequel, we will assume that $\mathbf{R}_{yd_k}$ can be estimated during operation of the algorithm. Some example strategies to estimate $\mathbf{R}_{yd_k}$ can be found in [10].

Notice that each node k only has access to observations of $\mathbf{y}_k$ which is a subset of the channels of the full signal $\mathbf{y}$. Therefore, to find the optimal MMSE solution (3.4) in each node, the observations of $\mathbf{y}_k$ in principle have to be communicated to all nodes in the network, which requires a large communication bandwidth. However, the DANSE$_K$ algorithm, reviewed in the next section, significantly compresses this communication bandwidth, yet still achieves the optimal linear MMSE estimator (3.4) at each node.

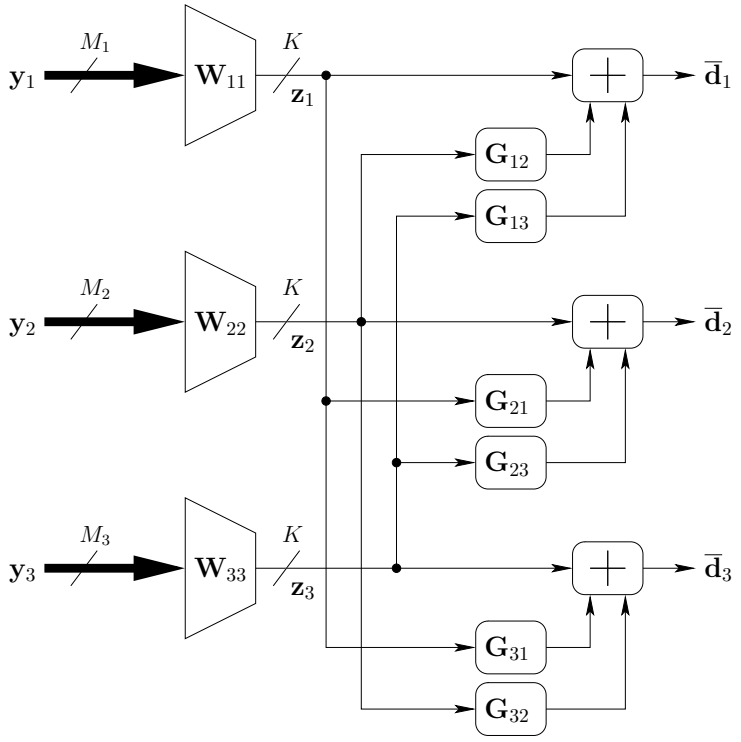


Figure 3.2: The DANSE$_K$ scheme with 3 nodes ($J = 3$). Each node k estimates a signal $\mathbf{d}_k$ using its own M_k-channel sensor signal observations, and 2 K-channel signals broadcast by the other two nodes.

3.3 The DANSE$_K$ Algorithm

In this section, we briefly review the DANSE$_K$ algorithm. For a more detailed description and analysis, we refer to [10]. In DANSE$_K$, the sensor nodes broadcast K linear combinations of their M_k-channel sensor signal observations. The DANSE$_K$ algorithm thus yields a compression with a factor of $\frac{M_k}{K}$ for the broadcasting by node k.

We define a partitioning of the estimator $\mathbf{W}_k$ as $\mathbf{W}_k = [\mathbf{W}_{k1}^T \ \dots \ \mathbf{W}_{kJ}^T]^T$ with $\mathbf{W}_{kq}$ denoting the $M_k \times K$ submatrix of $\mathbf{W}_k$ that is applied to $\mathbf{y}_q$, and where superscript T denotes the transpose operator. In this way, (3.3) is equivalent to

$$\hat{\mathbf{W}}_k = \begin{bmatrix} \hat{\mathbf{W}}_{k1} \\ \vdots \\ \hat{\mathbf{W}}_{kJ} \end{bmatrix} = \underset{\{\mathbf{W}_{k1},\dots,\mathbf{W}_{kJ}\}}{\arg\min} E\left\{ \|\mathbf{d}_k - \sum_{q=1}^{J} \mathbf{W}_{kq}^H \mathbf{y}_q\|^2 \right\} . \tag{3.5}$$

In the DANSE$_K$ algorithm, each node k broadcasts observations of the K-channel compressed signal $\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k$ to the other nodes. Notice that $\mathbf{W}_{kk}$ thus both acts as a compressor and a part of the estimator $\mathbf{W}_k$. A node k can transform the K-channel signal $\mathbf{z}_q$ that it receives from another node q by a $K \times K$ transformation matrix $\mathbf{G}_{kq}$. Notice that, as $\mathbf{G}_{kq}$ is a square matrix, no decompression is involved. The parametrization of the $\mathbf{W}_k$ effectively applied at node k is therefore

$$\widetilde{\mathbf{W}}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{kJ} \end{bmatrix} . \tag{3.6}$$

We use a tilde to indicate that the estimator is parametrized according to (3.6). In this parametrization, node k can only control the parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_k = [\mathbf{G}_{k1}^T \ \dots \ \mathbf{G}_{kJ}^T]^T$. To remove the ambiguity in $\mathbf{W}_{kk}\mathbf{G}_{kk}$, we assume that $\mathbf{G}_{kk} = \mathbf{I}_K$ with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix. A schematic illustration of this scheme in a 3-node network ($J = 3$), is shown in Fig. 3.2. The goal of the DANSE$_K$ algorithm is to iteratively update the parameters of (3.6) until $\left(\widetilde{\mathbf{W}}_1, \dots, \widetilde{\mathbf{W}}_J\right) = \left(\hat{\mathbf{W}}_1, \dots, \hat{\mathbf{W}}_J\right)$.

In the sequel, we will use the following notation and definitions. In general, we will use X^i to denote X at iteration i, where X can be a signal or a parameter. We let $\mathbf{z} = \left[\mathbf{z}_1^T \ \dots \ \mathbf{z}_J^T\right]^T$, and we use the notation $\mathbf{z}_{-k} = \left[\mathbf{z}_1^T \dots \mathbf{z}_{k-1}^T \mathbf{z}_{k+1}^T \dots \mathbf{z}_J^T\right]^T$, i.e. the vector $\mathbf{z}$ without the subvector $\mathbf{z}_k$. Similarly, we will use $\mathbf{G}_{k-q}$ to denote the matrix $\mathbf{G}_k$ without $\mathbf{G}_{kq}$.

The DANSE$_K$ algorithm will iteratively update the parameters in (3.6) by solving local MMSE problems at each node $k \in \mathcal{J}$. For node k, at iteration i, this update is computed as

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \underset{\mathbf{W}_{kk}, \mathbf{G}_{k-k}}{\arg\min} \ E\left\{ \left\| \mathbf{d}_k - \begin{bmatrix} \mathbf{W}_{kk}^H \mathbf{G}_{k-k}^H \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix} \right\|^2 \right\} . \tag{3.7}$$

In this local MMSE problem, the node-specific desired signal $\mathbf{d}_k$ is estimated by means of the input signals of node k, i.e. its M_k-channel sensor signal $\mathbf{y}_k$ and the $K(J-1)$-channel signal $\mathbf{z}_{-k}$ containing the broadcast signals of the other nodes. Let $\widetilde{\mathbf{y}}_k^i$ denote the stacked version of these local input signals at node k, i.e.

$$\widetilde{\mathbf{y}}_k^i = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix} . \tag{3.8}$$

Then the solution of (3.7) is

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \left(\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i\right)^{-1} \mathbf{R}_{\tilde{y}_k d_k}^i \tag{3.9}$$

with

$$\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k} = E\{\widetilde{\mathbf{y}}^i_k \widetilde{\mathbf{y}}^{i\,H}_k\} \tag{3.10}$$

$$\mathbf{R}^i_{\tilde{y}_k d_k} = E\{\widetilde{\mathbf{y}}^i_k \mathbf{d}^H_k\}\ . \tag{3.11}$$

We define a block size B which denotes the number of observations that the nodes collect in between two successive node updates, i.e. in between two increments of i. The DANSE$_K$ algorithm is described in Table 3.1 on the next page.

Remark I: Notice that the different iterations are spread out over time. Therefore, the iterative characteristics of the algorithm do not have an impact on the amount of data that is transmitted, i.e. each sample is only broadcast once since the time index in (3.12) and (3.14) shifts together with the iteration index. Therefore, an update of the estimator parameters only has an impact on future samples and old samples are not re-estimated.

Remark II: For implementation aspects regarding the estimation of the correlation matrices $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ and $\mathbf{R}^i_{\tilde{y}_k d_k}$, we refer to [10].

It is noted that the nodes update in a sequential round robin fashion. At each iteration, one specific node estimates the signal statistics of its input channels, and updates its parameters by optimizing its local node-specific estimation problem (3.7), while the parameters at the other nodes are freezed. The following theorem guarantees convergence of the DANSE$_K$ algorithm to the optimal estimators[3]:

Theorem 3.1 *If the sensor signal correlation matrix $\mathbf{R}_{yy}$ has full rank, and if (3.1) is satisfied with $K = Q$, then for the DANSE$_K$ algorithm as described above, the sequence $\left(\widetilde{\mathbf{W}}^i_k\right)_{i\in\mathbb{N}}$ converges to the optimal solution (3.4), $\forall\, k \in \mathcal{J}$, for any initialization of the parameters.*

Proof : See [10]. □

3.4 Simultaneous and Uncoordinated Updating

As mentioned in Section 3.3, the nodes in the DANSE$_K$ algorithm update their parameters in a sequential round-robin fashion. When node k performs an

[3]Theorem 3.1, and all convergence theorems in the sequel, assume that the signal statistics are perfectly known by the algorithm, i.e. as if an infinite observation window is used. In other words, estimation errors in $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ and $\mathbf{R}^i_{\tilde{y}_k d_k}$ are not taken into account.

The DANSE$_K$ Algorithm

1. Initialize: $i \leftarrow 0$, $u \leftarrow 1$
   Initialize $\mathbf{W}_{kk}^0$ and $\mathbf{G}_{k-k}^0$ with random matrices, $\forall\, k \in \mathcal{J}$.
2. Each node $k \in \mathcal{J}$ performs the following operation cycle:
   (a) Collect the sensor observations $\mathbf{y}_k[iB+n]$, $n = 0 \dots B-1$.
   (b) Compress these M_k-dimensional observations to K-dimensional vectors

$$\mathbf{z}_k^i[iB+n] = \mathbf{W}_{kk}^{i\,H}\mathbf{y}_k[iB+n],\ \ n = 0 \dots B-1\ . \quad (3.12)$$

   (c) Broadcast the compressed observations $\mathbf{z}_k^i[iB+n]$, $n = 0 \dots B-1$, to the other nodes.
   (d) Collect the $K(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \dots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   (e) Update the estimates of $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$ and $\mathbf{R}_{\tilde{y}_k d_k}^i$, by including the newly collected data.
   (f) Update the node-specific parameters:

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \begin{cases} \left(\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i\right)^{-1}\mathbf{R}_{\tilde{y}_k d_k}^i & \text{if } k = u \\ \begin{bmatrix} \mathbf{W}_{kk}^i \\ \mathbf{G}_{k-k}^i \end{bmatrix} & \text{if } k \neq u \end{cases} \quad (3.13)$$

   (g) Compute the estimate of $\mathbf{d}_k[iB+n]$ as

$$\begin{aligned} \overline{\mathbf{d}}_k[iB+n] =& \mathbf{W}_{kk}^{i+1\,H}\mathbf{y}_k[iB+n] \\ &+ \mathbf{G}_{k-k}^{i+1\,H}\mathbf{z}_{-k}^i[iB+n]\ . \end{aligned} \quad (3.14)$$

3. $i \leftarrow i+1$
4. $u \leftarrow (u \mod J)+1$
5. return to step 2

Table 3.1: The DANSE$_K$ algorithm

update at iteration i, the signal $\mathbf{z}_k^i$ changes to $\mathbf{z}_k^{i+1}$, which generally has different statistics. Therefore, the next node to perform an update needs sufficient time to collect a sufficient number of observations of $\mathbf{z}_k^{i+1}$ to reliably estimate the correlation coefficients involving this signal. Therefore, even though the DANSE$_K$ algorithm converges fast in terms of number of iterations, it may converge rather slowly in time, which affects its tracking performance.

This problem obviously becomes worse when the number of nodes is large, such that one round of updates takes a long time. We may expect that the network can react much faster to changes in the environment when nodes would be able to update simultaneously. Simulations in Section 3.5 will show that convergence is indeed faster in this case, both in time and in number of iterations. Unfortunately, these simulations will also show that convergence is no longer guaranteed. Therefore, in this section, we first extend the DANSE$_K$ algorithm, to restore convergence, and then consider a number of algorithmic simplifications.

3.4.1 The S-DANSE$_K$ Algorithm

Consider the case where all nodes update simultaneously, i.e. (3.9) is applied for all k simultaneously in iteration i. We refer to this as simultaneous-DANSE$_K$ or S-DANSE$_K$. In a network with two nodes ($J = 2$), convergence of DANSE$_K$ under sequential updating also implies convergence[4] of S-DANSE$_K$. Unfortunately, this is not always true if $J > 2$. Extensive simulations show that the S-DANSE$_K$ algorithm does not always converge to a stable estimator, but may get locked in a suboptimal limit cycle (see Fig. 3.5). This means that the parameters at the different nodes keep switching between multiple suboptimal estimators. The occurrence of these limit cycles heavily depends on the scenario, but a clear rule to predict whether the S-DANSE$_K$ algorithm will converge to a stable estimator or get locked in a limit cycle, has not been found. In the white noise experiments described in Section 3.5.1, the S-DANSE$_K$ algorithm mostly converges to the optimal solution. However, in the scenario of Section 3.5.2, and in the simulations in acoustic sensor networks described in [12], limit cycle behavior has been observed quite frequently.

The reason why S-DANSE$_K$ often fails to converge can be intuitively explained as follows. Assume that node k computes (3.9), i.e. it optimizes its estimators with respect to the current statistics of the broadcast signals in $\mathbf{z}_{-k}^i$. If the other nodes update simultaneously, all the signals in $\mathbf{z}_{-k}^i$ immediately switch to $\mathbf{z}_{-k}^{i+1}$. Since the newly applied estimator $\widetilde{\mathbf{W}}_k^{i+1}$ was optimized with respect to the signal statistics of $\mathbf{z}_{-k}^i$, it immediately becomes suboptimal again due to simultaneous updates of the other nodes. In fact, the MSE of the node-specific

[4] This can be shown by similar arguments as in [15], where it is explained that convergence of Gauss-Seidel iteration implies convergence of Jacobi iteration in the 2-node case.

estimators may therefore increase after the update, i.e. the new estimator $\overline{\mathbf{d}}_k^{\,i+1} = \mathbf{W}_k^{i+1\,H}\mathbf{y}$ for node k may be worse than the old estimator $\overline{\mathbf{d}}_k^{\,i} = \mathbf{W}_k^{i\,H}\mathbf{y}$ before the update.

This fundamental difference between the convergence properties of DANSE$_K$ and S-DANSE$_K$ is similar to the difference between the convergence properties of non-linear Gauss-Seidel iteration and non-linear Jacobi iteration, as described in [16]. In both the Gauss-Seidel and Jacobi procedures, subsequent optimization steps are performed over a subset of the variables of an objective function, while keeping the other variables fixed. However, both methods differ in the way they update the new iteration point. In Gauss-Seidel iteration, the iteration point is immediately updated after the optimization of a single subset of variables, whereas in Jacobi iteration, the actual iteration point is updated after an optimization of all the subsets simultaneously with respect to the current iteration point. In particular, for a cost function $f(\mathbf{w})$ with $\mathbf{w} = (w_1, w_2)$, the non-linear Gauss-Seidel and Jacobi procedure are as follows (both are illustrated in Fig. 3.3):

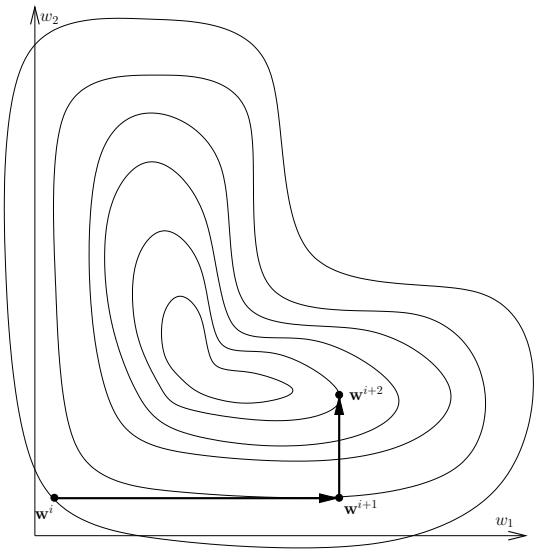
Non-linear Gauss-Seidel:

1. Initialize $\mathbf{w}^0 = (w_1^0, w_2^0)$ randomly.
2. $i \leftarrow 0$.
3. $\begin{cases} w_1^{i+1} \leftarrow \arg\min_{w_1} f(w_1, w_2^i) \\ w_2^{i+1} \leftarrow w_2^i \end{cases}$
   $\begin{cases} w_1^{i+2} \leftarrow w_1^{i+1} \\ w_2^{i+2} \leftarrow \arg\min_{w_2} f(w_1^{i+1}, w_2) \end{cases}$
4. $i \leftarrow i + 2$
5. Return to step 3.

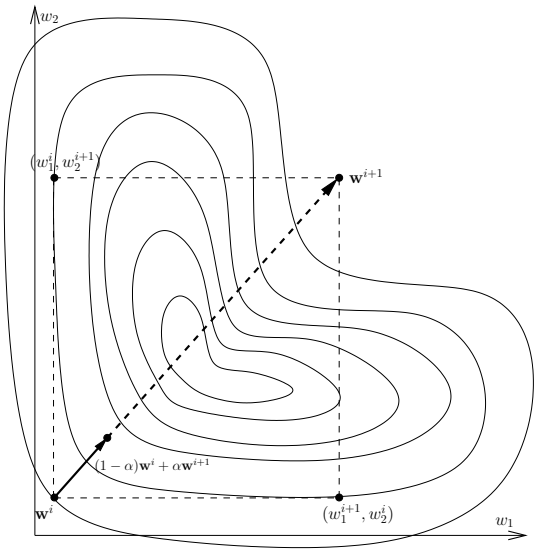
Non-linear Jacobi:

1. Initialize $\mathbf{w}^0 = (w_1^0, w_2^0)$ randomly.
2. $i \leftarrow 0$.
3. $\begin{cases} w_1^{i+1} \leftarrow \arg\min_{w_1} f(w_1, w_2^i) \\ w_2^{i+1} \leftarrow \arg\min_{w_2} f(w_1^i, w_2) \end{cases}$
4. $i \leftarrow i + 1$
5. Return to step 3.

It is obvious that the Gauss-Seidel procedure will always result in a decrease of the objective function in each iteration, i.e. $f(\mathbf{w}^{i+1}) \leq f(\mathbf{w}^i)$, and therefore convergence is generally not an issue. In the Jacobi procedure however, this argument fails since an update can result in $f(\mathbf{w}^{i+1}) > f(\mathbf{w}^i)$, as illustrated in Fig. 3.3(b). This is because each variable is optimized with respect to the previous value of the other variables, which are not retained after the update.



(a) non-linear Gauss-Seidel iteration



(b) non-linear Jacobi iteration and its relaxed version

Figure 3.3: Illustration of non-linear Gauss-Seidel iteration and non-linear Jacobi iteration.

Therefore, Jacobi iteration often fails to converge[5].

The DANSE$_K$ algorithm sequentially optimizes a specific element of $\{\mathbf{W}_{11}, \ldots, \mathbf{W}_{JJ}\}$ with respect to the current values of the other elements (corresponding to the other nodes). Notice that this is akin to Gauss-Seidel iteration. From the same point of view, the S-DANSE$_K$ algorithm is akin to Jacobi iteration since the elements in $\{\mathbf{W}_{11}, \ldots, \mathbf{W}_{JJ}\}$ are again optimized with respect to the current values of the other elements, but all of them are updated simultaneously. This relationship illustrates why S-DANSE$_K$ often fails to converge whereas DANSE$_K$ always converges to the optimal estimators. However, keep in mind that the similarity between DANSE$_K$ and S-DANSE$_K$ on the one hand and the Gauss-Seidel and Jacobi procedure on the other hand, only holds up to a certain level. Indeed, in DANSE$_K$ and S-DANSE$_K$ none of the variables are actually fixed. If node k optimizes $\mathbf{W}_{kk}$, it can also partially manipulate the $\mathbf{W}_{qq}$ of other nodes $q \neq k$ by means of the variable $\mathbf{G}_{kq}$. Furthermore, the objective function is different at each node, and therefore each element in $\{\mathbf{W}_{11}, \ldots, \mathbf{W}_{JJ}\}$ is optimized with respect to a different cost function.

3.4.2 The rS-DANSE$_K^+$ Algorithm

In the previous subsection, we explained that the S-DANSE$_K$ algorithm often fails to converge since each node optimizes its estimators with respect to signal statistics that immediately become invalid after the update. The idea is now to partially counter these dynamics by letting each node k perform a relaxed update of its partial estimator $\mathbf{W}_{kk}$ to a convex combination of $\mathbf{W}_{kk}^i$ and $\mathbf{W}_{kk}^{i+1}$, i.e. $(1-\alpha)\mathbf{W}_{kk}^i + \alpha\mathbf{W}_{kk}^{i+1}$ with $\alpha \in (0, 1]$. This allows the old estimator to remain partially active and to generate broadcast signals $\mathbf{z}_{-k}^{i+1}$ that have partial components equal to the old broadcast signals $\mathbf{z}_{-k}^i$. This corresponds to some first order memory in the updating dynamics[6], which is absent in the S-DANSE$_K$ algorithm. The intuition behind this relaxation procedure is illustrated in Fig. 3.3(b), where it is observed that relaxation dampens the Jacobi-iteration. If α is chosen small enough, the objective function decreases due to the relaxed update.

In this section, we will modify the S-DANSE$_K$ algorithm accordingly, to enforce convergence when nodes update simultaneously. Although this procedure may appear to be quite heuristic at first sight, it can be theoretically justified, i.e. we will prove that the new algorithm enforces convergence if the relaxation

[5] In the illustration of Fig. 3.3, the Jacobi iteration eventually will converge since the variables of the cost function are divided into two subsets ($\{w_1\}$ and $\{w_2\}$). Jacobi iteration always converges in this case [15]. However, this is generally not true when more than 2 variable subsets are used [16].

[6] The same technique is sometimes used in game theory to enforce convergence to a Nash equilibrium when all players simultaneously perform a best-reply to the current strategy setting (see e.g. [15, 17]). However, this technique only works for specific games that satisfy certain assumptions, which are mostly very technical and stringent.

stepsize satisfies certain properties.

Before describing the algorithm, we introduce some additional notation. The matrix $\mathbf{W}$ (without subscript) denotes the stacked matrix of all $\mathbf{W}_{kk}$ matrices, i.e.

$$\mathbf{W} = \left[\begin{array}{cccc} \mathbf{W}_{11}^T & \mathbf{W}_{22}^T & \dots & \mathbf{W}_{JJ}^T \end{array} \right]^T . \tag{3.15}$$

We also define the MSE cost functions corresponding to node k, namely

$$J_k(\mathbf{W}_k) = E\left\{ \|\mathbf{d}_k - \mathbf{W}_k^H \mathbf{y}\|^2 \right\} \tag{3.16}$$

as used in (3.3), and

$$\widetilde{J}_k\left(\mathbf{W}, \mathbf{G}_k\right) = J_k(\widetilde{\mathbf{W}}_k) \tag{3.17}$$

where $\widetilde{\mathbf{W}}_k$ is defined from $\mathbf{W}$ and $\mathbf{G}_k$ as in (3.6). Notice that $\mathbf{G}_k$ contains the entry $\mathbf{G}_{kk}$, which is never actually computed in the original DANSE$_K$ algorithm. We define $F_k(\mathbf{W}^i)$ as the function that generates $\mathbf{W}_{kk}^{i+1}$ according to (3.9), i.e.

$$F_k(\mathbf{W}^i) = \mathbf{W}_{kk}^{i+1} \tag{3.18}$$

$$= \left[\begin{array}{c|c} \mathbf{I}_{M_k} & \mathbf{O}_{M_k \times K(J-1)} \end{array} \right] \left(\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i\right)^{-1} \mathbf{R}_{\tilde{y}_k d_k}^i \tag{3.19}$$

with $\mathbf{O}_{P\times Q}$ denoting an all-zero $P \times Q$ matrix. It is noted that the right-hand side of (3.19) depends on all entries of the argument $\mathbf{W}^i$ through the signal $\mathbf{z}_{-k}^i$, which is not explicitly revealed in this expression. We also define the correlation matrices: $\mathbf{R}_{zz}^i = E\{\mathbf{z}^i \mathbf{z}^{i\,H}\}$ and $\mathbf{R}_{zd_k}^i = E\{\mathbf{z}^i \mathbf{d}_k^H\}$.

Now consider the algorithm algorithm described in Table 3.2 on the next page. We will refer to this algorithm as relaxed simultaneous-DANSE$_K^+$ or rS-DANSE$_K^+$, where the superscript + is added because of the extra optimization (3.21). Notice that in rS-DANSE$_K^+$, the variable $\mathbf{G}_{kk}^{i+1}$ is indeed explicitly computed, unlike in the DANSE$_K$ algorithm with sequential updating. However, it is merely applied as a transformation for $\mathbf{W}_{kk}^i$ before the relaxed update (3.23). After the update (3.23), the $\mathbf{G}_{kk}$ that is actually applied to $\mathbf{W}_{kk}$ in the parametrization (3.6) is again fixed to an identity matrix, i.e. the schematic representation shown in Fig. 3.2, in which the $\mathbf{G}_{kk}$'s are omitted, also holds for the rS-DANSE$_K^+$ algorithm.

Due to the strict convexity of the cost function $J_k(\mathbf{W}_k)$, the simultaneous update function $F(\mathbf{W}) = \left[F_1(\mathbf{W})^T \dots F_J(\mathbf{W})^T\right]^T$ has only one fixed point satisfying $\mathbf{W} = F(\mathbf{W})$, namely the optimal solution $\hat{\mathbf{W}}$ to which DANSE$_K$ converges if sequential updating is used. Notice that for any α^i, this point $\hat{\mathbf{W}}$ is also the equilibrium point of (3.21)-(3.23). The choice of α^i is critical and decides whether or not (3.21)-(3.23) converges to this fixed point. The following theorem presents a strategy for choosing the stepsize parameter α^i, that guarantees convergence to the optimal solution:

The rS-DANSE$_K^+$ Algorithm

1. Initialize: $i \leftarrow 0$
   Initialize $\mathbf{W}_{kk}^0$ and $\mathbf{G}_{k-k}^0$ with random matrices, $\forall\, k \in \mathcal{J}$.
2. Each node $k \in \mathcal{J}$ performs the following operation cycle simultaneously:
   (a) Collect the sensor observations $\mathbf{y}_k[iB+n]$, $n = 0 \dots B-1$.
   (b) Compress these M_k-dimensional observations to K-dimensional vectors

$$\mathbf{z}_k^i[iB+n] = \mathbf{W}_{kk}^{i\,H}\mathbf{y}_k[iB+n],\ \ n = 0 \dots B-1\ . \tag{3.20}$$

   (c) Broadcast the compressed observations $\mathbf{z}_k^i[iB+n]$, $n = 0 \dots B-1$, to the other nodes.
   (d) Collect the $K(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \dots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   (e) Update the estimates of $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$, $\mathbf{R}_{zz}^i$, $\mathbf{R}_{\tilde{y}_k d_k}^i$, and $\mathbf{R}_{z d_k}^i$ by including the newly collected data.
   (f) Compute a new estimate for $\mathbf{G}_k^i$:

$$\mathbf{G}_k^{i+1} = \arg\min_{\mathbf{G}_k} \widetilde{J}_k\left(\mathbf{W}^i, \mathbf{G}_k\right) \tag{3.21}$$

$$= \left(\mathbf{R}_{zz}^i\right)^{-1}\mathbf{R}_{z d_k}^i \tag{3.22}$$

   (g) Choose an $\alpha^i \in (0,1]$, and compute a new estimate for $\mathbf{W}_{kk}^i$:

$$\mathbf{W}_{kk}^{i+1} = (1-\alpha^i)\mathbf{W}_{kk}^i\mathbf{G}_{kk}^{i+1} + \alpha^i F_k(\mathbf{W}^i)\ . \tag{3.23}$$

   (h) Compute the estimate of $\mathbf{d}_k[iB+n]$ as

$$\begin{aligned}\overline{\mathbf{d}}_k[iB+n] =& \mathbf{W}_{kk}^{i+1\,H}\mathbf{y}_k[iB+n] \\ &+ \mathbf{G}_{k-k}^{i+1\,H}\mathbf{z}_{-k}^i[iB+n]\ .\end{aligned} \tag{3.24}$$

3. $i \leftarrow i+1$
4. return to step 2

Table 3.2: The rS-DANSE$_K^+$ algorithm

Theorem 3.2 *If the sensor signal correlation matrix* $\mathbf{R}_{yy}$ *has full rank, and if (3.1) is satisfied with* $K = Q$*, then for the rS-DANSE*$_K^+$ *algorithm as described above, with stepsizes* α^i *satisfying*

$$\alpha^i \in (0, 1] \tag{3.25}$$

$$\lim_{i\to\infty} \alpha^i = 0 \tag{3.26}$$

$$\sum_{i=0}^{\infty} \alpha^i = \infty \,, \tag{3.27}$$

the sequence $\left(\widetilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ *converges to the optimal solution (3.4),* $\forall\, k \in \mathcal{J}$*, for any initialization of the parameters.*

Proof : See Appendix 3.A. □

A possible choice for the sequence $\left(\alpha^i\right)_{i\in\mathbb{N}}$ is $\alpha^i = \frac{1}{i}$ or a slower decreasing sequence, e.g. $\alpha^i = \frac{1}{\log_{10}(10+i)}$. The conditions on the sequence $\left(\alpha^i\right)_{i\in\mathbb{N}}$ in Theorem 3.2 are however quite conservative. Extensive simulations indicate that there appears to exist a critical $\hat{\alpha}$ and a corresponding i^* such that rS-DANSE$_K^+$ converges as long as $\alpha^i < \hat{\alpha}$, $\forall i > i^*$. Since this critical $\hat{\alpha}$ is generally not known a priori, a good strategy consists in initially choosing $\alpha^i = 1$, i.e. no relaxation, in combination with a limit cycle detector. Only when a limit cycle is detected, α^i is decreased until the limit cycle disappears. The algorithm then converges with fixed α^i to an equilibrium point of F, i.e. the matrix $\mathbf{W}$ that corresponds to solution (3.4). It is noted that, even if the above mentioned critical $\hat{\alpha}$ does not exist, this procedure automatically guarantees that the conditions (3.25)-(3.27) on the stepsize α^i are satisfied, and therefore the algorithm will converge under all circumstances.

3.4.3 The rS-DANSE$_K$ Algorithm

Update rule (3.21) in rS-DANSE$_K^+$ increases the computational load in every node, since it represents an extra MSE minimization in addition to the implicit MSE minimization in $F_k(\mathbf{W}^i)$. However, extensive simulations indicate that this extra MSE minimization is not crucial to enforce convergence. The $\mathbf{G}_{k-k}^{i+1}$ defined in (3.9) that are generated as a by-product in the evaluation of $F_k(\mathbf{W}^i)$, may be used instead. We will refer to this modification as rS-DANSE$_K$, without the superscript +. The rS-DANSE$_K$ algorithm is described in Table 3.3 on the next page. Even in cases where S-DANSE$_K$ results in a suboptimal limit cycle, the rS-DANSE$_K$ algorithm is observed to converge to the optimal solution (3.4) under similar conditions as the rS-DANSE$_K^+$ algorithm, i.e. if α^i satisfies (3.25)-(3.27), or if it becomes smaller than a critical value. This

The rS-DANSE$_K$ Algorithm

1. Initialize: $i \leftarrow 0$
   Initialize $\mathbf{W}_{kk}^0$ and $\mathbf{G}_{k-k}^0$ with random matrices, $\forall\, k \in \mathcal{J}$.
2. Each node $k \in \mathcal{J}$ performs the following operation cycle simultaneously:
   (a) Collect the sensor observations $\mathbf{y}_k[iB+n]$, $n = 0 \dots B-1$.
   (b) Compress these M_k-dimensional observations to K-dimensional vectors

$$\mathbf{z}_k^i[iB+n] = \mathbf{W}_{kk}^{i\,H}\mathbf{y}_k[iB+n],\ \ n = 0 \dots B-1\ . \tag{3.28}$$

   (c) Broadcast the compressed observations $\mathbf{z}_k^i[iB+n]$, $n = 0 \dots B-1$, to the other nodes.
   (d) Collect the $K(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \dots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   (e) Update the estimates of $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$ and $\mathbf{R}_{\tilde{y}_k d_k}^i$ by including the newly collected data.
   (f) Compute

$$\begin{bmatrix} \mathbf{W}_{kk}^{\text{temp}} \\ \hline \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \left(\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i\right)^{-1}\mathbf{R}_{\tilde{y}_k d_k}^i\ . \tag{3.29}$$

   (g) Choose an $\alpha^i \in (0,1]$, and compute a new estimate for $\mathbf{W}_{kk}^i$:

$$\mathbf{W}_{kk}^{i+1} = (1-\alpha^i)\mathbf{W}_{kk}^i + \alpha^i\mathbf{W}_{kk}^{\text{temp}}\ . \tag{3.30}$$

   (h) Compute the estimate of $\mathbf{d}_k[iB+n]$ as

$$\begin{aligned}\overline{\mathbf{d}}_k[iB+n] =& \mathbf{W}_{kk}^{i+1\,H}\mathbf{y}_k[iB+n] \\ &+ \mathbf{G}_{k-k}^{i+1\,H}\mathbf{z}_{-k}^i[iB+n]\ .\end{aligned} \tag{3.31}$$

3. $i \leftarrow i+1$
4. return to step 2

Table 3.3: The rS-DANSE$_K$ algorithm

is stated here as an observation based on extensive simulation (see Section 3.5, and [12]), but without a formal proof[7]. Notice that the extra optimization (3.21) generally speeds up the convergence, especially so when α^i is small, which makes rS-DANSE$_K^+$ faster than rS-DANSE$_K$ (see Section 3.5). However, this faster convergence is mostly insignificant compared to the increase in convergence speed that is obtained by letting nodes update simultaneously instead of sequentially. Therefore, it may be desirable to use rS-DANSE$_K$ instead of rS-DANSE$_K^+$, to decrease the computational load at each node.

3.4.4 Asynchronous Updating

The DANSE$_K$ algorithm and its variations described in the previous sections, all imply the need for a network wide update protocol that coordinates the updating of nodes in the network. Here we consider the case in which the updating happens in an asynchronous[8] fashion, i.e. nodes can decide independently when and how often they update their parameters and nodes do not know when the other nodes perform an update. This can be viewed as the hybrid case between simultaneous updating and sequential round-robin updating.

The iterations in which node k updates its parameters is given by a binary sequence $\left(s_k^i\right)_{i\in\mathbb{N}}$, with $s_k^i \in \{0,1\}$, where $s_k^i = 1$ implies that node k performs an update at iteration i and where $s_k^i = 0$ implies that node k does not perform an update at iteration i. We will assume that

$$\forall\, k \in \mathcal{J} : \sum_{i=0}^{\infty} s_k^i = \infty \tag{3.32}$$

which means that none of the nodes permanently stops the updating process of its parameters. It is noted that the sequences $\left(s_k^i\right)_{i\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$, can be totally random and they are not known to the other nodes $q \neq k$. Therefore, the updates of each node can happen ex tempore, i.e. each node can decide on the fly when and how often it performs an update of its parameters. By setting the block size to $B = 1$, the iteration index i coincides with the time index t. This models the case where nodes are allowed to perform an update at any[9] sampling time t, i.e. fully asynchronously.

The relaxed asynchronous-DANSE$_K^+$ or rA-DANSE$_K^+$ algorithm is defined equiv-

[7] Due to a subtle difference, some parts of the proof of Theorem 3.2 are not applicable to the case of rS-DANSE$_K$. However, the main idea behind relaxation, as illustrated in Fig. 3.3(b), remains.

[8] The term 'asynchronous' here refers to the fact that there is no synchronization at the iteration level. However, at the sample level, an accurate synchronization of the sample clocks at the different nodes is still required.

[9] It is noted that setting $B = 1$ does not imply that nodes update at *every* sampling time t. As mentioned earlier, it is usually better to leave some time between successive updates of a certain node, i.e. using sparse sequences $\left(s_k^i\right)_{i\in\mathbb{N}}$.

alently to rS-DANSE^+_K, but the update (3.22)-(3.23) is replaced with

$$\mathbf{G}_k^{i+1} = \begin{cases} \left(\mathbf{R}_{zz}^i\right)^{-1}\mathbf{R}_{zd_k}^i & \text{if} \quad s_k^i = 1 \\ \mathbf{G}_k^i & \text{if} \quad s_k^i = 0 \end{cases} \tag{3.33}$$

$$\mathbf{W}_{kk}^{i+1} = \begin{cases} (1-\alpha^i)\mathbf{W}_{kk}^i\mathbf{G}_{kk}^{i+1} + \alpha^i F_k(\mathbf{W}^i) & \text{if} \quad s_k^i = 1 \\ \mathbf{W}_{kk}^i & \text{if} \quad s_k^i = 0 \end{cases} \tag{3.34}$$

Theorem 3.3 *If the sensor signal correlation matrix $\mathbf{R}_{yy}$ has full rank, and if (3.1) is satisfied with $K = Q$, then for the rA-DANSE^+_K algorithm as described above, with stepsizes α^i satisfying (3.25)-(3.27) and sequences $\left(s_k^i\right)_{i\in\mathbb{N}}$ satisfying (3.32), the sequence $\left(\widetilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ converges to the optimal solution (3.4), $\forall\, k \in \mathcal{J}$, for any initialization of the parameters.*

Proof : The proof is a straightforward modification of the proof of Theorem 3.2, as given in Appendix 3.A. □

The rA-DANSE^+_K algorithm can also be simplified to the rA-DANSE$_K$ algorithm, similarly to the rS-DANSE$_K$ algorithm as described in Section 3.4.3.

3.5 Numerical Simulations

3.5.1 Batch Mode Simulations

In this section, we simulate the different algorithms mentioned in this paper in batch mode. This means that all iterations are performed on the full signal length, including the estimation of the correlation matrices. We first evaluate the convergence speed of DANSE$_K$ and S-DANSE$_K$ in a scenario for which S-DANSE$_K$ converges. Then, we demonstrate the convergence of rS-DANSE^+_K and rS-DANSE$_K$ in a scenario for which S-DANSE$_K$ gets locked in a limit cycle.

The node-specific desired signals $\mathbf{d}_k$ are random mixtures of the Q channels of a latent signal $\mathbf{d}$, where $Q = 3$. All three channels of $\mathbf{d}$ are uniformly distributed random processes from which $N = 10000$ samples are generated. The M_k sensor signals of node k in $\mathbf{y}_k$ consist of different random mixtures of the three channels of the latent signal $\mathbf{d}$, with some uncorrelated Gaussian white noise added with half the power of the channels of $\mathbf{d}$. Each node has 10 sensors, i.e. $M_k = 10$, $\forall\, k \in \mathcal{J}$. All evaluations of the MSE cost functions J_k are performed on the equivalent least-squares (LS) cost functions, i.e.

$$\sum_{t=0}^{N} \|\mathbf{d}_k[t] - \mathbf{W}_k^H\mathbf{y}[t]\|^2 \,. \tag{3.35}$$

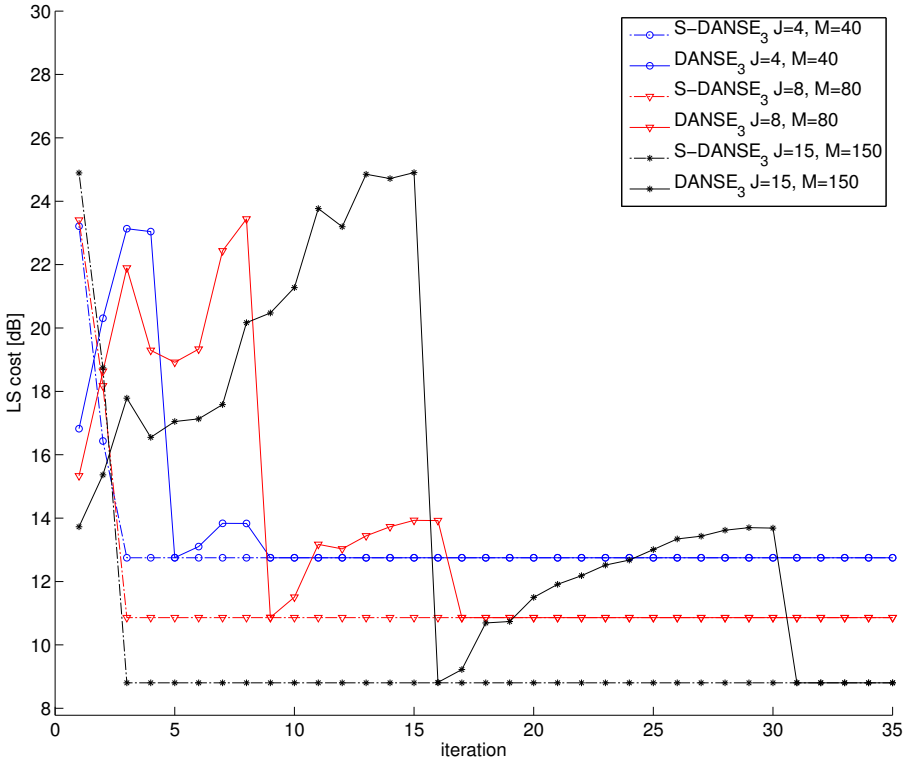


Figure 3.4: LS error of node 1 versus iteration i for fully connected networks with $J = 4$, $J = 8$ and $J = 15$ nodes respectively.

Also, the correlation matrices are replaced by their least squares equivalent, i.e. $E\{\mathbf{y}\mathbf{y}^H\}$ is replaced by $\mathbf{Y}\mathbf{Y}^H$ where $\mathbf{Y}$ denotes an $M \times N$ sample matrix that contains samples of the variable $\mathbf{y}$ in its columns.

The DANSE$_3$ algorithm and the S-DANSE$_3$ algorithm are simulated in networks with $J = 4$, $J = 8$ and $J = 15$ nodes. The result for node 1 is shown in Fig. 3.4. Notice that this is a case where S-DANSE$_3$ converges, and therefore no relaxation is required. Not surprisingly, the convergence time of the DANSE$_3$ algorithm, using sequential round-robin updates, increases linearly with the number of nodes. On the other hand, the convergence time of the S-DANSE$_3$ algorithm is unaffected when the number of nodes is increased. This shows that the simultaneous updating procedure generally yields faster convergence, especially when the number of nodes is large. Notice that in both algorithms, the algorithm has converged once each node has updated three times, irrespective of the number of nodes in the network.

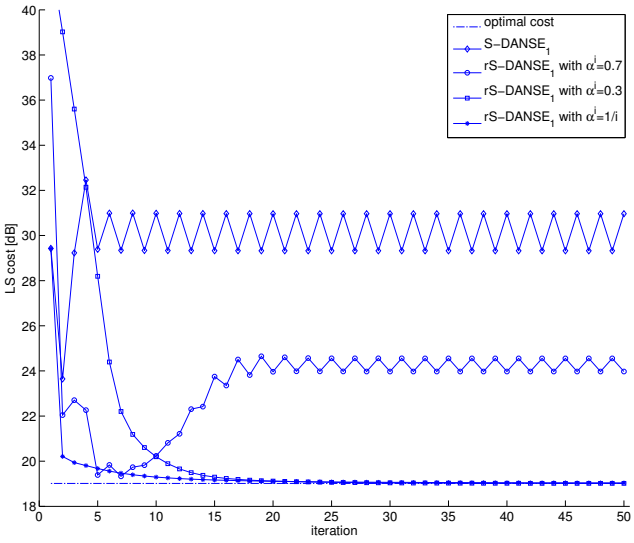
Fig. 3.5(a) shows a simulation result with $Q = 1$ where S-DANSE$_1$ does not converge to the optimal solution, but gets locked in a limit cycle. The network has $J = 4$ nodes with $M_k = 10$ for all k. All sensor signals are random mixtures of three white noise signals. Differently scaled versions of one of these white noise signals are used as desired signals for all four nodes. The other curves in figure 3.5(a) correspond to 3 versions of the rS-DANSE$_1$ algorithm. In the first version, $\alpha^i = 0.7 \; \forall \, i \in \mathbb{N}$, again resulting in a limit cycle. In the second version, α^i is set to $\alpha^i = 0.3 \; \forall \, i \in \mathbb{N}$, now yielding convergence to the optimal solution. In the third version, we choose the sequence $\left(\alpha^i\right)_{i\in\mathbb{N}}$ equal to $\alpha^i = \frac{1}{i}$, again yielding convergence.

Fig. 3.5(b) compares the convergence speed of rS-DANSE$_1$ with rS-DANSE$_1^+$ for the same scenario. Two relaxed versions are tested: $\alpha^i = 0.3 \; \forall \, i \in \mathbb{N}$, and $\alpha^i = \frac{1}{i}$. It is observed that rS-DANSE$_1^+$ converges faster due to the extra update (3.21). Notice that the version with $\alpha^i = \frac{1}{i}$ has an initial fast convergence, but eventually becomes very slow due to the decreasing stepsize.

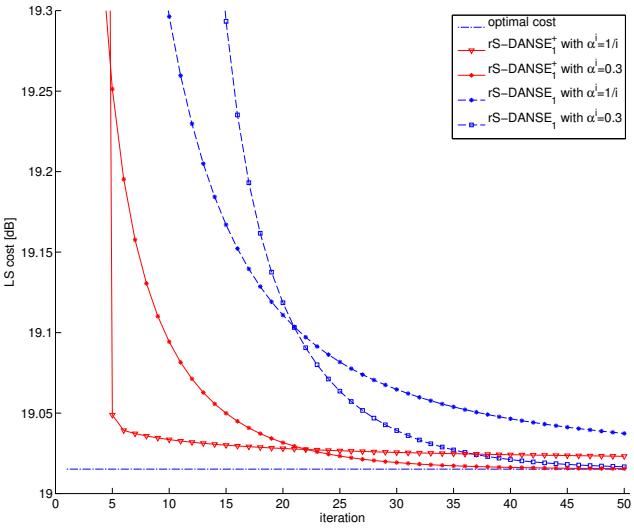
Simulation results with speech signals are provided in a follow-up paper [12]. In this paper, a distributed speech enhancement algorithm, based on the S-DANSE$_K$ algorithm and its relaxed variations, is tested in a simulated acoustic sensor network scenario. It is observed that limit cycles occur quite frequently, and therefore relaxation is required when nodes update simultaneously. Even though relaxation affects the adaptation speed, it is observed that rS-DANSE converges faster than DANSE with a sequential updating procedure.

3.5.2 Adaptive Implementation

In this section, we evaluate the tracking performance of the adaptive implementation of the rS-DANSE algorithm. The main difference with the batch



(a) S-DANSE$_1$ vs. rS-DANSE$_1$



(b) rS-DANSE$_1^+$ vs. rS-DANSE$_1$

Figure 3.5: Log-plot of the least square (LS) error of node 1 versus iteration i, for a case in which S-DANSE$_1$ does not converge. The network has $J = 4$ nodes with $M_k = 10$ for all k, and with $K = Q = 1$. By decreasing the relaxation parameter in the rS-DANSE$_1$ or rS-DANSE$_1^+$ algorithm, convergence occurs to the optimal value.

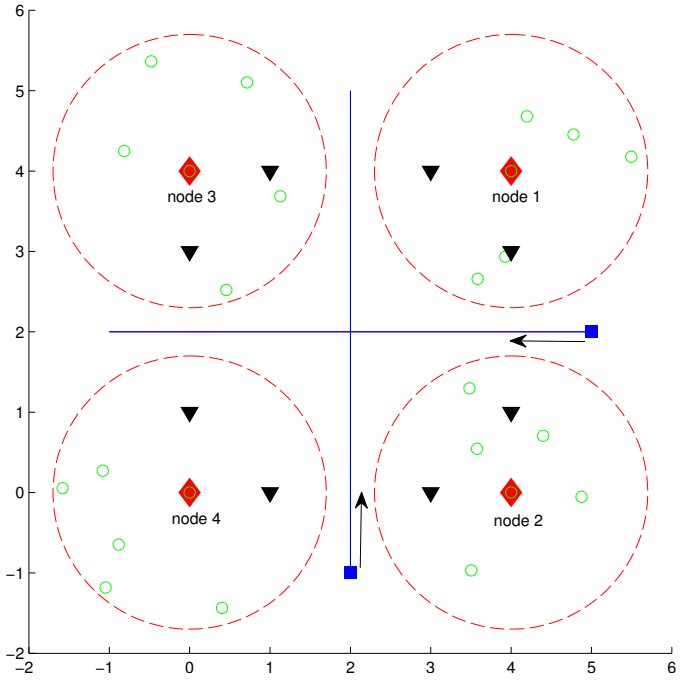


Figure 3.6: Description of the simulated scenario. The network contains 4 nodes ($\diamond$), each node collecting observations in a cluster of 6 sensors ($\circ$). One sensor of each cluster is positioned at the node itself. Two target sources ($\square$) are moving over the indicated straight lines. Eight noise sources are present ($\triangledown$).

mode simulations is that subsequent iterations are now performed on different signal segments, i.e. the same data segment is never used twice. This introduces a trade-off between tracking performance and estimation performance, which should be taken into account when a window length is chosen for the estimation of the signal statistics. Indeed, to have a fast tracking, the statistics must be estimated from short signal segments, yielding larger estimation errors in the correlation matrices that are used to compute the estimators at the different nodes. Another major difference with the previous section is that the correlation matrices are now estimated in a specific fashion by exploiting the on-off behavior of the target sources. We refer to [10] for further details.

The scenario is depicted in Fig. 3.6. The network contains $J = 4$ nodes ($\diamond$). Each node has a reference sensor at the node itself, and can collect observations of 5 additional sensors ($\circ$) that are uniformly distributed within a 1.6 m radius

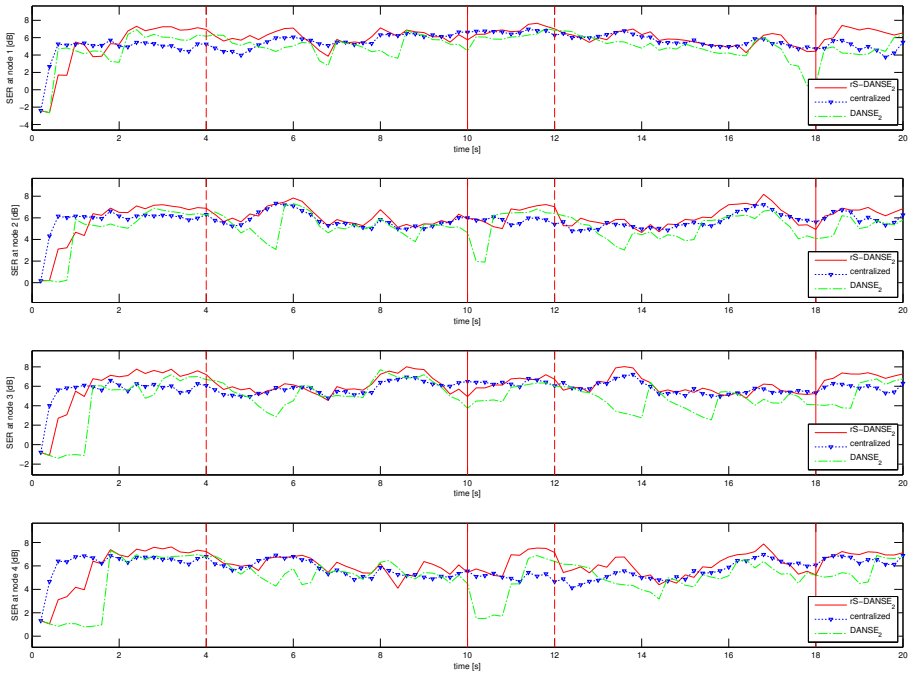


Figure 3.7: SER of DANSE$_2$ and rS-DANSE$_2$ vs. time at the 4 nodes depicted in Fig. 3.6, when the sources move at 1 m/s. The centralized version is added as a reference.

around the node. Eight localized white Gaussian noise sources ($\bigtriangledown$) are present. Two ($Q = 2$) target sources ($\square$) move back and forth over the indicated straight lines, and halt for 2 seconds at the end points of these lines. The goal for each node is to estimate the desired component d_{k1} in its reference sensor, which is a signal that consists of an unknown mixture of the two target sources, without any noise. Since the simulated scenario is exactly the same as in [10], we refer to the latter for further details on the data model and the signals involved.

We will use the signal-to-error ratio (SER) as a measure to assess the performance of the estimators. The instantaneous SER for node k at time t and iteration i is computed over 3200 samples, and is defined as

$$\mathrm{SER}_k^i[t] = \frac{\sum_{n=t+1}^{t+3200} |d_{k1}[n]|^2}{\sum_{n=t+1}^{t+3200} |d_{k1}[n] - \widetilde{\mathbf{w}}_k^i(1)^H \mathbf{y}[n]|^2} \tag{3.36}$$

where $\widetilde{\mathbf{w}}_k^i(1)$ denotes the first column of the estimator $\widetilde{\mathbf{W}}_k^i$, as defined in (3.6). Notice that this is the estimator that is of actual interest, since it estimates the desired component d_{k1} in the reference sensor.

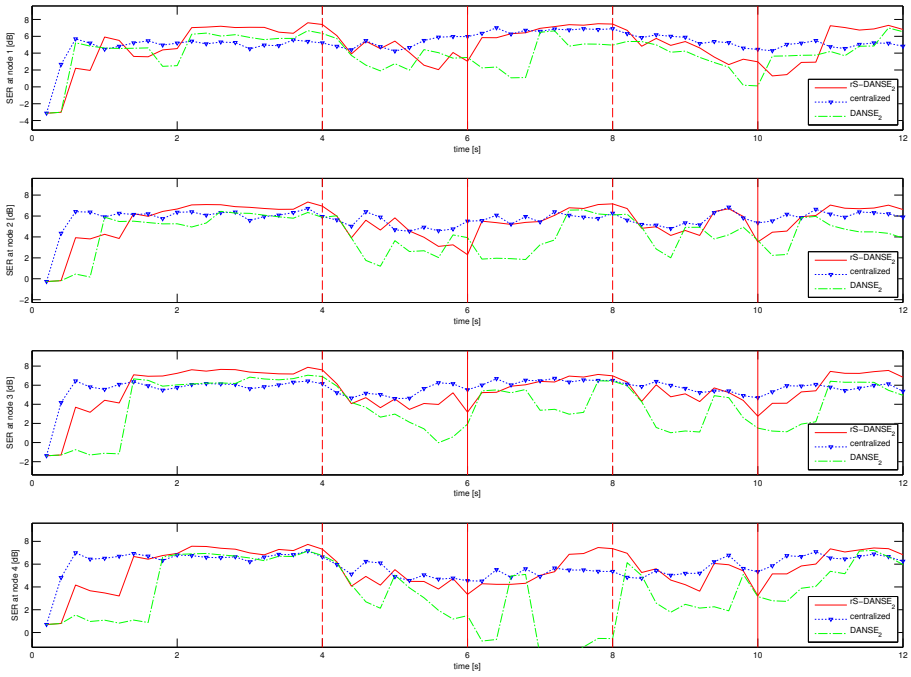


Figure 3.8: SER of DANSE_2 and rS-DANSE_2 vs. time at the 4 nodes depicted in Fig. 3.6, when the sources move at 3 m/s. The centralized version is added as a reference.

Fig. 3.7 shows the SER over time in the outputs of the four nodes, both for DANSE_2 and rS-DANSE_2, where the latter uses the relaxation parameter[10] $\alpha^i = 0.5$, $\forall\, i \in \mathbb{N}$, for the four different nodes. The window lengths for the estimation of the sensor signal correlation matrices and the noise correlation matrices[11] are $L_1 = 4200$ and $L_2 = 2200$, respectively. The time between two consecutive updates is 0.4 s, i.e. the iteration index i changes to $i+1$ every 0.4 seconds. The sources move at a speed of 1 m/s. Dashed vertical lines are plotted to indicate the points in time where both sources start moving, and full vertical lines indicate when they halt. The sources stand still in the time intervals [0-4] s, [10-12] s and [18-20] s. The performance is compared to the performance of the centralized version, in which all sensor signals are centralized in a single fusion center that computes the optimal estimators according to (3.4). The centralized version updates its estimators at every sample time, which results

[10]The value of the relaxation parameter is chosen manually. The S-DANSE_2 algorithm without relaxation is observed not to converge in this adaptive experiment.

[11]The noise correlation matrix is a sensor signal correlation matrix that is computed during noise-only periods, i.e. when the target sources are off. We refer to [10] for further details on why this noise correlation matrix has to be computed.

in very fast convergence.

It is observed that the rS-DANSE$_2$ algorithm has better tracking performance than the DANSE$_2$ algorithm with sequential updating. The latter has dips in the SER, corresponding to the time it takes to update all the nodes to obtain a good estimator for the current position of the sources. The rS-DANSE$_2$ can react more swiftly to the changing positions of the target sources, and is more or less able to follow the centralized estimators.

The same experiment is performed with the target sources moving three times faster, i.e. at a speed of 3 m/s. Now, the sources stand still in the time intervals [0-4] s, [6-8] s and [10-12] s. The results are shown in Fig. 3.8. It is observed that rS-DANSE$_2$ now has more difficulty following the centralized algorithm. However, once the sources stand still, the rS-DANSE$_2$ algorithm always regains optimality. The difference between rS-DANSE$_2$ and DANSE$_2$ is now even more significant, especially at nodes 3 and 4, where DANSE$_2$ shows some large SER dips.

3.6 Conclusion

In this paper, we have investigated the case in which the nodes in the DANSE$_K$ algorithm perform their parameter updates simultaneously or asynchronously. We have pointed out that the convergence and optimality of the DANSE$_K$ algorithm can no longer be guaranteed in this case. We have then modified the DANSE$_K$ algorithm with a relaxation operation to restore convergence and optimality under simultaneous and asynchronous updating. Convergence of the new algorithm is proven if the relaxation parameter satisfies certain conditions. Simulations indicate that a simplified version of the new algorithm can also be used, with a lower computational load, while maintaining convergence. Since all nodes can estimate the required statistics in parallel, the new algorithm adapts faster than the original DANSE$_K$ algorithm, especially so when the number of nodes is large. The convergence results are demonstrated with simulations in batch mode as well as with an adaptive version of the algorithm.

Appendix

3.A Proof of Theorem 3.2

Proof : For the sake of an easy exposition, we assume here that all signals are real valued. As explained in Appendix 3.B, the case with complex valued signals can be transformed to a real valued case for which the proof infra immediately

applies. In the proof, we also assume that all signal statistics are perfectly known, i.e. it is as if the algorithm uses an infinite observation window.

We define the cost function

$$J_k^p(\mathbf{w}_k) = |d_k(p) - \mathbf{w}_k^H \mathbf{y}|^2 \tag{3.37}$$

where $d_k(p)$ denotes the p-th entry of the vector $\mathbf{d}_k$. The cost function of the MMSE problem (3.3) is then equal to

$$J_k(\mathbf{W}_k) = \sum_{p=1}^{K} J_k^p(\mathbf{w}_k(p)) \tag{3.38}$$

where $\mathbf{w}_k(p)$ denotes the p-th column of $\mathbf{W}_k$. Similarly, we define $\mathbf{g}_k(p)$ as the p-th column of $\mathbf{G}_k$, and

$$\overline{\mathbf{W}}_k^i = \begin{bmatrix} \mathbf{W}_{11}^i \mathbf{G}_{k1}^{i+1} \\ \vdots \\ \mathbf{W}_{JJ}^i \mathbf{G}_{kJ}^{i+1} \end{bmatrix} \tag{3.39}$$

where $\mathbf{G}_k^{i+1} = [\mathbf{G}_{k1}^{i+1\,T} \dots \mathbf{G}_{kJ}^{i+1\,T}]^T$ is computed according to (3.22).

From (3.1), it readily follows that $\mathbf{d}_k^H = \mathbf{d}_q^H \mathbf{A}_{kq}$, where $\mathbf{A}_{kq} = \mathbf{A}_q^{-H} \mathbf{A}_k$. Using this, we can show that the results of update (3.22) are the same for all k, up to a transformation with the $K \times K$ matrix $\mathbf{A}_{kq}$, and therefore

$$\overline{\mathbf{W}}_k^i = \overline{\mathbf{W}}_q^i \mathbf{A}_{kq} \,. \tag{3.40}$$

We define the M-dimensional vector

$$\mathbf{s}_k^i(p) = \left[\mathbf{o}_{M_1}^T \; \dots \; \mathbf{o}_{M_{k-1}}^T \; \mathbf{w}_{kk}^{i\,T}(p) \; \mathbf{o}_{M_{k+1}}^T \; \dots \; \mathbf{o}_{M_J}^T \right]^T \tag{3.41}$$

where $\mathbf{o}_{M_k}$ denotes an all-zero M_k-dimensional vector, and where $\mathbf{w}_{kk}^i(p)$ denotes the p-th column of $\mathbf{W}_{kk}^i$. We define the following relationship between the matrix $\mathbf{W}^i$ and a KJ-dimensional subspace S^i of $\mathbb{R}^M$:

$$\mathbf{W}^i \rightarrow S^i = \mathit{Span}\{\mathbf{s}_1^i(1), \dots, \mathbf{s}_1^i(K), \dots, \mathbf{s}_J^i(1), \dots, \mathbf{s}_J^i(K)\} \,. \tag{3.42}$$

By definition, any column of $\mathbf{W}^i$ is inside S^i. The same holds true for all M-dimensional vectors of the form

$$\begin{bmatrix} \mathbf{W}_{11}^i \mathbf{h}_1 \\ \vdots \\ \mathbf{W}_{JJ}^i \mathbf{h}_J \end{bmatrix} \tag{3.43}$$

where $\mathbf{h}_q \in \mathbb{R}^K$, $\forall\, q \in \mathcal{J}$. Therefore, after update (3.22), the following expression holds:

$$J_k^p(\overline{\mathbf{w}}_k^i(p)) = \min_{\mathbf{w}_k \in S^i} J_k^p(\mathbf{w}_k) \,, \tag{3.44}$$

i.e. (3.22) defines the solution of the constrained optimization problem defined by the righthand side of (3.44), $\forall\, p \in \{1,\ldots,K\}$, where S^i defines the search space. This means that the subspace S^i is tangent to a sublevel set of J_k^p in the point $\overline{\mathbf{w}}_k^i(p)$, i.e. the p-th column of $\overline{\mathbf{w}}_k^i$. Therefore the gradient ∇J_k^p is orthogonal to S^i in $\overline{\mathbf{w}}_k^i(p)$, and therefore orthogonal to all vectors $\mathbf{s}_k^i(l)$, $\forall k \in \mathcal{J}$, $\forall\, l \in \{1,\ldots,K\}$. With (3.41), it is clear that the partitions of the gradient therefore satisfy

$$\begin{aligned}&\forall\, k,q \in \mathcal{J},\ \forall\, p,l \in \{1,\ldots,K\}:\\ &\quad \nabla_{\mathbf{w}_{kq}} J_k^p\left(\overline{\mathbf{w}}_k^i(p)\right) \perp \mathbf{w}_{qq}^i(l)\ .\end{aligned} \tag{3.45}$$

Now assume hypothetically, and without loss of generality, that node 1 updates the current values $\mathbf{W}_{11}^i$ and $\mathbf{G}_1^{i+1}$ to $\mathbf{W}_{11}^{(i+1)}$ and $\mathbf{G}_1^{(i+1)}$ respectively, following update (3.9) of the DANSE$_K$ algorithm. We added brackets to the iteration index to avoid confusion with update (3.21)-(3.27). From the proof of Theorem 3.1, as given in [10], it is found that

$$\begin{aligned}&\forall\, k \in \mathcal{J},\ \forall\, \mathbf{H}_k \in \mathbb{R}^{K\times K}:\\ &J_k\left(\begin{bmatrix}\mathbf{W}_{11}^{(i+1)}\\ \mathbf{W}_{22}^i\mathbf{G}_{12}^{(i+1)}\\ \vdots\\ \mathbf{W}_{JJ}^i\mathbf{G}_{1J}^{(i+1)}\end{bmatrix}\mathbf{A}_{k1}\right) \leq J_k\left(\begin{bmatrix}\mathbf{W}_{11}^i\mathbf{H}_1\\ \vdots\\ \mathbf{W}_{JJ}^i\mathbf{H}_J\end{bmatrix}\right)\ .\end{aligned} \tag{3.46}$$

We can show that this expression also holds for every cost function J_k^p separately. Therefore, the following inequality holds:

$$\begin{aligned}&\forall\, k \in \mathcal{J},\ \forall\, p \in \{1\ldots K\}:\\ &J_k^p\left(\begin{bmatrix}\mathbf{W}_{11}^{(i+1)}\\ \mathbf{W}_{22}^i\mathbf{G}_{12}^{(i+1)}\\ \vdots\\ \mathbf{W}_{JJ}^i\mathbf{G}_{1J}^{(i+1)}\end{bmatrix}\mathbf{a}_{k1}(p)\right) \leq J_k^p\left(\begin{bmatrix}\mathbf{W}_{11}^i\mathbf{g}_{k1}^{i+1}(p)\\ \vdots\\ \mathbf{W}_{JJ}^i\mathbf{g}_{kJ}^{i+1}(p)\end{bmatrix}\right)\end{aligned} \tag{3.47}$$

where $\mathbf{a}_{k1}(p)$ denotes the p-th column of $\mathbf{A}_{k1}$. By using (3.47), and since the cost functions are strictly convex, the directional vector $\triangle\mathbf{w}_k^{(i+1)}(p)$, defined by

$$\triangle\mathbf{w}_k^{(i+1)}(p) = \begin{bmatrix}\mathbf{W}_{11}^{(i+1)}\mathbf{a}_{k1}(p) - \mathbf{W}_{11}^i\mathbf{g}_{k1}^{i+1}(p)\\ \mathbf{W}_{22}^i(\mathbf{G}_{12}^{(i+1)}\mathbf{a}_{k1}(p) - \mathbf{g}_{k2}^{i+1}(p))\\ \vdots\\ \mathbf{W}_{JJ}^i(\mathbf{G}_{1J}^{(i+1)}\mathbf{a}_{k1}(p) - \mathbf{g}_{kJ}^{i+1}(p))\end{bmatrix} \tag{3.48}$$

defines a non-increasing direction of J_k^p in the current point $\overline{\mathbf{w}}_k^i(p)$. This means that

$$\forall\, k \in \mathcal{J},\ \forall\, p \in \{1,\ldots,K\}:$$

$$\nabla J_k^p \left(\overline{\mathbf{w}}_k^i(p)\right)^T \triangle \mathbf{w}_k^{(i+1)}(p) \leq 0 \; . \tag{3.49}$$

With (3.45), and the fact that $\mathbf{W}_{11}^{(i+1)} = F_1(\mathbf{W}^i)$, this yields

$$\forall\, k \in \mathcal{J},\, \forall\, p \in \{1, \ldots, K\} :$$

$$\sum_{p=1}^{K} \left(\nabla_{\mathbf{w}_{k1}} J_k^p \left(\overline{\mathbf{w}}_k^i(p)\right)^T F_1(\mathbf{W}^i) \mathbf{a}_{k1}(p) \right) \leq 0 \; . \tag{3.50}$$

By using a similar reasoning as above for any hypothetical update of any node q, we can show that

$$\forall\, k, q \in \mathcal{J},\, \forall\, p \in \{1, \ldots, K\} :$$

$$\sum_{p=1}^{K} \left(\nabla_{\mathbf{w}_{kq}} J_k^p \left(\overline{\mathbf{w}}_k^i(p)\right)^T F_q(\mathbf{W}^i) \mathbf{a}_{kq}(p) \right) \leq 0 \tag{3.51}$$

or equivalently, when using the matrix-valued gradient of the global cost function $J_k(\mathbf{W}_k)$

$$\forall\, k \in \mathcal{J},\, \forall\, q \in \mathcal{J} :$$

$$tr\left(\nabla_{\mathbf{W_{kq}}} J_k \left(\overline{\mathbf{W}}_k^i\right)^T F_q\left(\mathbf{W}^i\right) \mathbf{A}_{kq} \right) \leq 0 \tag{3.52}$$

with $tr(\mathbf{A})$ denoting the trace of matrix $\mathbf{A}$. It is noted that there is at least one $q \in \mathcal{J}$ for which the inequality in (3.52) is a strict inequality, as long as the equilibrium point has not been reached, i.e. as long as $\overline{\mathbf{W}}_k^i \neq \hat{\mathbf{W}}_k$, where $\hat{\mathbf{W}}_k$ denotes the optimal solution (3.4). Indeed, due to the strict convexity of the cost function J_k, the inequality in expression (3.47) is strict for at least one $p \in \{1, \ldots, J\}$ and at least one hypothetically updating node $q \in \mathcal{J}$ (although it might be required to choose another hypothetically updating node instead of node 1, which was chosen w.l.o.g. in expression (3.47)). Therefore, we find that

$$\overline{\mathbf{W}}_k^i \neq \hat{\mathbf{W}}_k \Leftrightarrow \forall\, k \in \mathcal{J},\, \exists\, q \in \mathcal{J} :$$

$$tr\left(\nabla_{\mathbf{W_{kq}}} J_k \left(\overline{\mathbf{W}}_k^i\right)^T F_q\left(\mathbf{W}^i\right) \mathbf{A}_{kq} \right) < 0 \; . \tag{3.53}$$

By defining

$$\triangle \mathbf{W}_{kk}^{i+1} = F_k(\mathbf{W}^i) - \overline{\mathbf{W}}_{kk}^i \tag{3.54}$$

we can rewrite (3.23) as

$$\mathbf{W}_{kk}^{i+1} = \overline{\mathbf{W}}_{kk}^i + \alpha^i \triangle \mathbf{W}_{kk}^{i+1} \; . \tag{3.55}$$

With this, the following expression is obtained

$$J_k(\overline{\mathbf{W}}_k^{i+1}) = \min_{\mathbf{G}_k} J_k \left(\begin{bmatrix} \mathbf{W}_{11}^{i+1} \mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}^{i+1} \mathbf{G}_{kJ} \end{bmatrix} \right) \tag{3.56}$$

$$\leq J_k \left(\begin{bmatrix} \left(\overline{\mathbf{W}}_{11}^i + \alpha^i \triangle \mathbf{W}_{11}^{i+1} \right) \mathbf{A}_{k1} \\ \vdots \\ \left(\overline{\mathbf{W}}_{JJ}^i + \alpha^i \triangle \mathbf{W}_{JJ}^{i+1} \right) \mathbf{A}_{kJ} \end{bmatrix} \right) . \tag{3.57}$$

With (3.40), the righthand side of (3.57) is equal to

$$J_k \left(\overline{\mathbf{W}}_k^i + \alpha^i \begin{bmatrix} \triangle \mathbf{W}_{11}^{i+1} \mathbf{A}_{k1} \\ \vdots \\ \triangle \mathbf{W}_{JJ}^{i+1} \mathbf{A}_{kJ} \end{bmatrix} \right) . \tag{3.58}$$

By using a first order approximation of the function J_k in the point $\overline{\mathbf{W}}_k^i$, we can rewrite (3.58) as

$$\begin{aligned} J_k \left(\overline{\mathbf{W}}_k^i \right) + \alpha^i tr \left(\nabla J_k \left(\overline{\mathbf{W}}_k^i \right)^T \begin{bmatrix} \triangle \mathbf{W}_{11}^{i+1} \mathbf{A}_{k1} \\ \vdots \\ \triangle \mathbf{W}_{JJ}^{i+1} \mathbf{A}_{kJ} \end{bmatrix} \right) \\ + \epsilon(\overline{\mathbf{W}}_k^i, \alpha^i) \end{aligned} \tag{3.59}$$

with the error term satisfying

$$\lim_{\alpha^i \to 0} \frac{\epsilon(\overline{\mathbf{W}}_k^i, \alpha^i)}{\alpha^i} = 0 . \tag{3.60}$$

The error term only depends on $\overline{\mathbf{W}}_k^i$ and α^i, i.e. the difference vector is implicitly computed from these two variables. With (3.54) and (3.45), we can rewrite expression (3.59) as

$$\begin{aligned} J_k \left(\overline{\mathbf{W}}_k^i \right) + \alpha^i tr \left(\nabla J_k \left(\overline{\mathbf{W}}_k^i \right)^T \begin{bmatrix} F_1 \left(\mathbf{W}^i \right) \mathbf{A}_{k1} \\ \vdots \\ F_J \left(\mathbf{W}^i \right) \mathbf{A}_{kJ} \end{bmatrix} \right) \\ + \epsilon(\overline{\mathbf{W}}_k^i, \alpha^i) . \end{aligned} \tag{3.61}$$

We can condense the chain (3.57)-(3.61) to

$$J_k \left(\overline{\mathbf{W}}_k^{i+1} \right) \leq J_k \left(\overline{\mathbf{W}}_k^i \right) + \alpha^i \left(V_k(\overline{\mathbf{W}}_k^i) + \frac{\epsilon(\overline{\mathbf{W}}_k^i, \alpha^i)}{\alpha^i} \right) , \tag{3.62}$$

where $V_k(\overline{\mathbf{W}}_k^i)$ denotes the function[12] described by the second term of (3.61). Because of (3.52) and (3.53), we know that

$$V_k(\overline{\mathbf{W}}_k^i) < 0 \Leftrightarrow \overline{\mathbf{W}}_k^i \neq \hat{\mathbf{W}}_k . \tag{3.63}$$

[12]Since $F(\overline{\mathbf{W}}_k^i) = F(\mathbf{W}^i)$, only $\overline{\mathbf{W}}_k^i$ is explicitly stated in the argument of the function V_k.

Using this observation, we can prove that the sequence $\left(\widetilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ converges to $\hat{\mathbf{W}}_k$. Since convergence of $\left(\widetilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ to $\hat{\mathbf{W}}_k$ is equivalent to convergence of the sequence $\left(\overline{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ to $\hat{\mathbf{W}}_k$, we only prove the latter. We do this by proving that

$$\forall\, \delta \in \mathbb{R}_0^+, \exists\, L \in \mathbb{N} : i > L \Rightarrow \|\hat{\mathbf{W}}_k - \overline{\mathbf{W}}_k^i\| < \delta\ . \tag{3.64}$$

Let $B_1 = B\left(\hat{\mathbf{W}}_k, \delta\right)$ denote a ball with center point $\hat{\mathbf{W}}_k$ and radius δ that is open, i.e. it does not contain its boundary. Since J_k is a convex quadratic function, there exists a sublevel set C of J_k that satisfies $C \subseteq B_1$. We choose any open ball $B_2 = B\left(\hat{\mathbf{W}}_k, \delta_2\right) \subset C$.

We now show by contradiction that the sequence $\left(\overline{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ contains an infinite number of elements in B_2. Let C_0 be the sublevel set of J_k that satisfies

$$C_0 = \{\mathbf{W}_k | J_k\left(\mathbf{W}_k\right) \leq J_k\left(\mathbf{O}_{M\times K}\right)\} \tag{3.65}$$

where $\mathbf{O}_{M\times K}$ denotes an $M \times K$ all-zero matrix. Because of (3.21), $\overline{\mathbf{W}}_k^i$ is always inside C_0. We define

$$\beta = \max_{\mathbf{W}_k \in C_0 \setminus B_2} V_k\left(\mathbf{W}_k\right)\ . \tag{3.66}$$

Because of (3.63) and the fact that B_2 is an open set, we know that this maximum exists, and that $\beta < 0$. We define

$$\epsilon^*(\alpha^i) = \max_{\mathbf{W}_k \in C_0 \setminus B_2} \epsilon\left(\mathbf{W}_k, \alpha^i\right)\ . \tag{3.67}$$

Because of (3.60), we know that

$$\exists L_1 \in \mathbb{N} : i > L_1 \Rightarrow \frac{\epsilon^*(\alpha^i)}{\alpha^i} < |\beta| - \kappa\ , \tag{3.68}$$

with an a priori chosen $\kappa \in (0, |\beta|)$. Let

$$\gamma = \sup\{\frac{\epsilon^*(\alpha^i)}{\alpha^i} | i > L_1\}\ , \tag{3.69}$$

and

$$\phi = \beta + \gamma\ . \tag{3.70}$$

By construction, ϕ must be strictly negative. By using (3.66)-(3.69), it follows from expression (3.62) that

$$(i > L_1 \wedge \overline{\mathbf{W}}_k^i \notin B_2) \Rightarrow J_k\left(\overline{\mathbf{W}}_k^{i+1}\right) \leq J_k\left(\overline{\mathbf{W}}_k^i\right) + \alpha^i\phi\ , \tag{3.71}$$

with $\wedge$ denoting the logic 'and' operator. Now suppose that $\exists\, L_2 > L_1 :\ i > L_2 \Rightarrow \overline{\mathbf{W}}_k^i \notin B_2$, then, because of (3.27) and the fact that ϕ is strictly negative,

$$\lim_{i\to\infty} J_k\left(\overline{\mathbf{W}}_k^i\right) = -\infty\ . \tag{3.72}$$

However, J_k has a lower bound, and therefore no such L_2 exists. This shows that there are an infinite number of elements in $\left(\overline{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ that are inside B_2.

Because of (3.26), the following expression holds:

$$\begin{array}{c} \exists L_3 \in \mathbb{N} : i > L_3 \\ \Downarrow \\ \max\limits_{\mathbf{W}_k\in B_2} \|Z_k^i(\mathbf{W}_k) - \mathbf{W}_k\| < \min\limits_{\mathbf{U}_1\in\partial C,\, \mathbf{U}_2\in B_2} \|\mathbf{U}_1 - \mathbf{U}_2\|\ , \end{array} \tag{3.73}$$

where ∂C is the boundary of the sublevel set C, and where $Z_k^i(\mathbf{W}_k)$ is defined as the function such that $\overline{\mathbf{W}}_k^{i+1} = Z_k^i(\mathbf{W}_k^i)$ according to the parallel update rules (3.21)-(3.27). Choose $L_4 > \max(L_1, L_3)$. Since there are an infinite number of elements in $\left(\overline{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ that are inside B_2, there exists an $i^* > L_4$ such that $\overline{\mathbf{W}}_k^{i^*}$ is inside B_2. Now we prove the induction hypothesis

$$\forall\, i > L_4 :\ \overline{\mathbf{W}}_k^i \in C \Rightarrow \overline{\mathbf{W}}_k^{i+1} \in C\ . \tag{3.74}$$

Assume that $\overline{\mathbf{W}}_k^i \in C$, then either $\overline{\mathbf{W}}_k^i \in C\backslash B_2$ or $\overline{\mathbf{W}}_k^i \in B_2$. In the case where $\overline{\mathbf{W}}_k^i \in C\backslash B_2$, we know that $\overline{\mathbf{W}}_k^{i+1} \in C$ because of (3.71) and the fact that $\phi < 0$. If $\overline{\mathbf{W}}_k^i \in B_2$, then $\overline{\mathbf{W}}_k^{i+1} \in C$ because of (3.73).

By applying the induction hypothesis (3.74) to $\overline{\mathbf{W}}_k^{i^*}$, we find that

$$\forall\, i > i^* :\ \overline{\mathbf{W}}_k^i \in C\ . \tag{3.75}$$

Because $C \subset B_1$, we have proved that expression (3.64) holds when choosing $L = i^*$. This completes the proof. □

3.B Transformation of Complex-Valued to Real-Valued DANSE

In this section, we briefly show how the DANSE scheme with complex valued signals can be tranformed to an isomorph scheme with only real valued signals. We define the following isomorphism between a complex number z and the 2×2 real matrix $\check{z}$:

$$z \leftrightarrow \check{z} = \begin{bmatrix} \Re z & -\Im z \\ \Im z & \Re z \end{bmatrix}\ , \tag{3.76}$$

where $\Re z$ and $\Im z$ denote the real and the imaginary part of z. We also define $\grave{z}$ and $\acute{z}$ as the first and second column of $\check{z}$. The isomorphism (3.76) defines another isomorphism between matrices

$$\mathbf{A} \in \mathbb{C}^{m\times n} \leftrightarrow \check{\mathbf{A}} \in \mathbb{R}^{2m\times 2n} \tag{3.77}$$

in such a way that each entry A_{ij} in $\mathbf{A}$ is replaced by the 2×2 matrix $\check{A}_{ij}$. Similarly, we define $\grave{\mathbf{A}}$ and $\acute{\mathbf{A}}$ as the matrices in which A_{ij} is replaced by $\grave{A}_{ij}$ and $\acute{A}_{ij}$, respectively. Notice that

$$z^* \leftrightarrow \check{z}^T \tag{3.78}$$

$$\mathbf{A}^H \leftrightarrow \check{\mathbf{A}}^T \ . \tag{3.79}$$

The cost function $J_k(\mathbf{W}_k)$, defined in (3.16), can now be transformed[13] to

$$\check{J}_k(\grave{\mathbf{W}}_k) = \frac{1}{2} E\left\{\|\check{\mathbf{d}}_k - \check{\mathbf{W}}_k^T \check{\mathbf{y}}\|_F^2\right\} \ , \tag{3.80}$$

with $\|.\|_F$ denoting the Frobenius norm. This cost function has real valued variables, and satisfies $\check{J}_k(\grave{\mathbf{W}}_k) = J_k(\mathbf{W}_k)$. The cost functions $\check{J}_k^p$ are defined similarly. If we let $\mathbf{D}$ denote the gradient of J_k^p in a certain point $\mathbf{w}_k$, then

$$\nabla \check{J}_k^p\left(\grave{\mathbf{w}}_k\right) = \grave{\mathbf{D}} \ . \tag{3.81}$$

The proof of Theorem 3.2 can now be extended to the complex valued case, by applying some minor changes using the above isomorphisms. The main changes that have to be made are as follows:

- Apply the isomorphisms (3.76) and (3.77) whenever a complex number or matrix is involved. For instance, (3.6) becomes:

$$\widetilde{\check{\mathbf{W}}}_k^i = \begin{bmatrix} \check{\mathbf{W}}_{11}^i \check{\mathbf{G}}_{k1}^i \\ \vdots \\ \check{\mathbf{W}}_{JJ}^i \check{\mathbf{G}}_{kJ}^i \end{bmatrix} \ . \tag{3.82}$$

- Change all cost functions into their equivalent with real valued variables, similarly to (3.80).
- The KJ dimensional subspace S^i in (3.42) now becomes the $2KJ$-dimensional subspace

$$\begin{aligned} S_i = Span\{&\grave{\mathbf{s}}_1^i(1), \acute{\mathbf{s}}_1^i(1), \ldots, \grave{\mathbf{s}}_1^i(K), \acute{\mathbf{s}}_1^i(K), \ldots, \\ &\grave{\mathbf{s}}_J^i(1), \acute{\mathbf{s}}_J^i(1), \ldots, \grave{\mathbf{s}}_J^i(K), \acute{\mathbf{s}}_J^i(K)\} \ . \end{aligned} \tag{3.83}$$

[13] We deliberately use (3.80) instead of $E\left\{\|\grave{\mathbf{d}}_k - \check{\mathbf{W}}_k^T \grave{\mathbf{y}}\|^2\right\}$, because this makes the relation between J_k and $\check{J}_k$ easier. The reason is that in (3.80), the full matrix $\check{\mathbf{W}}_k$ can be treated as a variable in the derivation of the gradient, instead of the argument $\grave{\mathbf{W}}_k$. The resulting gradient $\check{\mathbf{D}} = 2\mathbf{R}_{\check{y}\check{y}}\check{\mathbf{W}}_k - 2\mathbf{r}_{\check{y}\check{d}_k}$ will automatically have the correct structure, i.e. a matrix that is isomorph to the equivalent complex gradient $\mathbf{D} = 2\mathbf{R}_{yy}\mathbf{W}_k - 2\mathbf{r}_{yd_k}$, according to (3.77). This also holds in other cases, such as the update of $\check{\mathbf{G}}^{i+1}$ according to (3.22).

- All inner products between a gradient-vector or -matrix $\mathbf{D}$ and a second vector or matrix $\mathbf{V}$ are applied according to $\grave{\mathbf{D}}^T\grave{\mathbf{V}}$. In this case, (3.49) and similar expressions remain valid because of (3.81), and the fact that $\triangle\grave{\mathbf{w}}_k(p)$ is a descent direction of $\check{J}_k^p$ if and only if $\triangle\mathbf{w}_k(p)$ is a descent direction of J_k^p.

Bibliography

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. 2033–2036 vol.4, 2001.

[2] 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, Aug. 2007.

[3] ——, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, July 2008.

[4] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[5] I. Schizas, G. Giannakis, and Z.-Q. Luo, "Distributed estimation using reduced-dimensionality sensor observations," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4284–4299, Aug. 2007.

[6] Z.-Q. Luo, G. Giannakis, and S. Zhang, "Optimal linear decentralized estimation in a bandwidth constrained sensor network," *Proc. International Symposium on Information Theory (ISIT)*, pp. 1441–1445, Sept. 2005.

[7] K. Zhang, X. Li, P. Zhang, and H. Li, "Optimal linear estimation fusion - part VI: sensor data compression," *Proc. Sixth International Conference of Information Fusion*, vol. 1, pp. 221–228, 2003.

[8] Y. Zhu, E. Song, J. Zhou, and Z. You, "Optimal dimensionality reduction of sensor data in multisensor estimation fusion," *IEEE Transactions on Signal Processing*, vol. 53, no. 5, pp. 1631–1639, May 2005.

[9] A. Bertrand and M. Moonen, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2053–2056, April 2009.

[10] ——, "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, Oct. 2010.

[11] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, pp. 38–51, Jan. 2009.

[12] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435.

[13] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.

[14] A. Spriet, M. Moonen, and J. Wouters, "The impact of speech detection errors on the noise reduction performance of multi-channel wiener filtering and generalized sidelobe cancellation," *Signal Processing*, vol. 85, pp. 1073–1088, June 2005.

[15] T. Basar, "Relaxation techniques and asynchronous algorithms for on-line computation of non-cooperative equilibria," *Journal of Economic Dynamics and Control*, vol. 11, no. 4, pp. 531–549, December 1987.

[16] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

[17] S. Uryas'ev and R. Rubinstein, "On relaxation algorithms in computation of noncooperative equilibria," *IEEE Transactions on Automatic Control*, vol. 39, no. 6, pp. 1263–1267, Jun 1994.

Chapter 4

Robust DANSE for Speech Enhancement

Robust distributed noise reduction in hearing aids with external acoustic sensor nodes

Alexander Bertrand and Marc Moonen

Published in *EURASIP Journal on Advances in Signal Processing*,
vol. 2009, Article ID 530435, 14 pages, 2009.
doi:10.1155/2009/530435

Contributions of first author

- literature study
- co-development of the R-DANSE$_K$ algorithm
- co-establishment of proof of convergence and optimality of R-DANSE$_K$
- design of simulation scenarios
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

In this paper, the benefit of using external acoustic sensor nodes for noise reduction in hearing aids is demonstrated in a simulated acoustic scenario with multiple sound sources. A distributed adaptive node-specific signal estimation (DANSE) algorithm, that has a reduced communication bandwidth and computational load, is evaluated. Batch-mode simulations compare the noise reduction performance of a centralized MWF algorithm with DANSE. In the simulated scenario, DANSE is observed not to be able to achieve the same performance as its centralized multi-channel Wiener filtering (MWF) equivalent, although in theory both should generate the same set of filters. A modification to DANSE is proposed to increase its robustness, yielding smaller discrepancy between the performance of DANSE and the centralized MWF. Furthermore, the influence of several parameters such as the DFT size used for frequency domain processing and possible delays in the communication link between nodes is investigated.

4.1 Introduction

Noise reduction algorithms are crucial in hearing aids to improve speech understanding in background noise. For every increase of 1 dB in signal-to-noise ratio, speech understanding increases by roughly 10% [1]. By using an array of microphones, it is possible to exploit spatial characteristics of the acoustic scenario. However, in many classical beamforming applications, the acoustic field is sampled only locally because the microphones are placed close to each other. The noise reduction performance can often be increased when extra microphones are used at significantly different positions in the acoustic field. For example, an exchange of microphone signals between a pair of hearing aids in a binaural configuration, i.e. one at each ear, can significantly improve the noise reduction performance [2–11]. The distribution of extra acoustic sensor nodes in the acoustic environment, each having a signal processing unit and a wireless link, allows further performance improvement. For instance, small sensor nodes can be incorporated into clothing, or placed strategically either close to desired sources to obtain high SNR signals, or close to noise sources to collect noise references. In a scenario with multiple hearing aid users, the different hearing aids can exchange signals to improve their performance through cooperation.

The set-up envisaged here requires a wireless link between the hearing aid and the supporting external acoustic sensor nodes. A distributed approach using compressed signals is needed, since collecting and processing all available microphone signals at the hearing aid itself would require a large communication bandwidth and computational power. Furthermore, since the positions of the

external nodes are unknown, the algorithm should be adaptive and able to cope with unknown microphone positions. Therefore, a multi-channel Wiener filter (MWF) approach is considered, since an MWF optimally estimates the clean speech signal in MMSE sense without relying on prior knowledge on the microphone positions [12]. In [13, 14], a distributed adaptive node-specific signal estimation (DANSE) algorithm is introduced for MMSE signal estimation in a sensor network, which significantly reduces the communication bandwidth while still obtaining the optimal MMSE estimators, i.e. the Wiener filters, as if each node has access to all signals in the network. The term 'node-specific' refers to the scenario in which each node acts as a data-sink and estimates a different desired signal. This situation is particularly interesting in the context of noise reduction in binaural hearing aids where the two hearing aids estimate differently filtered versions of the same desired speech source signal, which is indeed important to preserve the auditory cues for directional hearing [15–18]. In [19], a pruned version of the DANSE algorithm, referred to as distributed multi-channel Wiener filtering (DB-MWF), has been used for binaural noise reduction. In the case of a single desired source signal, it was proven that DB-MWF converges to the optimal all-microphone Wiener filter settings in both hearing aids. The more general DANSE algorithm allows the incorporation of multiple desired sources and more than two nodes. Furthermore, it allows for uncoordinated updating where each node decides independently in which iteration steps it updates its parameters, possibly simultaneously with other nodes. This in particular avoids the need for a network wide protocol that coordinates the updates between nodes.

In this paper, batch-mode simulation results are described to demonstrate the benefit of using additional external sensor nodes for noise reduction in hearing aids. Furthermore, the DANSE algorithm is reformulated in a noise reduction context, and a batch-mode analysis of the noise reduction performance of DANSE is provided. The results are compared to those obtained with the centralized MWF algorithm that has access to all signals in the network to compute the optimal Wiener filters. Although in theory the DANSE algorithm converges to the same filters as the centralized MWF algorithm, this is not the case in the simulated scenario. The resulting decrease in performance is explained and a modified algorithm is then proposed to increase robustness and to allow the algorithm to converge to the same filters as in the centralized MWF algorithm. Furthermore, the effectiveness of relaxation is shown when nodes update their filters simultaneously, as well as the influence of several parameters such as the DFT size used for frequency domain processing, and possible delays within the communication link. The simulations in this paper show the potential of DANSE for noise reduction, as suggested in [13, 14], and provide a proof-of-concept for applying the algorithm in cooperative acoustic sensor networks for distributed noise reduction applications, such as hearing aids.

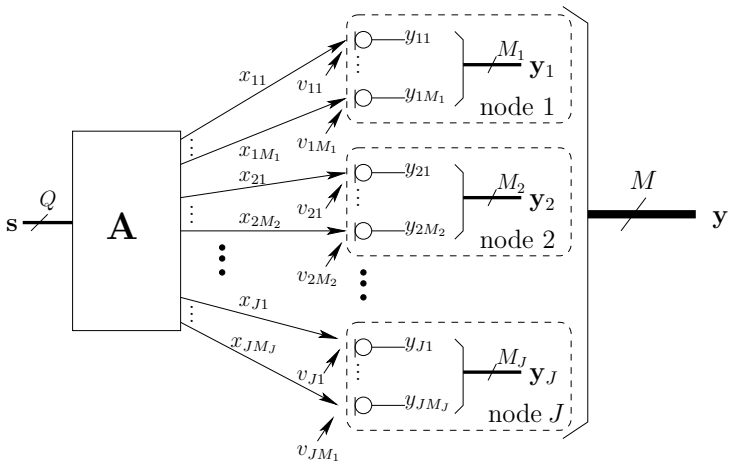


Figure 4.1: Data model for a sensor network with J sensor nodes, in which node k collects M_k noisy observations of the Q source signals in $\mathbf{s}$.

The outline of this paper is as follows. In Section 4.2, the data model is introduced and the multi-channel Wiener filtering process is reviewed. In Section 4.3, a description of the simulated acoustic scenario is provided. Moreover, an analysis of the benefits achieved using external acoustic sensor nodes is given. In Section 4.4, the DANSE algorithm is reviewed in the context of noise reduction. A modification to DANSE increasing robustness is introduced in Section 4.5. Batch-mode simulation results are given in Section 4.6. Since some practical aspects are disregarded in the simulations, some remarks and open problems concerning a practical implementation of the algorithm, are given in Section 4.7.

4.2 Data Model and Multi-Channel Wiener Filtering

4.2.1 Data Model and Notation

A general fully connected broadcasting sensor network with J nodes is considered, in which each node k has direct access to a specific set of M_k microphones, with $M = \sum_{k=1}^{J} M_k$ (see Fig. 4.1). Nodes can be either a hearing aid or a supporting external acoustic sensor node. Each microphone signal m of node k can be described in the frequency domain as

$$y_{km}(\omega) = x_{km}(\omega) + v_{km}(\omega), \;\; m = 1, ..., M_k \tag{4.1}$$

where $x_{km}(\omega)$ is a desired speech component and $v_{km}(\omega)$ an undesired noise component. Although $x_{km}(\omega)$ is referred to as the desired speech component, $v_{km}(\omega)$ is not necessarily non-speech, i.e. undesired speech sources may be included in $v_{km}(\omega)$. All subsequent algorithms will be implemented in the frequency domain, where (4.1) is approximated based on finite-length time-to-frequency domain transformations. For conciseness, the frequency-domain variable ω will be omitted. All signals y_{km} of node k are stacked in an M_k-dimensional vector $\mathbf{y}_k$, and all vectors $\mathbf{y}_k$ are stacked in an M-dimensional vector $\mathbf{y}$. The vectors $\mathbf{x}_k$, $\mathbf{v}_k$ and $\mathbf{x}$, $\mathbf{v}$ are similarly constructed. The network-wide data model can now be written as $\mathbf{y} = \mathbf{x} + \mathbf{v}$. Notice that the desired speech component $\mathbf{x}$ may consist of multiple desired source signals, for example when a hearing aid user is listening to a conversation between multiple speakers, possibly talking simultaneously. If there are Q desired speech sources, then

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{4.2}$$

where $\mathbf{A}$ is an $M \times Q$-dimensional steering matrix and $\mathbf{s}$ a Q-dimensional vector containing the Q desired sources. Matrix $\mathbf{A}$ contains the acoustic transfer functions (evaluated at frequency ω) from each of the speech sources to all microphones, incorporating room acoustics and microphone characteristics.

4.2.2 Centralized Multi-Channel Wiener Filtering

The goal of each node k is to estimate the desired speech component x_{km} in its m-th microphone, selected to be the reference microphone. Without loss of generality, it is assumed that the reference microphone always corresponds to $m = 1$. For the time being, it is assumed that each node has access to all microphone signals in the network. Node k then performs a filter-and-sum operation on the microphone signals with filter coefficients $\mathbf{w}_k$ that minimize the following MSE cost function

$$J_k(\mathbf{w}_k) = E\left\{|x_{k1} - \mathbf{w}_k^H \mathbf{y}|^2\right\} \tag{4.3}$$

in which the superscript H denotes the conjugate transpose operator. Notice that at each node k, one such MSE problem is to be solved for each frequency bin. The minimum of (4.3) corresponds to the well-known Wiener filter solution:

$$\hat{\mathbf{w}}_k = \mathbf{R}_{yy}^{-1}\mathbf{R}_{yx}\mathbf{e}_{k1} \tag{4.4}$$

with $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$, $\mathbf{R}_{yx} = E\{\mathbf{y}\mathbf{x}^H\}$ and $\mathbf{e}_{k1}$ an M-dimensional vector with only one entry equal to 1 and all other entries equal to 0, which selects the column of $\mathbf{R}_{yx}$ corresponding to the reference microphone of node k. This procedure is referred to as multi-channel Wiener filtering (MWF). If the desired speech sources are uncorrelated to the noise, then $\mathbf{R}_{yx} = \mathbf{R}_{xx} = E\{\mathbf{x}\mathbf{x}^H\}$. In the remaining of this paper, it is implicitly assumed that all Q desired sources

may be active at the same time, yielding a rank-Q speech correlation matrix $\mathbf{R}_{xx}$. In practice, $\mathbf{R}_{xx}$ is unknown, but can be estimated from

$$\mathbf{R}_{xx} = \mathbf{R}_{yy} - \mathbf{R}_{vv} \tag{4.5}$$

with $\mathbf{R}_{vv} = E\{\mathbf{v}\mathbf{v}^H\}$. The noise correlation matrix $\mathbf{R}_{vv}$ can be (re-)estimated during noise-only periods and $\mathbf{R}_{yy}$ can be (re-)estimated during speech-and-noise periods, requiring a voice activity detection (VAD) mechanism. Even when the noise sources and the speech source are not stationary, these practical estimators are found to yield good noise reduction performance [15, 19].

4.3 Simulation Scenario & the Benefit of External Acoustic Sensor Nodes

The performance of microphone array based noise reduction typically increases with the number of microphones. However, the number of microphones that can be placed on a hearing aid is limited, and the acoustic field is only sampled locally, i.e. at the hearing aid itself. Therefore, there is often a large distance between the location of the desired source and the microphone array, which results in signals with low SNR. In fact, the SNR decreases with 6 dB for every doubling of the distance between a source and a microphone. The noise reduction performance can therefore be greatly increased by using supporting external acoustic sensor nodes that are connected to the hearing aid through a wireless link.

To assess the potential improvement that can be obtained by adding external sensor nodes, a multi-source scenario is simulated using the image method [20]. Fig. 4.2 shows a schematic illustration of the scenario. The room is cubical (5m $\times$ 5m $\times$ 5m) with a reflection coefficient of 0.4 at the floor, the ceiling and at every wall. According to Sabine's formula this corresponds to a reverberation time of $T_{60} = 0.222$ s. There are two hearing aid users listening to speaker C, who produces a desired speech signal. One hearing aid user has 2 hearing aids (node 2 and 3) and the other has one hearing aid at the right ear (node 4). All hearing aids have three omnidirectional microphones with a spacing of 1 cm. Head shadow effects are not taken into account. Node 1 is an external microphone array containing six omnidirectional microphones placed 2 cm from each other. Speakers A and B both produce speech signals interfering with speaker C. All speech signals are sentences from the HINT ('Hearing in Noise Test') database [21]. The upper left loudspeaker produces multi-talker babble noise (Auditec) with a power normalized to obtain an input broadband SNR of 0 dB in the first microphone of node 4, which is used as the reference node. In addition to the localized noise sources, all microphone signals have an uncorrelated noise component which consist of white noise with power that

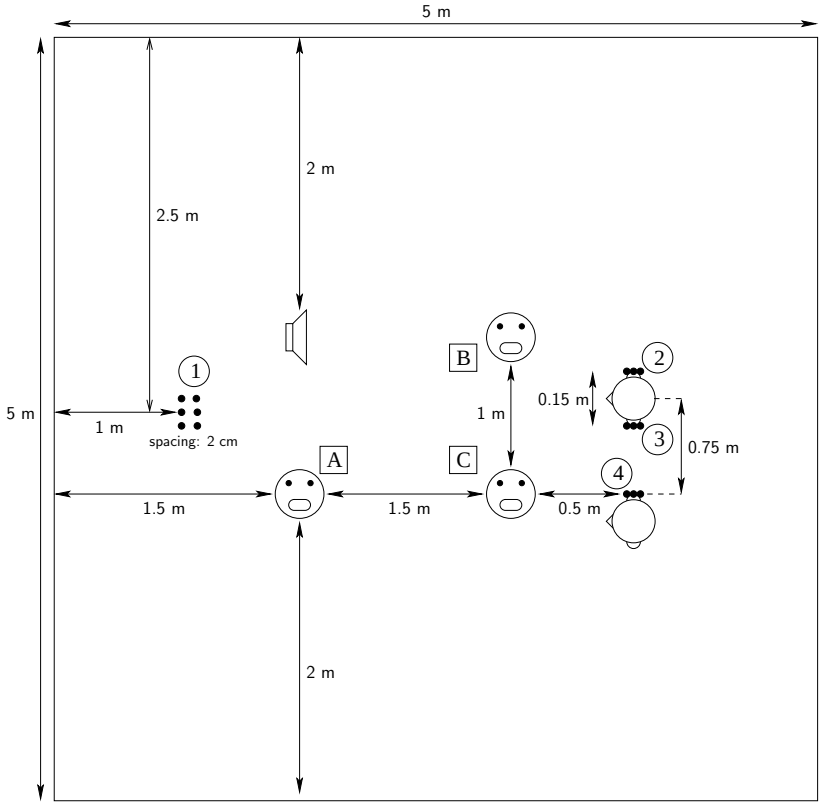


Figure 4.2: The acoustic scenario used in the simulations throughout this paper. Two persons with hearing aids are listening to speaker C. The other sources produce interference noise.

is 10% of the power of the desired signal in the first microphone of node 4. All nodes and all sound sources are in the same horizontal plane, 2 m above ground level.

Notice that this is a difficult scenario, with many sources and highly non-stationary (speech) noise. This kind of scenario brings many practical issues, especially with respect to reliable VAD decisions (cfr. Section 4.7). Throughout this paper, many of these practical aspects are disregarded. The aim here is to demonstrate the benefit that can be achieved with external sensor nodes, in particular in multi-source scenarios. Furthermore, the theoretical performance of the DANSE algorithm, introduced in Section 4.4, will be assessed with respect to the centralized MWF algorithm. To isolate the effects of VAD errors and estimation errors on the correlation matrices, all experiments are performed in batch mode with ideal VAD's.

Two performance measures are used to assess the quality of the noise reduction algorithms, namely the broadband signal-to-noise ratio (SNR) and the signal-to-distortion ratio (SDR). The SNR and SDR at node k are defined as

$$\mathrm{SNR} = 10\log_{10}\frac{E\left\{\hat{x}_k[t]^2\right\}}{E\left\{\hat{n}_k[t]^2\right\}} \tag{4.6}$$

$$\mathrm{SDR} = 10\log_{10}\frac{E\left\{x_{k1}[t]^2\right\}}{E\left\{\left(x_{k1}[t]-\hat{x}_k[t]\right)^2\right\}} \tag{4.7}$$

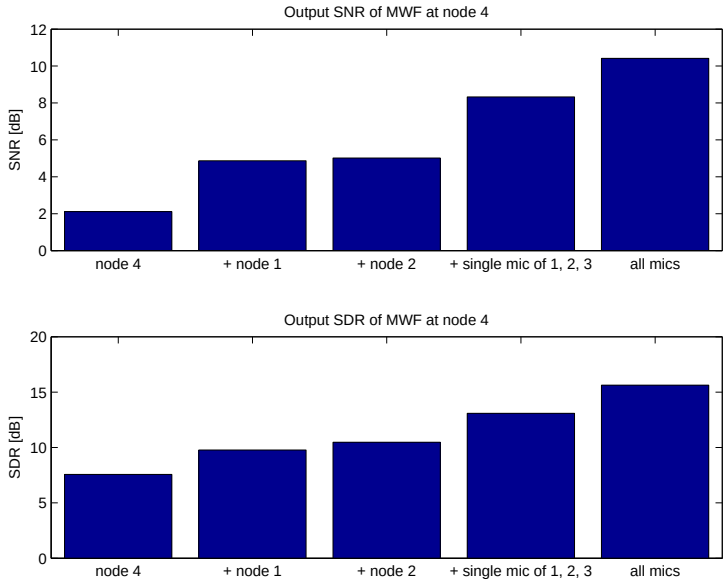
with $\hat{n}_k[t]$ and $\hat{x}_k[t]$ the time domain noise component and the desired speech component respectively at the output at node k, and $x_{k1}[t]$ the desired time domain speech component in the reference microphone of node k.

The sampling frequency is 32 kHz in all experiments. The frequency domain noise reduction is based on DFT's with size equal to $L = 512$ if not specified otherwise. Notice that L is equivalent to the filter length of the time domain filters that are implicitly applied to the microphone signals. The DFT size $L = 512$ is relatively large, which is due to the fact that microphones are far apart from each other, leading to higher time differences of arrival (TDOA) demanding longer filters to exploit spatial information. If the filter lengths are too short to allow a sufficient alignment between the signals, then the noise reduction performance degrades. This is evaluated in Section 4.6.4. To allow small DFT-sizes, yet large distances between microphones, delay compensation should be introduced in the local microphone signals or the received signals at each node. However, since hearing aids typically have hard constraints on the processing delay to maintain lip synchronization, this delay compensation is restricted. This, in effect, introduces a trade-off between input-output delay and noise reduction performance.

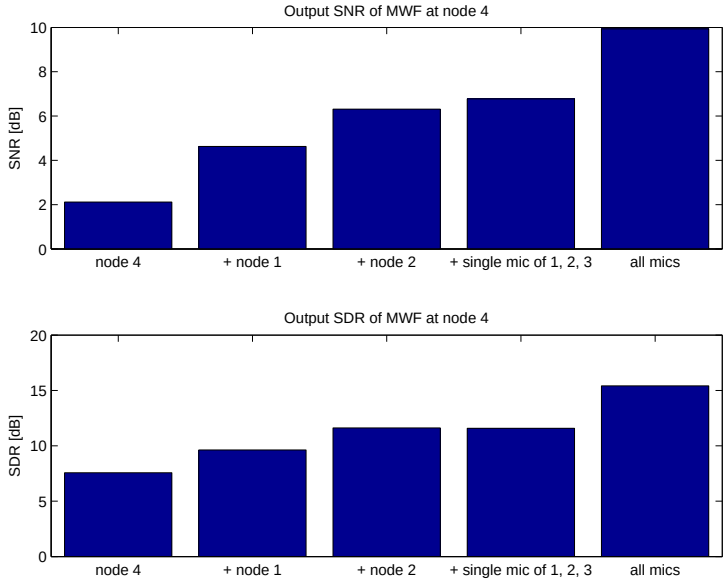
Fig. 4.3(a) shows the output SNR and SDR of the centralized MWF procedure at node 4 when five different subsets of microphones are used for the noise reduction:

1. The microphone signals of node 4 itself.
2. The microphone signals of node 1 in addition to the microphone signals of node 4 itself.
3. The microphone signals of node 2 in addition to the microphone signals of node 4 itself.
4. The first microphone signal at every node in addition to all microphone signals of node 4 itself. This is equivalent to a scenario where the network supporting node 4 consists of single-microphone nodes, i.e. $M_k = 1$, for $k = 1, \ldots, 3$.
5. All microphone signals in the network.

The benefit of adding external microphones is very clear in this graph. It also shows that microphones with a significantly different position contribute



(a) Scenario of Fig. 4.2



(b) Scenario of Fig. 4.2 with vertical position of node 1 reduced by 0.5 m

Figure 4.3: Comparison of output SNR and SDR of MWF at node 4 for five different microphone subsets.

more than microphones that are closely spaced. Indeed, cases 2, 3 and 4 both add three extra microphone signals, but the benefit is largest in case 4, in which the additional microphones are relatively set far apart. However, using multi-microphone nodes (case 5) still produces a significant benefit of about 25% (2 dB) in comparison to single-microphone nodes (case 4). Notice that the benefit of placing external microphones, and the benefit of using multi-microphone nodes in comparison to single-microphone nodes, is of course very scenario specific. For instance, if the vertical position of node 1 is reduced by 0.5 m in Fig. 4.2, then the difference between single-microphone nodes (case 4) and multi-microphone nodes (case 5) is more than 3 dB, as shown in Fig. 4.3(b), which correponds to an improvement of almost 50%.

4.4 The DANSE Algorithm

In Section 4.3, simulations showed that adding external microphones in addition to the microphones available in a hearing aid may yield a great benefit in terms of both noise suppression and speech distortion. Not surprisingly, adding external nodes with multiple microphones boosts the performance even more. However, the latter introduces a significant increase in communication bandwidth, depending on the number of microphones in each node. Furthermore, the dimensions of the correlation matrix to be inverted in formula (4.4) may grow significantly. However, if each node has its own signal processor unit, this extra communication bandwidth can be reduced and the computation can be distributed by using the distributed adaptive node-specific signal estimation (DANSE) algorithm, as proposed in [13, 14]. The DANSE algorithm computes the optimal network wide Wiener filter in a distributed, iterative fashion. In this section this algorithm is briefly reviewed and reformulated in a noise reduction context.

4.4.1 The DANSE$_K$ Algorithm

In the DANSE$_K$ algorithm, each node k estimates K different desired signals, corresponding to the desired speech components in K of its microphones (assuming that $K \leq M_k$, $\forall\, k \in \{1, \ldots, J\}$). Without loss of generality, it is assumed that the first K microphones are selected, i.e. the signal to be estimated is the K-channel signal $\overline{\mathbf{x}}_k = [x_{k1} \ \ldots \ x_{kK}]^T$. The first entry in this vector corresponds to the reference microphone, whereas the other $K-1$ entries should be viewed as auxiliary channels. They are required to fully capture the signal subspace spanned by the desired source signals. Indeed, if K is chosen equal to Q, the K channels of $\overline{\mathbf{x}}_k$ define the same signal subspace as defined by the channels in $\mathbf{s}$, i.e.

$$\overline{\mathbf{x}}_k = \mathbf{A}_k \mathbf{s} \tag{4.8}$$

where $\mathbf{A}_k$ denotes a $K \times K$ submatrix of the steering matrix $\mathbf{A}$ in formula (4.2). K being equal to Q is a requirement for DANSE$_K$ to be equivalent to the centralized MWF solution (see Theorem 4.1). The case in which $K \neq Q$ is not considered here. For a more detailed discussion why these auxiliary channels are introduced, we refer to [13].

Each node k estimates its desired signal $\overline{\mathbf{x}}_k$ with respect to a corresponding MSE cost function

$$J_k(\mathbf{W}_k) = E\left\{\|\overline{\mathbf{x}}_k - \mathbf{W}_k^H \mathbf{y}\|^2\right\} \tag{4.9}$$

with $\mathbf{W}_k$ an $M \times K$ matrix, defining a multiple-input multiple-output (MIMO) filter. Notice that this corresponds to K independent estimation problems in which the same M-channel input signal $\mathbf{y}$ is used. Similarly to (4.3), the Wiener solution of (4.9) is given by

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^{-1}\mathbf{R}_{xx}\mathbf{E}_k \tag{4.10}$$

with

$$\mathbf{E}_k = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{O}_{(M-K)\times K} \end{bmatrix} \tag{4.11}$$

with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix and $\mathbf{O}_{U\times V}$ denoting an all-zero $U \times V$ matrix. The matrix $\mathbf{E}_k$ selects the first K columns of $\mathbf{R}_{xx}$, corresponding to the K-channel signal $\overline{\mathbf{x}}_k$. The DANSE$_K$ algorithm will compute (4.10) in an iterative, distributed fashion. Notice that only the first column of $\hat{\mathbf{W}}_k$ is of actual interest, since this is the filter that estimates the desired speech component in the reference microphone. The auxiliary columns of $\hat{\mathbf{W}}_k$ are by-products of the DANSE$_K$ algorithm.

A partitioning of the matrix $\mathbf{W}_k$ is defined as $\mathbf{W}_k = [\mathbf{W}_{k1}^T \ \dots \ \mathbf{W}_{kJ}^T]^T$ where $\mathbf{W}_{kq}$ denotes the $M_k \times K$ submatrix of $\mathbf{W}_k$ that is applied to $\mathbf{y}_q$ in (4.9). Since node k only has access to $\mathbf{y}_k$, it can only apply the partial filter $\mathbf{W}_{kk}$. The K-channel output signal of this filter, defined by $\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k$, is then broadcast to the other nodes. Another node q can filter this K-channel signal $\mathbf{z}_k$ that it receives from node k by a MIMO filter defined by the $K \times K$ matrix $\mathbf{G}_{qk}$. This is illustrated in Fig. 4.4 for a three-node network ($J = 3$). Notice that the actual $\mathbf{W}_k$ that is applied by node k is now parametrized as

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k1} \\ \mathbf{W}_{22}\mathbf{G}_{k2} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{kJ} \end{bmatrix}. \tag{4.12}$$

In what follows, the matrices $\mathbf{G}_{kk}$, $\forall\, k \in \{1, \dots, J\}$, are assumed to be $K \times K$ identity matrices $\mathbf{I}_K$ to minimize the degrees of freedom (they are omitted in Fig. 4.4). Node k can only manipulate the parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_{k1} \ \dots \ \mathbf{G}_{kJ}$. If (4.8) holds, it is shown in [13] that the solution space defined by the parametrization (4.12) contains the centralized solution $\hat{\mathbf{W}}_k$.

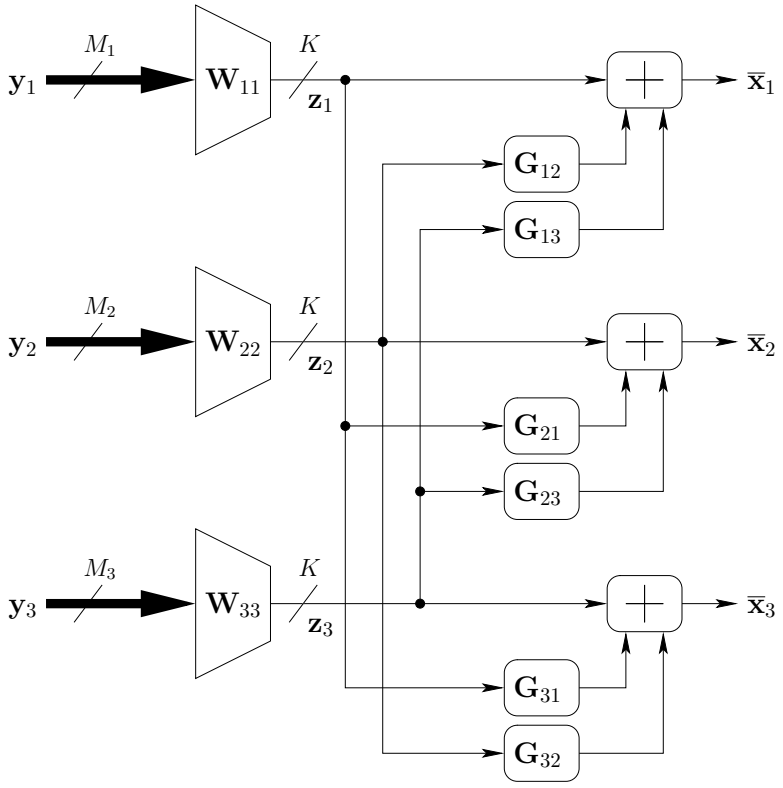


Figure 4.4: The DANSE$_K$ scheme with 3 nodes ($J = 3$). Each node k estimates the desired signal $\overline{\mathbf{x}}_k$ using its own M_k-channel microphone signal, and 2 K-channel signals broadcast by the other two nodes.

Notice that each node k broadcasts a K-channel[1] signal $\mathbf{z}_k$, which is the output of the $M_k \times K$ MIMO filter $\mathbf{W}_{kk}$, acting both as a compressor and an estimator at the same time. The subscript K thus refers to the (maximum) number of channels of the broadcast signal. DANSE$_K$ compresses the data to be sent by node k by a factor of $\max\{\frac{M_k}{K}, 1\}$. Further compression is possible, since the channels of the broadcast signal $\mathbf{z}_k$ are highly correlated, but this is not taken into consideration throughout this paper.

The DANSE$_K$ algorithm will iteratively update the elements at the righthand side of eq. (4.12) to optimally estimate the desired signals $\overline{\mathbf{x}}_k$, $\forall k \in \{1, \ldots, J\}$. To describe this updating procedure, the following notation is used. The matrix $\mathbf{G}_k = [\mathbf{G}_{k1}^T \ldots \mathbf{G}_{kJ}^T]^T$ stacks all transformation matrices of node k. The matrix $\mathbf{G}_{k,-q}$ defines the matrix $\mathbf{G}_k$ in which $\mathbf{G}_{kq}$ is omitted. The $K(J-1)$-channel

[1]Here it is assumed without loss of generality that $K \leq M_k$, $\forall k \in \{1, \ldots, J\}$. If this does not hold at a certain node k, this node will transmit its unfiltered microphone signals.

signal $\mathbf{z}_{-k}$ is defined as $\mathbf{z}_{-k} = \left[\mathbf{z}_1^T \dots \mathbf{z}_{k-1}^T \mathbf{z}_{k+1}^T \dots \mathbf{z}_J^T\right]^T$. In what follows, a superscript i refers to the value of the variable at iteration step i. Using this notation, the DANSE$_K$ algorithm consists of the following iteration steps:

1. Initialize
   $i \leftarrow 0$
   $k \leftarrow 1$
   $\forall\, q \in \{1, ..., J\}$: $\mathbf{W}_{qq} \leftarrow \mathbf{W}_{qq}^0$, $\mathbf{G}_{q,-q} \leftarrow \mathbf{G}_{q,-q}^0$, $\mathbf{G}_{qq} \leftarrow \mathbf{I}_K$, where $\mathbf{W}_{qq}^0$ and $\mathbf{G}_{q,-q}^0$ are random matrices of appropriate dimension.
2. Node k updates its local parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_{k,-k}$ by solving a local estimation problem based on its own local microphone signals $\mathbf{y}_k$ together with the compressed signals $\mathbf{z}_q^i = \mathbf{W}_{qq}^{i\,H}\mathbf{y}_q$ that it receives from the other nodes $q \neq k$, i.e. it minimizes

$$\widetilde{J}_k^i(\mathbf{W}_{kk}, \mathbf{G}_{k,-k}) = E\left\{\|\overline{\mathbf{x}}_k - \left[\mathbf{W}_{kk}^H \mid \mathbf{G}_{k,-k}^H\right]\tilde{\mathbf{y}}_k^i\|^2\right\} \tag{4.13}$$

   where

$$\tilde{\mathbf{y}}_k^i = \left[\frac{\mathbf{y}_k}{\mathbf{z}_{-k}^i}\right] . \tag{4.14}$$

   Define $\tilde{\mathbf{x}}_k^i$ similarly as (4.14), but now only containing the desired speech components in the considered signals. The update performed by node k is then

$$\left[\frac{\mathbf{W}_{kk}^{i+1}}{\mathbf{G}_{k,-k}^{i+1}}\right] = \left(\widetilde{\mathbf{R}}_{yy,k}^i\right)^{-1}\widetilde{\mathbf{R}}_{xx,k}^i\mathbf{E}_k \tag{4.15}$$

   with

$$\mathbf{E}_k = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{O}_{(M_k-K+K(J-1))\times K} \end{bmatrix} \tag{4.16}$$

$$\widetilde{\mathbf{R}}_{yy,k}^i = E\left\{\tilde{\mathbf{y}}_k^i\tilde{\mathbf{y}}_k^{i\,H}\right\} \tag{4.17}$$

$$\widetilde{\mathbf{R}}_{xx,k}^i = E\left\{\tilde{\mathbf{x}}_k^i\tilde{\mathbf{x}}_k^{i\,H}\right\} . \tag{4.18}$$

   The parameters of the other nodes do not change, i.e.

$$\forall\, q \in \{1, \dots, J\}\backslash\{k\} :\ \mathbf{W}_{qq}^{i+1} = \mathbf{W}_{qq}^i,\ \mathbf{G}_{q,-q}^{i+1} = \mathbf{G}_{q,-q}^i . \tag{4.19}$$

3. $\mathbf{W}_{kk} \leftarrow \mathbf{W}_{kk}^{i+1}$, $\mathbf{G}_{k,-k} \leftarrow \mathbf{G}_{k,-k}^{i+1}$
   $k \leftarrow (k \mod J) + 1$
   $i \leftarrow i + 1$
4. Return to step 2

Notice that node k updates its parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_{k,-k}$, according to a local multi-channel Wiener filtering problem with respect to its $M_k + (J-1)K$ input channels. This MWF problem is solved in the same way as the MWF problem given in (4.3) or (4.9).

Theorem 4.1 *Assume that $K = Q$. If $\overline{\mathbf{x}}_k = \mathbf{A}_k\mathbf{s}$, $\forall\, k \in \{1, \ldots, J\}$, with $\mathbf{A}_k$ a full rank $K \times K$ matrix, then the DANSE$_K$ algorithm converges for any k to the optimal filters (4.10) for any initialization of the parameters.*

Proof : See [13]. □

Notice that DANSE$_K$ theoretically provides the same output as the centralized MWF algorithm if $K = Q$. The requirement that $\overline{\mathbf{x}}_k = \mathbf{A}_k\mathbf{s}$, $\forall\, k \in \{1, \ldots, J\}$, is satisfied because of (4.2). However, notice that the data model (4.2) is only approximately fullfilled in practice due to a finite-length DFT size. Consequently, the rank of the speech correlation matrix $\mathbf{R}_{xx}$ is not Q, but it has Q dominant eigenvalues instead. Therefore, the theoretical claims of convergence and optimality of DANSE$_K$, with $K = Q$, are only approximately true in practice due to frequency domain processing.

4.4.2 Simultaneous Updating

The DANSE$_K$ algorithm as described in Section 4.4.1 performs sequential updating in a round-robin fashion, i.e. nodes update their parameters one at a time. In [22], it is observed that convergence of DANSE is no longer guaranteed when nodes update simultaneously, or in an uncoordinated fashion where each node decides independently in which iteration steps it updates its parameters. This is however an interesting case, since a simultaneous updating procedure allows for parallel computation, and uncoordinated updating removes the need for a network wide protocol that coordinates the updates between nodes.

Let $\mathbf{W} = [\mathbf{W}_{11}^T \mathbf{W}_{22}^T \ldots \mathbf{W}_{JJ}^T]^T$ and let $F(\mathbf{W})$ be the function that defines the simultaneous DANSE$_K$ update of all parameters in $\mathbf{W}$, i.e. F applies (4.15) $\forall\, k \in \{1, \ldots J\}$ simultaneously. Experiments in [22] show that the update $\mathbf{W}^{i+1} = F(\mathbf{W}^i)$ may lead to limit cycle behavior. To avoid these limit cycles, the following relaxed version of DANSE is suggested in [22]:

$$\mathbf{W}^{i+1} = (1 - \alpha^i)\mathbf{W}^i + \alpha^i F(\mathbf{W}^i) \tag{4.20}$$

with stepsizes α^i satisfying

$$\alpha^i \in (0, 1] \tag{4.21}$$

$$\lim_{i \to \infty} \alpha^i = 0 \tag{4.22}$$

$$\sum_{i=0}^{\infty} \alpha^i = \infty\,. \tag{4.23}$$

The suggested conditions on the stepsize α^i are however quite conservative and may result in slow convergence. In most cases, the simultaneous update procedure converges already when a constant value for α^i is chosen $\forall i \in \mathbb{N}$ that

is sufficiently small. In all simulations performed for the scenario in Section 4.3, a value of $\alpha^i = 0.5$, $\forall\, i \in \mathbb{N}$ was found to eliminate limit cycles in every set-up.

4.5 Robust DANSE

4.5.1 Robustness Issues in DANSE

In Section 4.6, simulation results will show that the DANSE algorithm does not achieve the optimal noise reduction performance as predicted by theorem 4.1. There are two important reasons for this suboptimal performance.

The first reason is the fact that the DANSE$_K$ algorithm assumes that the signal space spanned by the channels of $\overline{\mathbf{x}}_k$ is well-conditioned, $\forall\, k \in \{1, \ldots, J\}$. This assumption is reflected in theorem 4.1 by the condition that $\mathbf{A}_k$ be full rank for all k. Although this is mostly satisfied in practice, the $\mathbf{A}_k$'s are often ill-conditioned. For instance, the distance between microphones in a single node is mostly small, yielding a steering matrix with several columns that are almost identical, i.e. an ill-conditioned matrix $\mathbf{A}_k$ in the formulation of theorem 4.1.

The microphones of nodes that are close to a noise source typically collect low SNR signals. Despite the low SNR, these signals can boost the performance of the MWF algorithm, since they can act as noise references to cancel out noise in the signals recorded by other nodes. However, the DANSE algorithm cannot fully exploit this since the local estimation problem at such low SNR nodes is ill-conditioned. If node k has low SNR microphone signals $\mathbf{y}_k$, the correlation matrix $\overline{\mathbf{R}}_{xx,k} = E\{\overline{\mathbf{x}}_k \overline{\mathbf{x}}_k^H\}$ has large estimation errors, since the corresponding noise correlation matrix $\overline{\mathbf{R}}_{vv,k}$ and the speech+noise correlation matrix $\overline{\mathbf{R}}_{yy,k}$ are very similar, i.e. $\overline{\mathbf{R}}_{vv,k} \approx \overline{\mathbf{R}}_{yy,k}$. Notice that $\overline{\mathbf{R}}_{xx,k}$ is a submatrix of $\widetilde{\mathbf{R}}_{xx,k}$ defined in (4.18), which is used in the DANSE$_K$ algorithm. From another point of view, this also relates to an ill-conditioned steering matrix $\mathbf{A}$, since the submatrix $\mathbf{A}_k$ is close to an all-zero matrix compared to the submatrices corresponding to nodes with higher SNR signals.

4.5.2 Robust DANSE (R-DANSE)

In this section, a modification to the DANSE algorithm is proposed to achieve a better noise reduction performance in the case of low SNR nodes or ill-conditioned steering matrices. The main idea is to replace an ill-conditioned $\mathbf{A}_k$ matrix by a better conditioned matrix by changing the estimation problem at node k. The new algorithm is referred to as 'robust DANSE' or R-DANSE. In what follows, the notation $v(p)$ is used to denote the p-th entry in a vector $\mathbf{v}$, and $\mathbf{m}(p)$ is used to denote the p-th column in the matrix $\mathbf{M}$.

For each node k, the channels in $\overline{\mathbf{x}}_k$ that cause ill-conditioned steering matrices, or that correspond to low SNR signals, are discarded and replaced by the desired speech components in the signal(s) $\mathbf{z}_q^i$ received from other (high SNR) nodes $q \neq k$, i.e.

$$\overline{x}_k^i(p) = \mathbf{w}_{qq}^i(l)^H \mathbf{x}_q, \;\; q \in \{1,\ldots,J\}\backslash\{k\}, \;\; l \in \{1,\ldots,K\}\;, \tag{4.24}$$

if x_{kp} causes an ill-conditioned steering matrix or if x_{kp} corresponds to a low SNR microphone, and

$$\overline{x}_k^i(p) = x_{kp} \tag{4.25}$$

otherwise. Notice that the desired signal $\overline{\mathbf{x}}_k^i$ may now change at every iteration, which is reflected by the superscript i denoting the iteration index.

To decide whether to use (4.24) or (4.25), the condition number of the matrix $\mathbf{A}_k$ does not necessarily have to be known. In principle, it is always better to replace the $K-1$ auxiliary channels in $\overline{\mathbf{x}}_k$ as in formula (4.24), where a different q should be chosen for every p. Indeed, since microphones of different nodes are typically far apart from each other, better conditioned steering matrices are then obtained. Also, since the correlation matrix $\widetilde{\mathbf{R}}_{xx,k}$ is better estimated when high SNR signals are available, the chosen q's preferably correspond to high SNR nodes. Therefore, the decision procedure requires knowledge of the SNR at the different nodes. For a low SNR node k, one can also replace all K channels in $\overline{\mathbf{x}}_k$ as in (4.24), including the reference microphone. In this case, there is no estimation of the speech component that is collected by the microphones of node k itself. However, since the network wide problem is now better conditioned, the other nodes in the network will benefit from this.

The R-DANSE$_K$ algorithm performs the same steps as explained in Section 4.4.1 for the DANSE$_K$ algorithm, but now $\overline{\mathbf{x}}_k^i$ replaces $\overline{\mathbf{x}}_k$ in (4.13)-(4.18). This means that in R-DANSE, the $\mathbf{E}_k$ matrix in (4.16) now may contain ones at row indices that are higher than M_k. To guarantee convergence of R-DANSE, the placement of ones in (4.16), or equivalently the choices for q and l in (4.24), are not completely free, as explained in the next section.

4.5.3 Convergence of R-DANSE

To provide convergence results, the dependencies of each individual estimation problem are described by means of a directed graph[2] $\mathcal{G}$ with KJ vertices, where each vertex corresponds to one of the locally computed filters, i.e. a specific column of $\mathbf{W}_{kk}$ for $k = 1 \ldots J$. The graph contains an arc from filter a to b, described by the ordered pair (a, b), if the output of filter b contains the desired speech component that is estimated by filter a. For example, formula (4.24) defines the arc ($\mathbf{w}_{kk}(p)$,$\mathbf{w}_{qq}(l)$). A vertex v that has no departing arc is

[2]Readers that are not familiar with the jargon of graph theory might want to consult [23], although in principle no prior knowledge on graph theory is assumed.

referred to as a direct estimation filter (DEF), i.e. the signal to be estimated is the desired speech component in one of the node's own microphone signals, as in formula (4.25).

To illustrate this, a possible graph is shown in Fig. 4.5 for DANSE$_2$ applied to the scenario described in Section 4.3. Since the microphone signals of node 1 have a low SNR, the two desired signals in $\overline{\mathbf{x}}_1$ that are used in the computation of $\mathbf{W}_{11}$ are replaced by the filtered desired speech component in the received signals from higher SNR nodes 2 and 4, i.e. $\mathbf{w}_{22}(1)^H\mathbf{x}_2$ and $\mathbf{w}_{44}(1)^H\mathbf{x}_4$ respectively. This corresponds to the arcs $(\mathbf{w}_{11}(1), \mathbf{w}_{22}(1))$ and $(\mathbf{w}_{11}(2), \mathbf{w}_{44}(1))$. To calculate $\mathbf{w}_{22}(1)$, $\mathbf{w}_{33}(1)$ and $\mathbf{w}_{44}(1)$, the desired speech components x_{21}, x_{31} and x_{41} in the respective reference microphones are used. These filters are DEF's, and are shaded in Fig. 4.5. The microphones at node 2 are very close to each other. Therefore, to avoid an ill-conditioned matrix $\mathbf{A}_2$ at node 2, the signals to be estimated by $\mathbf{w}_{22}(2)$ should be provided by another node, and not by another microphone signal of node 2 itself. Therefore, the arc $(\mathbf{w}_{22}(2), \mathbf{w}_{44}(1))$ is added. For similar reasons, the arcs $(\mathbf{w}_{33}(2), \mathbf{w}_{44}(1))$ and $(\mathbf{w}_{44}(2), \mathbf{w}_{22}(1))$ are also added.

Theorem 4.2 *Let all assumptions of theorem 4.1 be satisfied. Let $\mathcal{G}$ be the directed graph describing the dependencies of the estimation problems in the R-DANSE$_K$ algorithm as described above. If $\mathcal{G}$ is acyclic, then the R-DANSE$_K$ algorithm converges to the optimal filters to estimate the desired signals defined by $\mathcal{G}$.*

Proof : The proof of theorem 4.1 in [13] on convergence of DANSE$_K$ is based on the assumption that the desired K-channel signals $\overline{\mathbf{x}}_k$, $\forall\, k \in \{1, \ldots, J\}$, are all in the same K-dimensional signal subspace spanned by the K sources in $\mathbf{s}$, i.e.

$$\overline{\mathbf{x}}_k = \mathbf{A}_k\mathbf{s}\ . \tag{4.26}$$

This assumption remains valid in R-DANSE$_K$. Indeed, since $\mathbf{x}_q$ contains M_q linear combination of the Q sources in $\mathbf{s}$, the signal $x_k^i(p)$ given by (4.24) is again a linear combination of the source signals. However, the coefficients of this linear combinations may change at every iteration as the signal $x_k^i(p)$ is an output of the adaptive filter $\mathbf{w}_{qq}^i(l)$ in another node q. This then leads to a modified version of theorem 4.1 for DANSE$_K$ in which the matrix $\mathbf{A}_k$ in (4.26) is not fixed, but may change at every iteration, i.e.

$$\overline{\mathbf{x}}_k^i = \mathbf{A}_k^i\mathbf{s}\ . \tag{4.27}$$

Define

$$\overline{\mathbf{W}}_{kq}^i = \underset{\mathbf{W}_{kq}}{\arg\min}\left(\min_{\mathbf{G}_{k,-q}} E\left\{\|\overline{\mathbf{x}}_k - \left[\mathbf{W}_{kq}^H \mid \mathbf{G}_{k,-q}^H\right]\tilde{\mathbf{y}}_q^i\|^2\right\}\right)\ . \tag{4.28}$$

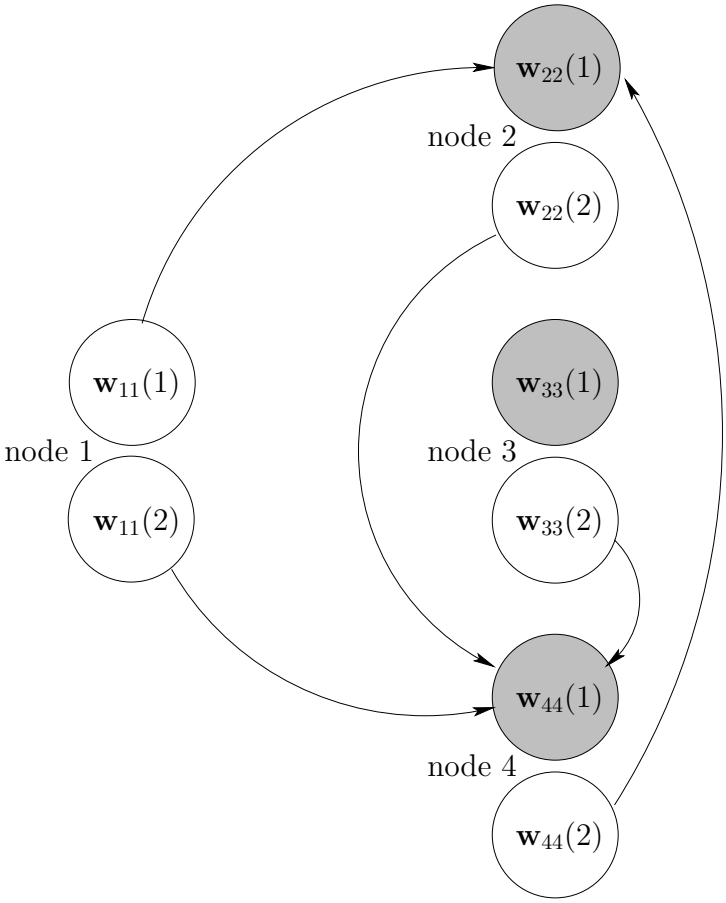


Figure 4.5: Possible graph describing dependencies of estimations problems for DANSE_2 applied to the acoustic scenario described in Section 4.3.

This corresponds to the hypothetical case in which node k would optimise $\mathbf{W}_{kq}^i$ directly, without the constraint $\mathbf{W}_{kq}^i = \mathbf{W}_{qq}^i \mathbf{G}_{kq}^i$ where node k depends on the parameter choice of node q.

In [13] it is proven that for DANSE$_K$, under the assumptions of theorem 4.1 the following holds:

$$\forall\, q, k \in \{1, \ldots, J\}: \;\; \overline{\mathbf{W}}_{kq}^i = \overline{\mathbf{W}}_{qq}^i \mathbf{A}_{kq} \tag{4.29}$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H} \mathbf{A}_k^H$. This means that the columns of $\overline{\mathbf{W}}_{qq}^i$ span a K-dimensional subspace that also contains the columns of $\overline{\mathbf{W}}_{kq}^i$, which is the optimal update with respect to the cost function J_k^i of node k, as if there were no constraints on $\mathbf{W}_{kq}^i$. Or in other words: an update by node q automatically optimizes the cost function of any other node k with respect to $\mathbf{W}_{kq}$, if node k performs a responding optimization of $\mathbf{G}_{kq}$, yielding $\mathbf{G}_{kq}^{\text{opt}} = \mathbf{A}_{kq}$. Therefore, the following expression holds:

$$\forall\, k \in \{1, ..., J\},\, \forall\, i \in \mathbb{N}:$$

$$\min_{\mathbf{G}_{k,-k}} \widetilde{J}_k^{i+1}\left(\mathbf{W}_{kk}^{i+1}, \mathbf{G}_{k,-k}\right) \leq \min_{\mathbf{G}_{k,-k}} \widetilde{J}_k^{i}\left(\mathbf{W}_{kk}^{i}, \mathbf{G}_{k,-k}\right) \;. \tag{4.30}$$

Notice that this holds at every iteration for every node. In the case of R-DANSE$_K$, the $\mathbf{A}_{kq}$ matrix of expression (4.29) changes at every iteration. At first sight, expression (4.30) remains valid, since changes in the matrix $\mathbf{A}_{kq}$ are compensated by the minimization over $\mathbf{G}_{kq}$ in (4.30). However, this is not true since the desired signals $\overline{\mathbf{x}}_k^i$ also change at every iteration, and therefore the cost functions at different iterations cannot be compared. However, (4.30) can be partitioned in K sub-expressions:

$$\forall\, p \in \{1, \ldots, K\},\, \forall\, k \in \{1, ..., J\},\, \forall\, i \in \mathbb{N}:$$

$$\min_{\mathbf{g}_{k,-k}(p)} \widetilde{J}_{kp}^{i+1}\left(\mathbf{w}_{kk}^{i+1}(p), \mathbf{g}_{k,-k}(p)\right) \leq \min_{\mathbf{g}_{k,-k}(p)} \widetilde{J}_{kp}^{i}\left(\mathbf{w}_{kk}^{i}(p), \mathbf{g}_{k,-k}(p)\right) \tag{4.31}$$

with

$$\widetilde{J}_{kp}^i(\mathbf{w}_{kk}, \mathbf{g}_{k,-k}) = E\left\{\left|\overline{x}_k(p) - \left[\mathbf{w}_{kk}^H \mid \mathbf{g}_{k,-k}^H\right] \widetilde{\mathbf{y}}_k^i\right|^2\right\} \;. \tag{4.32}$$

For the R-DANSE$_K$ case, (4.32) remains the same, except that $\overline{x}_k(p)$ has to be replaced with $\overline{x}_k^i(p)$. As explained above, due to this modification, expression (4.31) does not hold anymore. However, it does hold for the cost functions J_{kp}^i corresponding to a DEF $\mathbf{w}_{kk}(p)$, i.e. a filter for which the desired signal is directly obtained from one of the microphone signals of node k. Indeed, every DEF $\mathbf{w}_{kk}(p)$ has a well-defined cost function $\widetilde{J}_{kp}^i$, since the signal $\overline{x}_k^i(p)$ is fixed over different iteration steps. Because $\widetilde{J}_{kp}^i$ has a lower bound, (4.31) shows that the sequence $\{\min_{\mathbf{g}_{k,-k}^p} \widetilde{J}_{kp}^i\}_{i\in\mathbb{N}}$ converges. The convergence of this sequence implies convergence of the sequence $\{\mathbf{w}_{kk}^i(p)\}_{i\in\mathbb{N}}$, as shown in [13].

After convergence of all $\mathbf{w}_{kk}(p)$ parameters corresponding to a DEF, all vertices in the graph $\mathcal{G}$ that are directly connected to this DEF have a stable desired signal, and their corresponding cost functions become well-defined. The above argument shows that these filters then also converge.

Continuing this line of thought, convergence properties of the DEF will diffuse through the graph. Since the graph is acyclic, all vertices converge. Convergence of all $\mathbf{W}_{kk}$ parameters for $k = 1 \dots J$ automatically yields convergence of all $\mathbf{G}_k$ parameters, and therefore convergence of all $\mathbf{W}_k$ filters for $k = 1 \dots J$. Optimality of the resulting filters can be proven using the same arguments as in the optimality proof of theorem 4.1 for DANSE$_K$ in [13]. □

4.6 Performance of DANSE and R-DANSE

In this section, the batch mode performance of DANSE and R-DANSE is compared for the acoustic scenario of Section 4.3. In this batch version of the algorithms, all iterations of DANSE and R-DANSE are on the full signal length of about 20 seconds. In real-life applications however, iterations will of course be spread over time, i.e. subsequent iterations are performed on different signal segments. To isolate the influence of VAD errors, an ideal VAD is used in all experiments. Correlation matrices are estimated by time averaging over the complete length of the signal. The sampling frequency is 32 kHz and the DFT size is equal to $L = 512$ if not specified otherwise.

4.6.1 Experimental Validation of DANSE and R-DANSE

Three different measures are used to assess the quality of the outputs at the hearing aids: the signal-to-noise ratio (4.6), the signal-to-distortion ratio (4.7), and the mean squared error (MSE) between the coefficients of the optimal multichannel Wiener filter $\hat{\mathbf{w}}_k$ and the filter obtained by the DANSE algorithm, i.e.

$$\text{MSE} = \frac{1}{L}\|\hat{\mathbf{w}}_k - \mathbf{w}_k(1)\|^2 \tag{4.33}$$

with L the DFT size, $\hat{\mathbf{w}}_k$ defined by (4.4), and $\mathbf{w}_k(1)$ denoting the first column of $\mathbf{W}_k$ in (4.12), i.e. the filter that estimates the speech component x_{k1} in the reference microphone at node k.

Two different scenarios are tested. In **scenario 1** the dimension Q of the desired signal space is $Q = 1$, i.e. both hearing aid users are listening to speaker C, whereas speakers A and B and the babble-noise loudspeaker are considered to be background noise. In Fig. 4.6, the three quality measures are plotted (for node 4) versus the iteration index for DANSE$_1$ and R-DANSE$_1$, with either sequential updating or simultaneous updating (without relaxation).

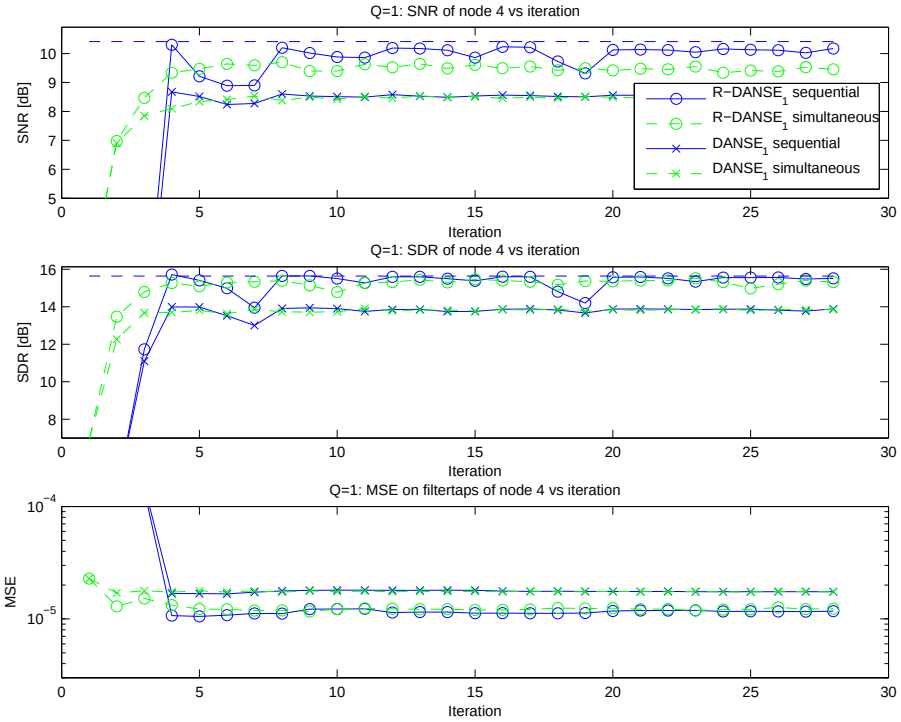


Figure 4.6: **Scenario 1**: SNR, SDR and MSE on filtertaps vs. iterations for DANSE_1 and R-DANSE_1 at node 4, for both sequential and simultaneous updates. Speaker C is the only target speaker.

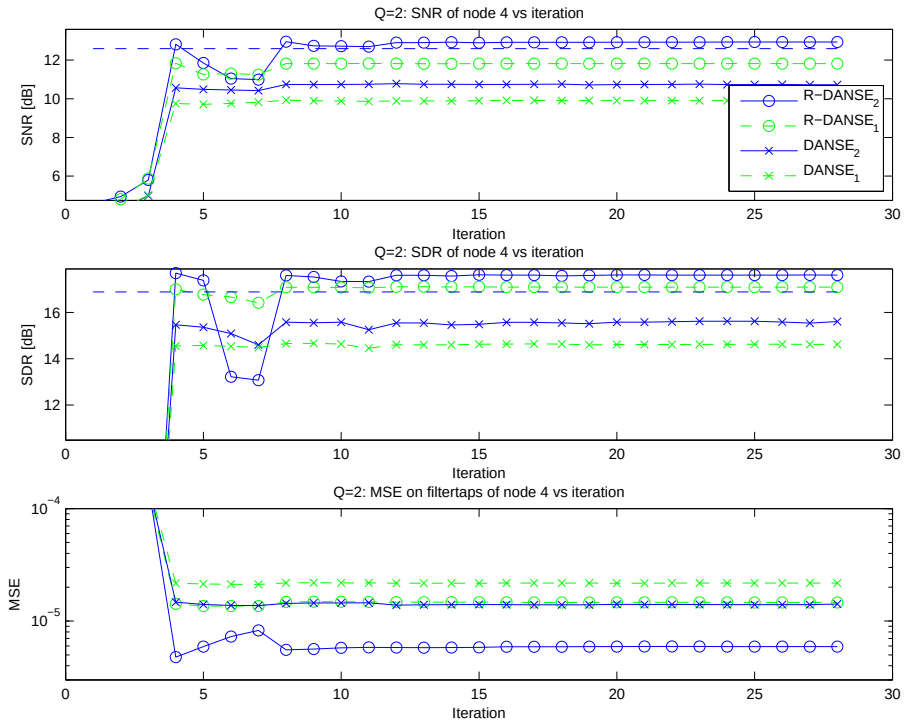


Figure 4.7: **Scenario 2**: SNR, SDR and MSE on filtertaps vs. iterations for DANSE$_1$, R-DANSE$_1$, DANSE$_2$ and R-DANSE$_2$ at node 4. Speakers B and C are target speakers.

Also an upper bound is plotted, which corresponds to the centralized MWF solution defined in (4.4). The R-DANSE$_1$ graph consists of only DEF nodes, except for $\mathbf{w}_{11}$, which has an arc $(\mathbf{w}_{11}, \mathbf{w}_{44})$ to avoid performance loss due to low SNR. Since there is only one desired source, DANSE$_1$ theoretically should converge to the upper bound performance, but this is not the case. The R-DANSE$_1$ algorithm performs better than the DANSE$_1$ algorithm, yielding an SNR increase of 1.5 to 2 dB, which is an increase of about 20% to 25%. The same holds for the other two hearing aids, i.e. node 2 and 3, which are not shown here. The simultaneous update typically converges faster but it converges to a suboptimal limit cycle, since no relaxation is used. Although this limit cycle is not very clear in these plots, a loss in SNR of roughly 1 dB is observed in every hearing aid. This can be avoided by using relaxation, which will be illustrated in Section 4.6.2.

In **scenario 2**, the case in which $Q = 2$ is considered, i.e. there are two desired sources: both hearing aid users are listening to speakers B and C, who talk

simultaneously, yielding a speech correlation matrix $\mathbf{R}_{xx}$ of approximately rank 2. The R-DANSE$_2$ graph is illustrated in Fig. 4.5. For this 2-speaker case, both DANSE$_1$ and DANSE$_2$ are evaluated (with sequential node-updating), where the latter should theoretically converge to the upper bound performance. The results for node 4 are plotted in Fig. 4.7. While the MSE is lower for DANSE$_2$ compared to DANSE$_1$, it is observed that DANSE$_2$ does not reach the optimal noise reduction performance. R-DANSE$_2$ is however able to reach the upper bound performance at every hearing aid. The SNR improvement of R-DANSE$_2$ in comparison with DANSE$_2$ is between 2 and 3 dB at every hearing aid, which is again an increase of about 20% to 25%. Notice that R-DANSE$_2$ even slightly outperforms the centralized algorithm. This may be because R-DANSE$_2$ performs its matrix inversions on correlation matrices with smaller dimensions than the all-microphone correlation matrix $\mathbf{R}_{yy}$ in the centralized algorithm, which is more favorable in a numerical sense.

4.6.2 Simultaneous Updating with Relaxation

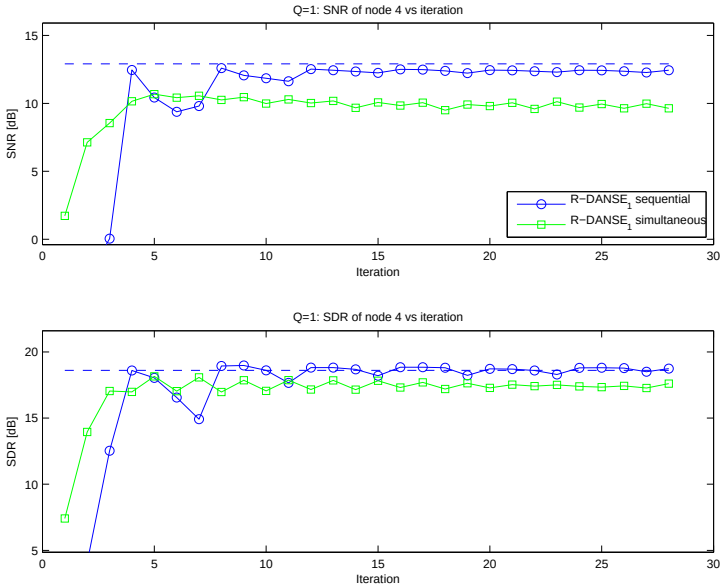
Simulations on different acoustic scenarios show that in most cases, DANSE$_K$ with simultaneous updating results in a limit cycle oscillation. The occurrence of limit cycles appears to depend on the position of the nodes and sound sources, the reverberation time, as well as on the DFT size, but no clear rule was found to predict the occurrence of a limit cycle.

To illustrate the effect of relaxation, the simulation results of R-DANSE$_1$ in the scenario of Section 4.3 are given in Fig. 4.8(a), where now the DFT size is $L = 1024$, which results in clearly visible limit cycle oscillations when no relaxation is used. This causes an overall loss in SNR of 2 or 3 dB at every hearing aid.

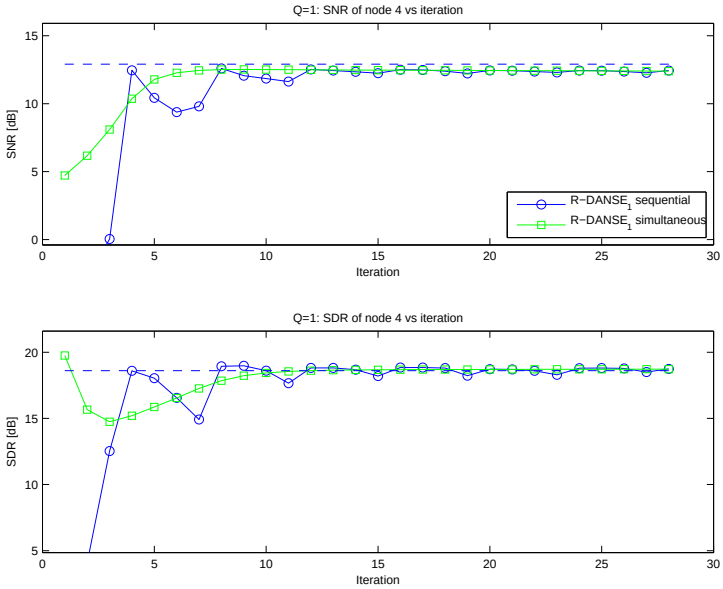
Fig. 4.8(b) shows the same experiment where relaxation is used as in formula (4.20) with $\alpha^i = 0.5$, $\forall\, i \in \mathbb{N}$. In this case, the limit cycle does not appear and the simultaneous updating algorithm indeed converges to the same values as the sequential updating algorithm. Notice that the simultaneous updating algorithm converges faster than the sequential updating algorithm.

4.6.3 DFT Size

In Fig. 4.9, the SNR and SDR of the output signal of R-DANSE$_1$ at nodes 3 and 4 is plotted as a function of the DFT size L, which is equivalent to the length of the time domain filters that are implicitly applied to the signals at the nodes. 28 iterations were performed with sequential updating for $L = 256$, $L = 512$, $L = 1024$, and $L = 2048$. The outputs of the centralized version and the scenario in which nodes do not share any signals, are also given as a reference.

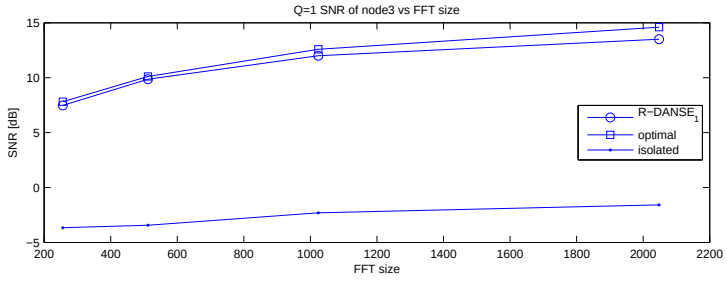


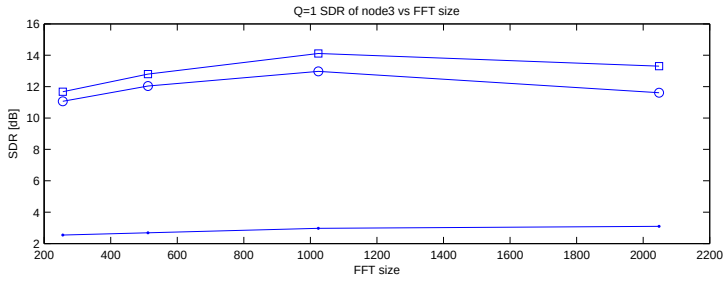
(a) without relaxation



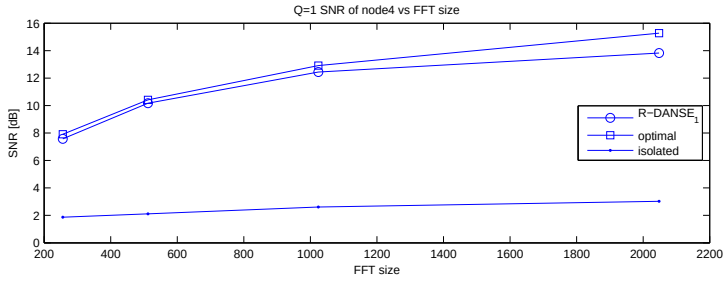
(b) with relaxation ($\alpha^i = 0.5$, $\forall\, i \in \mathbb{N}$)

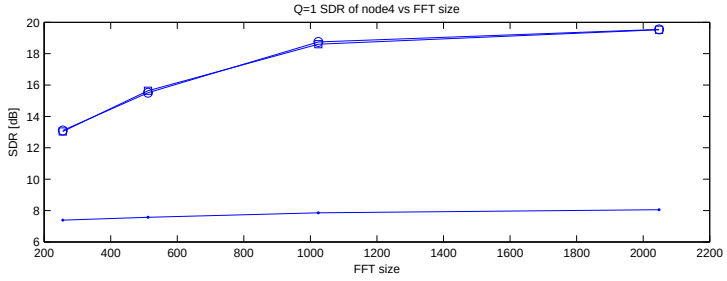
Figure 4.8: SNR and SDR for R-DANSE$_1$ vs. iterations at node 4 with sequential and simultaneous updating.





(a) node3





(b) node 4

Figure 4.9: Output SNR and SDR after 28 iterations of R-DANSE$_1$ with sequential updating vs. DFT size L at nodes 3 and 4.

As expected, the performance increases with increasing DFT size. However, the discrepancy between the centralized algorithm and R-DANSE$_1$ grows for increasing DFT size. One reason for this observation is that, for large DFT sizes, R-DANSE often converges slowly once the filters at all nodes are close to the optimal filters.

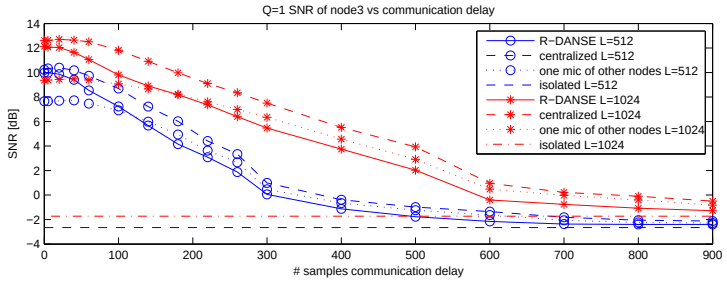
The scenario with isolated nodes is less sensitive to the DFT size. This is because the tested DFT sizes are quite large, yielding long filters. As explained in the next section, shorter filter lengths are sufficient in the case of isolated nodes since the microphones are very close to each other, yielding small time differences of arrival (TDOA).

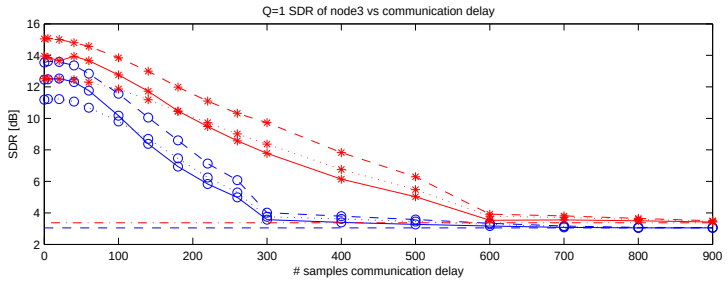
4.6.4 Communication Delays or Time Differences of Arrival

To exploit the spatial coherence between microphone signals, the noise reduction filters attempt to align the signal components resulting from the same source in the different microphone signals. However, alignment of the direct components of the source signals is only possible when the filter lengths are at least twice the maximum time difference of arrival (TDOA) between all the microphones. This means that in general, the noise reduction performance degrades with increasing TDOA's and fixed filter lengths. Large TDOA's require longer filters, or appropriate delay compensation. As already mentioned in Section 4.3, delay compensation is restricted in hearing aids due to lip synchronization constraints.

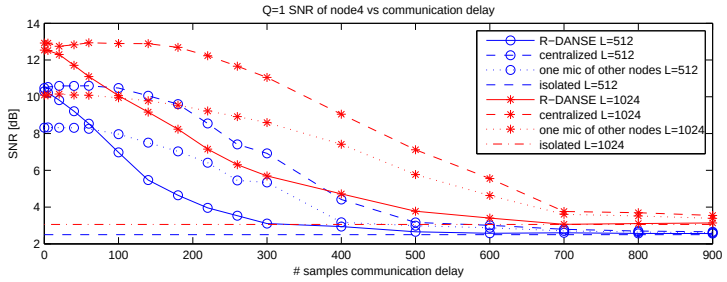
The TDOA depends on the distance between the microphones, the position of the sources and the delay introduced by the communication link. Fig. 4.10 shows the performance degradation of R-DANSE at nodes 3 and 4 when the TDOA increases, in this case modelled by an increasing communication delay between the nodes. There is no delay compensation, i.e. none of the signals are delayed before filtering. DFT sizes $L = 512$ and $L = 1024$ are evaluated. The outputs of the centralized MWF procedure are also given as a reference, as well as the procedure where every node broadcasts its first microphone signal, which corresponds to the scenario in which all supporting nodes are single-microphone nodes. The lower bound is defined by the scenario where all nodes are isolated, i.e. each node only uses its own microphones in the estimation process.

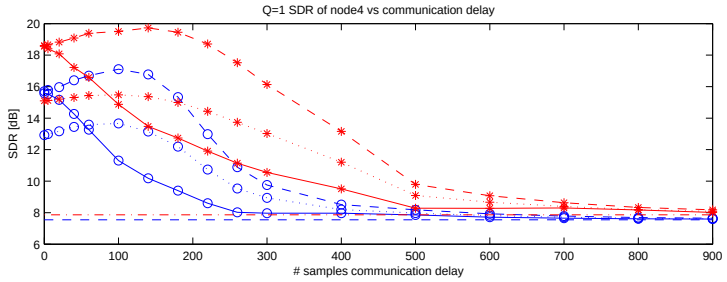
As expected, when the communication delay increases, the performance degrades due to increasing time lags between signals. At node 3, the R-DANSE algorithm is slightly more sensitive to the communication delay than the centralized MWF. The behavior at node 2 is very similar, and is omitted here. Furthermore, for large communication delays, R-DANSE is outperformed by the single-microphone nodes scenario. At node 4, both the centralized MWF





(a) Node 3





(b) Node 4

Figure 4.10: Output SNR and SDR at nodes 3 and 4 after 12 iterations of R-DANSE$_1$ with sequential updating vs. delay of the communication link.

and the single-microphone nodes scenario even benefit from communication delays. Apparently, the additional delay allows the estimation process to align the signals more effectively.

The reason why R-DANSE is more sensitive to a communication delay than the centralized MWF is that the latter involves independent estimation processes, whereas in R-DANSE, the estimation at any node k depends on the quality of estimation at every other node $q \neq k$. Notice however that the influence of communication delay is of course very dependent on the scenario and its resulting TDOA's. The above results only give an indication of this influence.

4.7 Practical Issues and Open Problems

In the batch-mode simulations provided in this paper, some practical aspects have been disregarded. Therefore, the actual performance of the MWF and the DANSE$_K$ algorithm may be worse than what is shown in the simulations. In this section, some of these practical aspects are briefly discussed.

The VAD is a crucial ingredient in MWF-based noise reduction applications. A simple VAD may not behave well in the simulated scenario as described in Fig. 4.2 due to the fact that the noise component also contains competing speech signals. Especially the VAD's at nodes that are close to an interfering speech source (e.g. node 1 in Fig. 4.2) are bound to make many wrong decisions, which will then severely deteriorate the output of the DANSE algorithm. To solve this, a speaker selective VAD should be used, e.g. [24]. Also, low SNR nodes should be able to use VAD information from high SNR nodes. By sharing VAD information, better VAD decisions can be made [25]. How to organize this, and how a consensus decision can be found between different nodes, is still an open research problem.

A related problem is the actual selection of the desired source, versus the noise sources. A possible strategy is that the speech source with the highest power at a certain reference node is selected as the desired source. In hearing aid applications, it is often assumed that the desired source is in front of the listener. Since the actual positions of the hearing aid microphones are known (to a certain accuracy), the VAD can be combined with a source localization algorithm or a fixed beamformer to distinguish between a target speaker and an interfering speaker. Again, this information should be shared between nodes so that all nodes can eventually make consistent selections.

A practical aspect that needs special attention is the adaptive estimation of the correlation matrices in the DANSE$_K$ algorithm. In many MWF implementations, correlation matrices are updated with the instantaneous sample

correlation matrix and by using a forgetting factor $0 < \lambda < 1$, i.e.

$$\mathbf{R}_{yy}[t] = \lambda \mathbf{R}_{yy}[t-1] + (1-\lambda)\mathbf{y}[t]\mathbf{y}^H[t] \tag{4.34}$$

where $\mathbf{y}[t]$ denotes the sample of the multi-channel signal $\mathbf{y}$ at time t. The forgetting factor λ is chosen close to 1 to obtain long-term estimates that mainly capture the spatial coherence between the microphone signals. In the DANSE$_K$ algorithm, however, the statistics of the input signal $\tilde{\mathbf{y}}_k$ in node k, defined by (4.14), change whenever a node $q \neq k$ updates its filters, since some of the channels in $\tilde{\mathbf{y}}_k$ are indeed outputs from a filter in node q. Therefore, when node q updates its filters, parts of the estimated correlation matrices $\widetilde{\mathbf{R}}_{yy,k}$ and $\widetilde{\mathbf{R}}_{xx,k}$, $\forall\, k \in \{1,\ldots,J\}\backslash\{q\}$, may become invalid. Therefore, strategy (4.34) may not work well, since every new estimate of the correlation matrix then relies on previous estimates. Instead, either downdating strategies should be considered, or the correlation matrices have to be completely recomputed.

4.8 Conclusions

The simulation results described in this paper demonstrate that noise reduction performance in hearing aids may be significantly improved when external acoustic sensor nodes are added to the estimation process. Moreover, these simulation results provide a proof-of-concept for applying DANSE$_K$ in cooperative acoustic sensor networks for distributed noise reduction applications, such as in hearing aids. A more robust version of DANSE$_K$, referred to as R-DANSE$_K$, has been introduced and convergence has been proven. Batch-mode experiments showed that R-DANSE$_K$ significantly outperforms DANSE$_K$. The occurrence of limit cycles and the effectiveness of relaxation in the simultaneous updating procedure has been illustrated. Additional tests have been performed to quantify the influence of several parameters, such as the DFT size and TDOA's or delays within the communication link.

Bibliography

[1] H. Dillon, *Hearing Aids*. Boomerang Press, 2001.

[2] B. Kollmeier, J. Peissig, and V. Hohmann, "Real-time multiband dynamic compression and noise reduction for binaural hearing aids," *Journal of Rehabilitation Research and Development*, vol. 30, pp. 82–94, 1993.

[3] J. Desloge, W. Rabinowitz, and P. Zurek, "Microphone-array hearing aids with binaural output–Part I: Fixed-processing systems," *IEEE Trans. Speech and Audio Processing*, vol. 5, pp. 529–542, 1997.

[4] D. Welker, J. Greenberg, J. Desloge, and P. Zurek, "Microphone-array hearing aids with binaural output–Part II: A two-microphone adaptive system," *IEEE Trans. Speech and Audio Processing*, vol. 5, pp. 543–551, November 1997.

[5] I. Merks, M. Boone, and A. Berkhout, "Design of a broadside array for a binaural hearing aid," *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, October 1997.

[6] V. Hamacher, "Comparison of advanced monaural and binaural noise reduction algorithms for hearing aids," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4008–4011, May 2002.

[7] R. Nishimura, Y. Suzuki, and F. Asano, "A new adaptive binaural microphone array system using a weighted least squares algorithm," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1925–1928, May 2002.

[8] T. Wittkop and V. Hohmann, "Strategy-selective noise reduction for binaural digital hearing aids," *Speech Communication*, vol. 39, pp. 111–138, January 2003.

[9] M. Lockwood, D. Jones, R. Bilger, C. Lansing, W. O'Brien, B. Wheeler, and A. Feng, "Performance of time- and frequency-domain binaural beamformers based on recorded signals from real rooms," *Journal of the Acoustical Society of America*, vol. 115, pp. 379–391, January 2004.

[10] T. Lotter and P. Vary, "Dual-channel speech enhancement by superdirective beamforming," *EURASIP Journal on Applied Signal Processing*, 2006.

[11] O. Roy and M. Vetterli, "Rate-constrained beamforming for collaborating hearing aids," *Proc. International Symposium on Information Theory (ISIT)*, pp. 2809–2813, July 2006.

[12] S. Doclo and M. Moonen, "GSVD-based optimal filtering for single and multimicrophone speech enhancement," *IEEE Transactions on Signal Processing*, vol. 50, pp. 2230–2244, Sep. 2002.

[13] 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, Oct. 2010.

[14] ——, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2053–2056, April 2009.

[15] T. Klasen, T. Van den Bogaert, M. Moonen, and J. Wouters, "Binaural noise reduction algorithms for hearing aids that preserve interaural time delay cues," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1579–1585, April 2007.

[16] S. Doclo, R. Dong, T. Klasen, J. Wouters, S. Haykin, and M. Moonen, "Extension of the multi-channel Wiener filter with localisation cues for noise reduction in binaural hearing aids," *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC)*, pp. 221–224, Sept. 2005.

[17] S. Doclo, T. 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," *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC)*, Sept. 2006.

[18] T. Van den Bogaert, S. Doclo, M. Moonen, and J. Wouters, "Binaural cue preservation for hearing aids using an interaural transfer function multichannel Wiener filter," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 565–568, April 2007.

[19] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, pp. 38–51, Jan. 2009.

[20] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," *Journal of the Acoustical Society of America*, vol. 65, pp. 943–950, April 1979.

[21] M. Nilsson, S. Soli, and A. Sullivan, "Development of the Hearing in Noise Test for the measurement of speech reception thresholds in quiet and in noise," *Journal of the Acoustical Society of America*, vol. 95, pp. 1085–1099, Feb. 1994.

[22] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part II: Simultaneous and asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5292 –5306, Oct. 2010.

[23] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*. American Elsevier Publishing Company, Inc., New York, New York.

[24] S. Maraboina, D. Kolossa, P. Bora, and R. Orglmeister, "Multi-speaker voice activity detection using ICA and beampattern analysis," *Proc. of the European signal processing conference (EUSIPCO)*, 2006.

[25] V. Berisha, H. Kwon, and S. Spanias, "Real-time implementation of a distributed voice activity detector," in *Proc. IEEE Workshop on Sensor Array and Multichannel Processing*, July 2006, pp. 659–662.

Chapter 5

DANSE in Networks with a Tree Topology

Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology

Alexander Bertrand and Marc Moonen

Published in *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2196 - 2210, May 2011.

Contributions of first author

- literature study
- co-analysis of feedback in simply connected networks
- co-development of the T-DANSE$_K$ algorithm
- co-establishment of proof of convergence and optimality of T-DANSE$_K$
- co-development of TFC and RFC concepts
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

We present a distributed adaptive node-specific signal estimation (DANSE) algorithm that operates in a wireless sensor network with a tree topology. The algorithm extends the DANSE algorithm for fully connected sensor networks, as described in previous work. It is argued why a tree topology is the natural choice if the network is not fully connected. If the node-specific desired signals share a common latent signal subspace, it is shown that the distributed algorithm converges to the same linear MMSE solutions as obtained with the centralized version of the algorithm. The computational load is then shared between the different nodes in the network, and nodes exchange only linear combinations of their sensor signal observations and data received from their neighbors. Despite the low connectivity of the network and the multi-hop signal paths, the algorithm is fully scalable in terms of communication bandwidth and computational power. Two different cases are considered concerning the communication protocol between the nodes: point-to-point transmission and local broadcasting. The former assumes that there is a reserved communication link between node-pairs, whereas with the latter, nodes communicate the same data to all of their neighbors simultaneously. The convergence properties of the algorithm are demonstrated by means of numerical examples.

5.1 Introduction

A wireless sensor network (WSN) consists of sensor nodes that cooperate to perform a certain task, such as the estimation of a parameter or signal, where data is shared between nearby nodes through a wireless link. A general objective is to utilize all available data throughout the network, possibly through a fusion center that gathers all sensor signal observations and performs all computations. However, in many cases a distributed approach is preferred, which is scalable with respect to both communication resources and computational power. In this case, data diffuses through the network and each node contributes to the processing (e.g. [1–4]).

In this paper, we consider distributed signal estimation, rather than parameter estimation. This means that the number of variables to estimate grows linearly with the number of temporal observations, i.e. for each sample time of the sensors, a new sample of the desired signal(s) needs to be estimated. The estimation then usually relies on linear spatial filtering or beamforming, as often used in signal enhancement [5–8]. In parameter estimation problems on the other hand, the number of estimation variables is fixed, i.e. it does not grow with the number of temporal observations [1–4, 9, 10]. Usually, intermediate estimates are then exchanged and iteratively improved over time, often without exchanging the actual sensor observations. In the case of distributed

beamforming or signal estimation, (fused or compressed) signal observations are exchanged between nodes, and then the aim is to iteratively improve the local fusion rules to better estimate future samples. These type of WSN's are usually smaller in size, and operate with a larger bandwidth and energy consumption compared to traditional large-scale WSN's [7, 8], especially in applications with high sampling rates. They are often also assumed to be more robust, especially in real-time systems, since packet loss can then result in instantaneous signal degradation. Therefore, in this paper, we assume that the communication links are ideal, i.e. they are not subject to noise or random failures.

In [11, 12], a distributed adaptive node-specific signal estimation (DANSE) algorithm is presented, based on linear MMSE estimation. It operates in a fully connected sensor network where each node has access to multi-channel sensor signal observations. The term 'node-specific' refers to the fact that each node estimates a different desired signal. Node-specific estimation is relevant in cases where inherent spatial information in the local observations of the target sources needs to be preserved during the estimation, e.g. to serve as an input for a localization algorithm, or in collaborating hearing aids when the aim is to also preserve the cues for directional hearing [13]. Due to the linearity of the proposed centralized estimator, the algorithm can be made distributed and it significantly compresses the signals that are communicated between nodes, provided that the desired signals of the different nodes share a common low dimensional latent signal subspace (which is assumed to be unknown). Although the nodes broadcast only a few linear combinations of their sensor signal observations, the DANSE algorithm provides linear minimum mean squared error (MMSE) estimates as if all data were available at each node. In [14], the DANSE algorithm is extended to also guarantee convergence when nodes update simultaneously or asynchronously, which generally results in faster tracking. In [7], a more robust version of DANSE has been formulated, referred to as R-DANSE. Simulations in [7, 8] illustrate the potential of the algorithm for distributed speech enhancement in acoustic sensor networks.

A limitation of the DANSE algorithm in [11] is that the network is assumed to be fully connected, which is only possible in practice if the nodes have sufficient transmission power, and if the available communication bandwidth is large enough. Furthermore, the amount of data that is received and processed by each node increases with the number of nodes in the network. In this paper, we modify the DANSE algorithm, such that it can operate in a network with a tree topology, avoiding the aforementioned issues. We refer to this algorithm as tree-DANSE or T-DANSE. The choice for a tree topology is justified by the fact that it contains no cycles, and hence does not introduce any feedback paths. We will explain that feedback paths harm the convergence and optimality of the DANSE algorithm. The formulation of T-DANSE was briefly introduced in [15], without a theoretical analysis. In this paper, we provide more details,

and include a convergence and optimality proof.

In the T-DANSE algorithm, the signal observations of the different nodes are fused in a decentralized way, i.e. each node linearly combines its own sensor signal observations with data obtained from its neighbors before forwarding them to the next node. Remarkably, despite this distributed fusion process, each node is able to optimally estimate its own node-specific signal as if all data of the complete network were available. The local estimation task at each node is equivalent to the centralized estimation problem, but at a much smaller scale, i.e. with a drastically reduced amount of signals.

As opposed to fully connected DANSE, the number of signals that need to be processed by a node in T-DANSE does not depend on the total number of nodes in the network, but only on the number of neighbors of that node. This is important since the number of signals that a single node can receive and process in real-time is usually limited, especially when operating at large sampling rates. An additional advantage of using multi-hop networks, is the fact that nodes can transmit with lower power, and it enables spatial reuse of the spectrum. Furthermore, even when nodes only have access to observations of a single-channel sensor signal, the T-DANSE algorithm yields a benefit in terms of communication bandwidth efficiency, whereas fully connected DANSE is only useful if the number of sensor signals per node is larger than the dimension of the latent signal subspace that contains the desired signals [11].

We will consider two different communication protocols: point-to-point transmission and local broadcasting. The former assumes that there is a reserved communication link between node-pairs, whereas with the latter, a node communicates the same data to multiple neighbors simultaneously. We will show that both cases can be treated equivalently from a theoretical point of view. However, the broadcast protocol is obviously more efficient in terms of communication bandwidth.

The outline of this paper is as follows. In Section 5.2, the general problem statement for distributed node-specific linear MMSE estimation is given. We briefly review the DANSE algorithm [11] for fully connected networks in Section 5.3, which will act as a starting point and which allows us to introduce some necessary notation[1]. In Section 5.4, we point out that feedback paths in the network topology harm the convergence and optimality of the DANSE algorithm. In Section 5.5, we introduce the T-DANSE algorithm in a network with a tree topology, and prove convergence and optimality. In Section 5.6, we explain how T-DANSE can be used in a communication protocol that supports local broadcasting. Section 5.7 provides some simulation results. Conclusions are given in Section 5.8.

[1] Although this paper does not assume prior knowledge on the fully connected DANSE algorithm, it is recommended to read [11] first, since it addresses a much simpler case, which allows the reader to get familiar with the notation, the problem statement, and the algorithm.

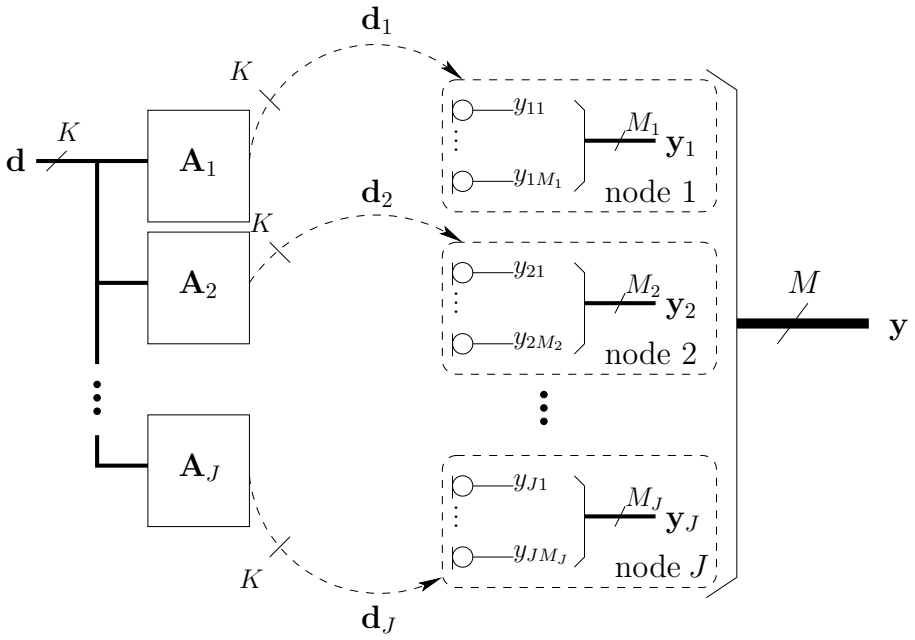


Figure 5.1: Description of the scenario. The network (with undefined topology) contains J sensor nodes, $k = 1 \dots J$, where node k collects M_k-channel sensor signal observations and estimates a node-specific desired signal $\mathbf{d}_k$, which is a mixture of the K channels of a common latent signal $\mathbf{d}$.

5.2 Problem Formulation and Notation

In this section, we briefly describe the data model and the problem statement. More details can be found in [11]. This section can be skipped if the reader is familiar with the set-up and the notation in [11].

5.2.1 Data Model

Consider a network with sensor nodes $\{1, \dots, J\} = \mathcal{J}$. At this point, we do not yet make any assumptions on the topology of this network. In the sequel, we assume that all mentioned signals are stationary and ergodic[2]. Sensor node k collects observations of a complex[3] valued random M_k-channel sensor signal

[2]In practice, the stationarity and ergodicity assumption can be relaxed to short-term stationarity and ergodicity, in which case the theory should be applied to finite signal segments that are assumed to be stationary and ergodic.

[3]Throughout this paper, all signals are assumed to be complex valued to permit frequency domain descriptions. In this case, multi-tap estimation is also covered, and the data model (5.1) then corresponds to a frequency domain description of a convolutive mixture.

$\mathbf{y}_k[t]$, where $t \in \mathbb{N}$ is the discrete sample time index. For the sake of an easy exposition, we will omit the time index when referring to a signal, and we will only write the time index when referring to one specific observation, i.e. $\mathbf{y}_k[t]$ is the observation of the signal $\mathbf{y}_k$ at time t. We define $\mathbf{y}$ as the M-channel signal in which all $\mathbf{y}_k$ are stacked, where $M = \sum_{k=1}^{J} M_k$. This scenario is depicted in Fig. 5.1. The different channels of the signal $\mathbf{y}_k$ may correspond to different sensors at node k (as it is the case in Fig. 5.1), or different delayed versions of its sensor signals to exploit temporal information.

The objective for each node k is to optimally estimate a node-specific K-channel desired signal $\mathbf{d}_k$ that is assumed to be correlated to $\mathbf{y}$. We consider the general case where $\mathbf{d}_k$ is not an observed signal, i.e. it is assumed to be unknown, as it is the case in signal enhancement. We assume that the node-specific desired signals $\mathbf{d}_k$ share a common K-dimensional latent signal subspace, defined by the channels of an unknown K-channel latent signal $\mathbf{d}$, i.e.

$$\mathbf{d}_k = \mathbf{A}_k \mathbf{d}, \ \forall\, k \in \mathcal{J} \tag{5.1}$$

with $\mathbf{A}_k$ a full rank $K \times K$ matrix with unknown coefficients[4]. It is noted that we assume (without loss of generality) that the number of channels of the desired signals $\mathbf{d}_k$, $\forall\, k \in \mathcal{J}$, is equal to the dimension of the latent signal subspace defined by $\mathbf{d}$. In many practical cases, only a subset of the channels of $\mathbf{d}_k$ may be of actual interest, in which case the other channels should be seen as auxiliary channels to capture the entire K-dimensional signal subspace defined by $\mathbf{d}$ (the reason for this will be explained later).

To make this more concrete, we give an example in the context of noise reduction for speech enhancement that fits the aforementioned data model. Assume a scenario with K speech sources, stacked in the signal $\mathbf{d}$. The observed signals at the M_k sensors (i.e. microphones) of node k are then described by the linear sensor data model

$$\mathbf{y}_k = \mathbf{U}_k \mathbf{d} + \mathbf{n}_k \tag{5.2}$$

with $\mathbf{U}_k$ an (unknown) $M_k \times K$ steering matrix to the M_k microphones of node k, and $\mathbf{n}_k$ a noise component containing point-source interferers, diffuse noise and sensor noise. Note that (5.2) is a frequency domain representation, transforming the convolutive acoustic mixture of the speech signals in an instantaneous mixture. The goal of each node is to estimate the mixture of the signals in $\mathbf{d}$ observed at one of their microphones[5], referred to as the reference microphone. If $K > 1$, additional auxiliary reference microphones need to

[4]It is noted that node-specific estimation also exists in a distributed parameter estimation context, e.g. in random-field estimation [9, 10]. However, usually it is assumed that the covariance or other parameters describing the depencies between the hidden node-specific variables are known. This is not the case in our approach, i.e. we do not know the $\mathbf{A}_k$'s or the cross-correlation between the $\mathbf{d}_k$'s. We only exploit the prior knowledge that the $\mathbf{d}_k$'s share a common laten signal subspace, but we do not know anything about this subspace, except for its dimension.

[5]This is the best one can do in a blind approach, i.e. when there is neither knowledge

be selected to obtain a K-dimensional node-specific desired signal $\mathbf{d}_k$. If the first K microphones are selected in each node, then $\mathbf{A}_k$ in (5.1) corresponds to the first K rows of $\mathbf{U}_k$ in (5.2). For more information regarding this acoustic application, we refer to [7].

We emphasize that expression (5.2) is given here as an example, and we do not restrict ourselves to this sensor data model in the design of the algorithms in the sequel.

5.2.2 Centralized Linear MMSE Estimation

We first consider the centralized estimation problem, i.e. we assume that all nodes have access to observations of all M channels of $\mathbf{y}$. Node k uses a linear estimator $\mathbf{W}_k$ to estimate $\mathbf{d}_k$ as

$$\overline{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y} \tag{5.3}$$

where $\mathbf{W}_k$ is a complex $M \times K$ matrix, and where superscript H denotes the conjugate transpose operator. This is similar to beamforming frameworks [5], where multiple signals are linearly combined to generate an output signal with suppressed interferers and background noise. We consider linear MMSE estimation (similar to multi-channel Wiener filtering [6]) based on a node-specific estimator $\hat{\mathbf{W}}_k$, i.e.

$$\hat{\mathbf{W}}_k = \arg\min_{\mathbf{W}_k} E\{\|\mathbf{d}_k - \mathbf{W}_k^H \mathbf{y}\|^2\} \,, \tag{5.4}$$

where $E\{.\}$ denotes the expected value operator. Assuming that the correlation matrix $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$ has full rank, the solution of (5.4) is

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^{-1} \mathbf{R}_{yd_k} \tag{5.5}$$

with $\mathbf{R}_{yd_k} = E\{\mathbf{y}\mathbf{d}_k^H\}$. Based on the assumption that the signals are ergodic, $\mathbf{R}_{yy}$ and $\mathbf{R}_{yd_k}$ can be estimated by time averaging. The $\mathbf{R}_{yy}$ is estimated from the sensor signal observations. Since $\mathbf{d}_k$ is assumed to be unknown, $\mathbf{R}_{yd_k}$ has to be estimated indirectly. A possible way to estimate $\mathbf{R}_{yd_k}$, is to periodically transmit training sequences, or by exploiting the on-off behavior of the desired signal, e.g. in speech enhancement applications [7, 16]. More information on the estimation of $\mathbf{R}_{yd_k}$ can be found in [11]. In the sequel we will assume that $\mathbf{R}_{yd_k}$, or its distributed variants, can be estimated adaptively during operation of the algorithm.

In the distributed case, each node k only has access to observations of $\mathbf{y}_k$ which is a subset of the channels of the full signal $\mathbf{y}$. Therefore, to find the

about $\mathbf{d}$ nor the mixing system. The desired signal is then the observed mixture of target sources at the local sensors. This technique is often used in noise reduction applications for speech enhancement [6–8].

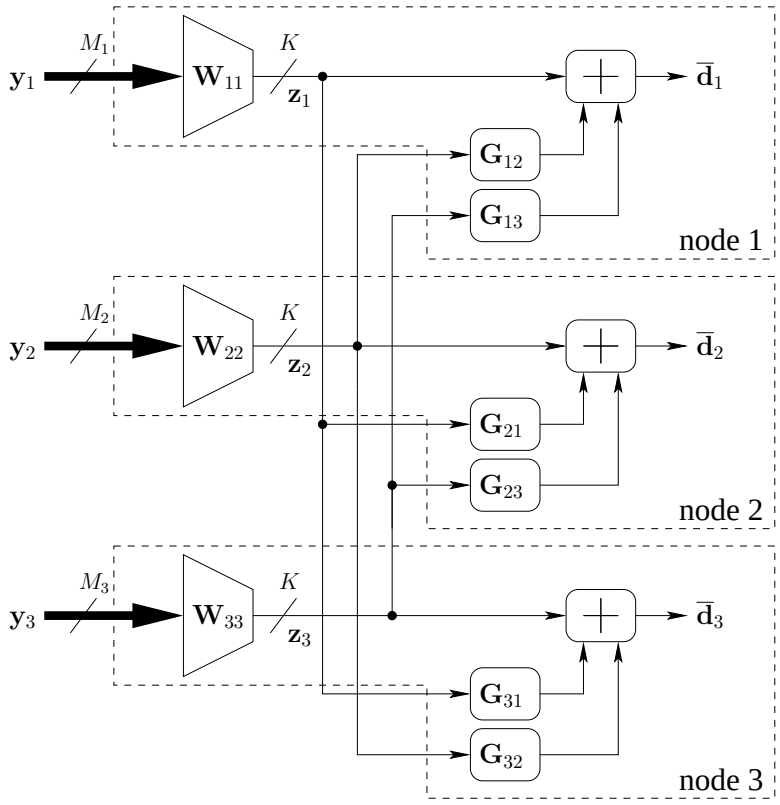


Figure 5.2: The DANSE$_K$ scheme with 3 nodes ($J = 3$). Each node k estimates a signal $\mathbf{d}_k$ using its own M_k-channel sensor signal observations, and 2 compressed K-channel signal observations broadcast by the other two nodes.

optimal MMSE solution (5.5) in each node, the observations of $\mathbf{y}_k$ in principle have to be communicated to all nodes in the network, which requires a large communication bandwidth, especially if the network is not fully connected, i.e. if multi-hop transmission is required. As shown in the sequel, due to the linearity of the centralized estimator (5.5) and the low dimension of the latent signal subspace, we are still able to obtain the linear MMSE solution (5.5) at each node, even when the data communicated by the nodes is significantly compressed. We assume ideal communication links, i.e. they are not subject to noise or random failures. The issue of link failures in real-time signal estimation in wireless networks is briefly addressed in [17].

5.3 The DANSE Algorithm in a Fully Connected Network

In this section, we briefly review the DANSE$_K$ algorithm[6] in a fully connected sensor network, as introduced in [11, 12]. This is important to introduce some notation, and to grasp the underlying idea on how we can distribute each estimator over different nodes. In Section 5.5, we will extend this framework for signal estimation in networks with a tree topology.

We define a partitioning of the matrix $\mathbf{W}_k$ as $\mathbf{W}_k = [\mathbf{W}_{k1}^T \ \dots \ \mathbf{W}_{kJ}^T]^T$ where $\mathbf{W}_{kq} \in \mathbb{C}^{M_k \times K}$ is the part of $\mathbf{W}_k$ that is applied to the sensor signal observations of $\mathbf{y}_q$. The equivalent of (5.4) is then

$$\hat{\mathbf{W}}_k = \begin{bmatrix} \hat{\mathbf{W}}_{k1} \\ \hat{\mathbf{W}}_{k2} \\ \vdots \\ \hat{\mathbf{W}}_{kJ} \end{bmatrix} = \underset{\{\mathbf{W}_{k1},\dots,\mathbf{W}_{kJ}\}}{\arg\min} \ E\{\|\mathbf{d}_k - \sum_{q=1}^{J} \mathbf{W}_{kq}^H \mathbf{y}_q\|^2\} \ . \tag{5.6}$$

In the DANSE$_K$ algorithm, $\mathbf{y}_k$ is linearly compressed to a K-channel signal $\mathbf{z}_k$ (the compression rule will be defined later), which is then broadcast to the remaining $J-1$ nodes. We define the $(J-1)K$-channel signal $\mathbf{z}_{-k} = \left[\mathbf{z}_1^T \dots \mathbf{z}_{k-1}^T \mathbf{z}_{k+1}^T \dots \mathbf{z}_J^T\right]^T$. Node k then collects observations of its own sensor signals in $\mathbf{y}_k$ and the signals in $\mathbf{z}_{-k}$ obtained from the other nodes in the network. Similar to (5.4), node k can then compute the linear MMSE estimator with respect to these input signals, i.e.

$$\begin{bmatrix} \mathbf{W}_{kk} \\ \mathbf{G}_{k,-k} \end{bmatrix} = \underset{\mathbf{W}_{kk},\mathbf{G}_{k,-k}}{\arg\min} \ E\left\{\|\mathbf{d}_k - \left[\mathbf{W}_{kk}^H \ \mathbf{G}_{k,-k}^H\right] \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k} \end{bmatrix} \|^2\right\} \tag{5.7}$$

where $\mathbf{W}_{kk}$ is the part of the estimator that is applied to node k's own sensor signals in $\mathbf{y}_k$ and where

$$\mathbf{G}_{k,-k} = [\mathbf{G}_{k1}^T \ \dots \ \mathbf{G}_{k,k-1}^T \ \mathbf{G}_{k,k+1}^T \ \dots \ \mathbf{G}_{kJ}^T]^T$$

with $\mathbf{G}_{kq} \in \mathbb{C}^{K \times K}$ denoting the part of the estimator that is applied to $\mathbf{z}_q$. The linear compression rule that generates the broadcast signal $\mathbf{z}_k$ is then given by

$$\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k \ . \tag{5.8}$$

A schematic illustration of this scheme is shown in Fig. 5.2, for a network with $J = 3$ nodes. It is noted that $\mathbf{W}_{kk}$ both acts as a compressor and as a part of

[6] The subscript K refers to the number of channels in the broadcast signals of each node. To obtain the optimal estimators, this number should be equal to the dimension of the latent signal subspace defined by $\mathbf{d}$, as proven in [11].

the estimator $\mathbf{W}_k$. Based on Fig. 5.2, it can be seen that the parametrization of the $\mathbf{W}_k$ effectively applied at node k, to generate $\overline{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y}$, is then

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{kJ} \end{bmatrix} \tag{5.9}$$

where we assume that $\mathbf{G}_{kk} = \mathbf{I}_K$ with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix. Expression (5.9) defines a solution space for all $\mathbf{W}_k$, $k \in \mathcal{J}$, simultaneously, where node k can only control the parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_{k,-k}$. The following theorem explains how this parametrization is still able to provide an optimal signal estimate in each node. A similar result will be obtained in Section 5.5 for the case of tree topology networks.

Theorem 5.1 *If (5.1) holds, then the optimal estimators $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$, given in (5.5) are in the solution space defined by parametrization (5.9).*

Proof : Since $\mathbf{d}_k = \mathbf{A}_k\mathbf{d}$, and because of (5.5), we know that the following holds:

$$\forall\, k, q \in \mathcal{J} : \hat{\mathbf{W}}_k = \hat{\mathbf{W}}_q \mathbf{A}_{kq} \tag{5.10}$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H}\mathbf{A}_k^H$. The theorem is proven by comparing (5.10) with (5.9), and by setting $\mathbf{G}_{kq} = \mathbf{A}_{kq}$, $\forall\, q \in \mathcal{J}$. □

The DANSE$_K$ algorithm iteratively updates the parameters in (5.9), by letting each node k compute (5.7), $\forall\, k \in \mathcal{J}$, in a sequential round robin fashion[7]. It is noted that each node then essentially performs a similar task as in a centralized computation, but on a smaller scale, i.e. with less signals. In [11], it is proven that this procedure converges to the centralized linear MMSE estimators at all nodes, i.e. $\lim_{i\to\infty} \mathbf{W}_k^i = \hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$. It is noted that we are not directly interested in the $\mathbf{W}_k$'s, but rather in the estimated samples of the $\mathbf{d}_k$'s. The estimate of a sample $\mathbf{d}_k[t]$ of the desired signal $\mathbf{d}_k$ in node k at any point in the iterative process is computed as

$$\overline{\mathbf{d}}_k[t] = \mathbf{W}_{kk}^H \mathbf{y}_k[t] + \sum_{q\neq k} \mathbf{G}_{kq}^H \mathbf{z}_q[t] \; . \tag{5.11}$$

Remark I: It is assumed that the cross-correlations $E\{\mathbf{y}_k\mathbf{d}_k^H\}$ and $E\{\mathbf{z}_{-k}\mathbf{d}_k^H\}$ can be (re-)estimated during operation of the algorithm. As explained earlier, this is only possible in certain applications, e.g. when the target source has an on-off behaviour or when training sequences can be used. We will not elaborate on this issue here, and we refer to [11] instead for further details.

[7] Results where nodes update simultaneously are also available [14], but these are not addressed here.

Remark II: The iterative nature of the DANSE$_K$ algorithm may suggest that the same sensor signal observations are compressed and broadcast multiple times, i.e. once after every iteration. However, in practical applications, iterations are spread over time, which means that successive updates of the estimators use different sensor signal observations. By exploiting the (short-term) stationarity assumption, updated estimators are only used for new (future) observations. This means that the iterations are not performed on the same block of data, but only on the local fusion rules at the nodes. For a detailed non-batch description of the algorithm, we refer to [11].

5.4 DANSE in Simply Connected Networks with Cycles

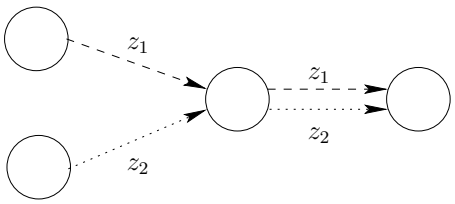
If the network is not fully connected, information must be passed from one side of the network to the other over multiple edges of the network graph. One can make the network virtually fully connected by letting nodes act as relays and so eventually provide every node with all observations of $\mathbf{z}_k$, as shown in Fig. 5.3(a). However, this is not scalable in terms of communication bandwidth and computational power, and the routing of the data streams can become very complex for large networks. A more desirable approach is to let each node transmit linear combinations of all its inputs, i.e. its own sensor signal observations as well as the data provided by other nodes, as shown in Fig. 5.3(b). We will first describe a straightforward fusion rule, and we will point out that this approach is problematic if the network contains cycles, since this introduces feedback components. We will then explain how this feedback can be avoided, which will lead to the tree-DANSE algorithm in Section 5.5.

5.4.1 A Straightforward Fusion Rule

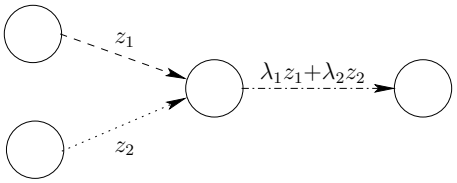
To pass information from node to node without increasing the communication bandwidth, one can apply the same DANSE$_K$ algorithm as in the previous section, but now let each node k transmit the observations of the K-channel signal

$$\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k + \sum_{q \in \mathcal{N}_k} \mathbf{G}_{kq}^H \mathbf{z}_q \tag{5.12}$$

to its neighbors, with $\mathcal{N}_k$ denoting the set of nodes that are connected to node k, node k excluded. The $\mathbf{z}_k$ signal is then a fused signal containing all the input signals of node k. Notice that this $\mathbf{z}_k$ corresponds to the node-specific estimated signal of node k, i.e. $\mathbf{z}_k = \overline{\mathbf{d}}_k$. This is illustrated in Fig. 5.4 for a 3-node network with a line topology. From this figure, it can be seen that the implicit definition of the $\mathbf{W}_k$, that is applied to $\mathbf{y}$ to generate $\overline{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y}$, is



(a) The relay case



(b) Linear combination of inputs

Figure 5.3: Two different types of data fusion in a network that is not fully connected.

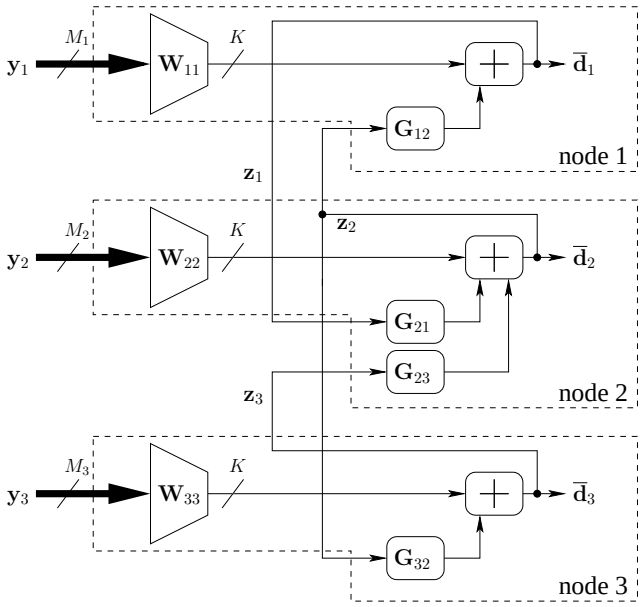


Figure 5.4: Visualization of parametrization (5.13) in a 3-node sensor network with a line topology.

then

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{O} \\ \mathbf{W}_{kk} \\ \mathbf{O} \end{bmatrix} + \sum_{q \in \mathcal{N}_k} \mathbf{W}_q \mathbf{G}_{kq} \ , \tag{5.13}$$

with $\mathbf{O}$ denoting an all-zero matrix of appropriate dimension. Parametrization (5.13) defines a solution space for all $\mathbf{W}_k$, $k \in \mathcal{J}$, simultaneously. We assume that the matrices $\mathbf{G}_{kq}$ are all-zero matrices, or do not exist, if there is no connection between node k and node $q \notin \mathcal{N}_k$.

In [15], it is pointed out that the parametrization (5.13) has an impact on the dynamics of the algorithm, but also on the solution space. Unfortunately, if (5.1) holds, i.e. if the desired signals share a K-dimensional latent signal subspace[8], the algorithm cannot obtain the optimal estimators (5.5), which was proven in [15] for a 2-node network. In the following theorem, we proof the general statement.

Theorem 5.2 *Consider a network with any topology. If (5.1) holds, then the optimal estimators $\hat{\mathbf{W}}_k$ given by (5.5) are not in the solution space defined by parametrization (5.13).*

Proof : We prove the theorem by contradiction, so we assume that the optimal centralized solution $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$, is in the solution space defined by (5.13). By substituting (5.10) in parametrization (5.13) for $k = 1$ (w.l.o.g.), we obtain

$$\begin{bmatrix} \mathbf{O}_{M_1 \times K} \\ \hat{\mathbf{W}}_{12} \\ \vdots \\ \hat{\mathbf{W}}_{1J} \end{bmatrix} = \sum_{q \in \mathcal{N}_k} \begin{bmatrix} \hat{\mathbf{W}}_{11} \\ \hat{\mathbf{W}}_{12} \\ \vdots \\ \hat{\mathbf{W}}_{1J} \end{bmatrix} \mathbf{A}_{q1} \mathbf{G}_{1q} \tag{5.14}$$

where $\mathbf{O}_{M_1 \times K}$ denotes an all-zero $M_1 \times K$ matrix. From the first M_1 rows in (5.14), we obtain

$$\mathbf{O}_{M_1 \times K} = \hat{\mathbf{W}}_{11} \sum_{q \in \mathcal{N}_k} \mathbf{A}_{q1} \mathbf{G}_{1q} \ . \tag{5.15}$$

From the last $M - M_1$ rows in (5.14), we obtain[9]

$$\mathbf{I}_K = \sum_{q \in \mathcal{N}_k} \mathbf{A}_{q1} \mathbf{G}_{1q} \ . \tag{5.16}$$

[8]Remarkably, if (5.1) does not hold, the solution space defined by parametrization (5.13) contains the same estimators as when using parametrization (5.9) [15]. However, the optimal solution (5.5) cannot be achieved in this case since it cannot be parametrized by (5.9).

[9]We implicitly assume that the submatrix $\left[\hat{\mathbf{W}}_{12}^T \ \dots \ \hat{\mathbf{W}}_{1J}^T\right]^T$ of the optimal centralized estimator given by (5.5), has full rank. Although this is not fully guaranteed by the imposed assumptions, this is satisfied in most practical cases since $M - M_1 \gg K$, and because $\mathbf{R}_{yy}^{-1}$ and the stacked $M_k \times K$ submatrices of $\mathbf{R}_{yd_k}$ in (5.5) have full rank (due to spatial diversity of the sensors).

Combining (5.15) and (5.16) yields $\hat{\mathbf{W}}_{11} = \mathbf{O}_{M_1 \times K}$, meaning that the sensor signals of node 1 are not used in the centralized solution (5.5). Instead of choosing $k = 1$, we can use a similar reasoning for all $k \in \mathcal{J}$ to eventually find that $\hat{\mathbf{W}}_k = \mathbf{O}_{M \times K}$, $\forall\, k \in \mathcal{J}$, which contradicts (5.5). □

The fundamental problem with parametrization (5.13) is the feedback in the signal paths. Indeed, the observations of $\mathbf{z}_q$ that node k receives from neighboring nodes $q \in \mathcal{N}_k$ also contain a component with the sensor signal observations $\mathbf{y}_k$ of node k itself. This results in a solution space for the DANSE$_K$ algorithm that does not contain the optimal estimators (5.5), as pointed out by Theorem 5.2.

We will distinguish between two forms of feedback: direct and indirect feedback. Direct feedback is caused by the feedback path from node k to a neighboring node q and back to node k, i.e. a cycle of length two. In Section 5.4.2, we show that this type of feedback can be easily controlled. Indirect feedback is more difficult to deal with. It occurs when data transmitted by node k travels through a path in the network, containing more than two different nodes, and eventually arrives back at node k. In Section 5.4.3, we will explain that indirect feedback can be avoided by removing direct feedback and by pruning the network to a tree topology.

5.4.2 Direct Feedback Cancellation

To avoid direct feedback, each node can send different data to each of its neighbors instead of locally broadcasting (5.12). Let $\mathbf{z}_{kq}$ denote the signal of which observations are transmitted from node k to node q, then direct feedback is avoided by choosing

$$\forall\, k \in \mathcal{J}, \forall\, q \in \mathcal{N}_k :\ \mathbf{z}_{kq} = \mathbf{W}_{kk}^H \mathbf{y}_k + \sum_{l \in \mathcal{N}_k \setminus \{q\}} \mathbf{G}_{kl}^H \mathbf{z}_{lk} \tag{5.17}$$

$$= \mathbf{z}_k - \mathbf{G}_{kq}^H \mathbf{z}_{qk}\ . \tag{5.18}$$

We will refer to this strategy as 'transmitter feedback cancellation' (TFC), since the direct feedback at node q is cancelled by the transmitting node k. We will refer to the signals defined in (5.17) as TFC-signals. It is noted that expression (5.17) provides an implicit definition of the TFC-signals, and that it is difficult to obtain a general closed form expression due to the remaining indirect feedback.

The TFC strategy matches perfectly with a point-to-point communication protocol, in which each individual node pair has a reserved communication link. In this case, direct feedback can be avoided without an increase in communication bandwidth. However, when the communication protocol supports local broadcasting, a more efficient strategy is possible, based on expression (5.18), which

we will describe in Section 5.6.2. This strategy will be referred to as 'receiver feedback cancellation' (RFC). However, from a theoretical point of view, the TFC and RFC strategies are equivalent. For the sake of an easy exposition we use the former in the theoretical analysis.

5.4.3 Removal of Indirect Feedback

As mentioned in Section 5.4.2, direct feedback can be easily removed. Unfortunately, indirect feedback is more difficult to avoid. However, if direct feedback is removed, the data diffuses through the network in one direction, i.e. data sent by node k over a specific edge of the network graph cannot return to node k through the same edge in opposite direction, and so it can only return through a different edge that is part of a cycle. Hence, if the network has a tree topology, i.e. the network graph contains no cycles, indirect feedback is automatically removed if direct feedback is removed. In the sequel, we assume that the network graph has been pruned to a spanning tree of the initial graph. Optimal spanning trees can be defined and computed in several ways. An overview of different spanning tree problems can be found in [18].

It is noted that the combination of TFC with a tree topology has some similarities with the message passing for belief propagation (BP) in trees (see e.g [19]). Furthermore, in Section 5.6.1, we will decompose the signal flow in an inwards fusion and an outwards diffusion flow, which is also similar in BP. Despite these strong similarities in the data flow, the estimation frameworks of both algorithms are very different and incomparable, even on a higher level of abstraction.

5.5 DANSE in a Network with a Tree Topology (T-DANSE)

In this section, we will extend the DANSE$_K$ algorithm, to operate in networks with a tree topology. We will refer to this as tree-DANSE$_K$ or T-DANSE$_K$. In the sequel, we will often refer to Fig. 5.5, showing an example network with a tree topology.

5.5.1 T-DANSE$_K$ Algorithm

A node k transmits observations of the K-channel TFC-signal $\mathbf{z}_{kq}$, defined by (5.17), to a node $q \in \mathcal{N}_k$. Fig. 5.6 illustrates this for a network graph with a line topology, which is a subgraph of the network graph in Fig. 5.5.

It is noted that a tree topology defines a unique path between any pair of nodes,

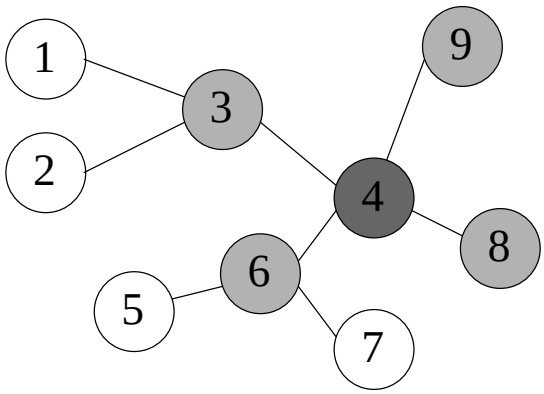


Figure 5.5: Example of a network graph with tree topology with 9 sensor nodes.

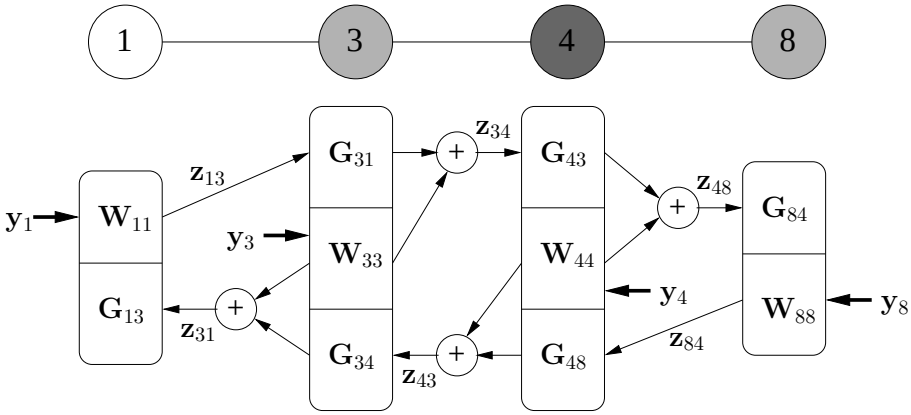


Figure 5.6: The T-DANSE$_K$ scheme in a network graph with a line topology.

if an edge can only be used once. Let $P_{p_1 \to p_t} = (p_1, p_2, \ldots, p_{t-1}, p_t)$ denote the ordered set of nodes defining the unique path from node p_1 to node p_t, and let $P_{p_1 \leftarrow p_t}$denote the inverse path, i.e. $P_{p_1 \leftarrow p_t} = P_{p_t \to p_1}$. Define

$$\mathbf{G}_{p_1 \leftarrow p_t} = \mathbf{G}_{p_{t-1} p_t} \mathbf{G}_{p_{t-2} p_{t-1}} \ldots \mathbf{G}_{p_2 p_3} \mathbf{G}_{p_1 p_2} \tag{5.19}$$

with p_j denoting the j-th node in the path $P_{p_1 \to p_t}$. We define $\mathbf{G}_{k \leftarrow k} = \mathbf{G}_{kk} = \mathbf{I}_k$. The order of the $\mathbf{G}$'s in (5.19) must be the same as the order of the nodes in the inverse path $P_{p_1 \leftarrow p_t}$.

> ***Example:*** The matrix $\mathbf{G}_{1 \leftarrow 8}$ for the network graph depicted in Fig. 5.5 is $\mathbf{G}_{1 \leftarrow 8} = \mathbf{G}_{48} \mathbf{G}_{34} \mathbf{G}_{13}$. This structure is clearly visible in the network graph of Fig. 5.6, defined by the path $P_{1 \leftarrow 8}$. Notice that $\mathbf{G}_{8 \leftarrow 1} = \mathbf{G}_{1 \to 8} = \mathbf{G}_{31} \mathbf{G}_{43} \mathbf{G}_{84}$.

The parametrization of the $\mathbf{W}_k$ effectively applied at node k, to generate $\overline{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y}$, is then

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_{11} \mathbf{G}_{k \leftarrow 1} \\ \vdots \\ \mathbf{W}_{JJ} \mathbf{G}_{k \leftarrow J} \end{bmatrix}. \tag{5.20}$$

Parametrization (5.20) defines a solution space for all $\mathbf{W}_k$, $k \in \mathcal{J}$, simultaneously, that depends on the network topology. Notice that its structure is very similar to (5.9), as used in fully connected DANSE.

Theorem 5.3 *If (5.1) holds, then the optimal estimators $\hat{\mathbf{W}}_k$ given in (5.5) are in the solution space defined by parametrization (5.20).*

Proof : The proof is based on expression (5.10), which follows from (5.1) and (5.5), and which is repeated here for convenience:

$$\forall\, k, q \in \mathcal{J} : \hat{\mathbf{W}}_k = \hat{\mathbf{W}}_q \mathbf{A}_{kq} \tag{5.21}$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H} \mathbf{A}_k^H$. By setting all $\mathbf{G}_{kq}$ matrices equal to $\mathbf{G}_{kq} = \mathbf{A}_{kq} = \mathbf{A}_q^{-H} \mathbf{A}_k^H$, we automatically have that $\mathbf{G}_{k \leftarrow l} = \mathbf{A}_{kl}$ for any k and l, since $\mathbf{A}_{nl} \mathbf{A}_{kn} = \mathbf{A}_{kl}$, for any k,l and n. By using a similar reasoning as in the proof of Theorem 5.1, we can show that all $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$, are in the solution space defined by parametrization (5.20). □

Let the matrix $\mathbf{G}_{k,-q}$ denote the stacked version of all $\mathbf{G}_{kn}$ matrices for which $n \in \mathcal{N}_k \backslash \{q\}$. Vector $\mathbf{z}_{\to k}$ denotes the vector in which all K-channel signals $\mathbf{z}_{qk}$ are stacked, for all $q \in \mathcal{N}_k$, i.e. it contains all signals that node k receives from its neighbors. Let P denote an ordered set of nodes that contains all nodes in

the network, possibly with repetition of nodes. Let p_j denote the j-th element in this set and let $|P|$ denote the number of elements in P. In general, we will use X^i to denote X at iteration i, where X can be a signal or a parameter.

The T-DANSE$_K$ algorithm then consists of the following steps:

The T-DANSE$_K$ Algorithm

1.
   - Initialize $\mathbf{W}_{qq}^0$ and $\mathbf{G}_{q,-q}^0$, $\forall\, q \in \mathcal{J}$, as random matrices.
   - $i \leftarrow 0$.
   - $k \leftarrow p_1$.
2.
   - Node k updates its local parameters $\mathbf{W}_{kk}^i$ and $\mathbf{G}_{k,-k}^i$ by minimizing its MSE criterion, based on observations of its own inputs sensor signal $\mathbf{y}_k$ and of the compressed signals $\mathbf{z}_{qk}^i$, that it receives from nodes $q \in \mathcal{N}_k$:

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k,-k}^{i+1} \end{bmatrix} = \underset{\mathbf{W}_{kk},\mathbf{G}_{k,-k}}{\arg\min}\; E\left\{ \| \mathbf{d}_k - \begin{bmatrix} \mathbf{W}_{kk}^H & \mathbf{G}_{k,-k}^H \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{\rightarrow k}^i \end{bmatrix} \|^2 \right\} . \tag{5.22}$$

   - The other nodes do not change their variables:

$$\forall\, q \in \mathcal{J}\backslash\{k\} :\; \mathbf{W}_{qq}^{i+1} = \mathbf{W}_{qq}^i,\; \mathbf{G}_{q,-q}^{i+1} = \mathbf{G}_{q,-q}^i \, . \tag{5.23}$$

3.
   - $i \leftarrow i+1$.
   - $k \leftarrow p_t$ with $t = (i \mod |P|) + 1$.
4. Return to step 2

The estimate of a sample $\mathbf{d}_k[t]$ of the desired signal $\mathbf{d}_k$ at node k at any point in the iterative process is computed as

$$\overline{\mathbf{d}}_k[t] = \mathbf{W}_{kk}^{i\,H} \mathbf{y}_k[t] + \sum_{q \in \mathcal{N}_k} \mathbf{G}_{kq}^{i\,H} \mathbf{z}_{qk}^i[t] \, . \tag{5.24}$$

Remark I: We emphasize again that the iterative nature of the algorithm does not mean that the same sensor signal observations are retransmitted after every iteration. In practical applications, iterations are spread over time, i.e. if $\mathbf{W}_{kk}^i$ is updated to $\mathbf{W}_{kk}^{i+1}$ at time t_0, this is only used to compress new observations $\mathbf{y}_k[t]$ and estimate the samples $\mathbf{d}_k[t]$ for which $t > t_0$, while previous observations for $t \leq t_0$ are neither recompressed, retransmitted nor re-estimated.

Effectively, each sensor signal observation is compressed and transmitted only once. This is motivated by the stationarity assumption of the involved signals.

Remark II: In the T-DANSE$_K$ algorithm, each node solves an estimation problem that is equivalent to the centralized problem (5.4), but on a much smaller scale, i.e. with less signals. This means that the computations are distributed over the nodes in the network.

Remark III: It is noted that, even when nodes only have access to observations of a single-channel sensor signal, i.e. $M_k = 1$, $\forall\, k \in \mathcal{J}$, the T-DANSE$_K$ algorithm yields a benefit in terms of communication bandwidth efficiency, compared to the relay case depicted in Fig. 5.3(a). In the fully connected case, DANSE$_K$ only yields an improvement in bandwidth efficiency if $M_k > K$ [11]. Furthermore, the T-DANSE$_K$ algorithm is fully scalable, i.e. the amount of data transmitted between each node pair does not depend on the number of nodes J, and the computational effort at each node only depends on the number of neighboring nodes (but not on J). This is important since the number of signals that a single node can receive and process in real-time is usually limited, especially when operating at large sampling rates. Based on the complexity analysis in [11], we find that the computations at node k (in a recursive implementation) have a complexity of

$$O\left((M_k + K|\mathcal{N}_k|)^2\right) \ . \tag{5.25}$$

This is to be compared with the complexity $O\left((M_k + K(J-1))^2\right)$ of fully connected DANSE, which depends on the total number of nodes J.

5.5.2 Convergence and Optimality

The following theorem provides a sufficient condition on the updating order P for T-DANSE$_K$ to guarantee convergence to the optimal estimators.

Theorem 5.4 *Consider a network with a tree topology. Let P denote an ordered set of nodes that defines a path through the network that starts in k and ends in any $q \in \mathcal{N}_k$, such that $\forall\, p \in \mathcal{J}: \ p \in P$. If (5.1) holds, then the T-DANSE$_K$ algorithm as described in Section 5.5.1 converges for any initialization of its parameters to the linear MMSE solution (5.5) for all k.*

Proof : The proof of this theorem is elaborate, and can be found in Appendix 5.A. Some additional concepts and lemmas are used in the proof, which can be found in Appendix 5.A and 5.B. □

Theorem 5.4 states that the updating order of the nodes must correspond to a path through the network. This means that if node k updates in iteration i,

then the node that updates in iteration $i+1$ must be in $\mathcal{N}_k$. For example, for the network in Fig. 5.5, a possible choice for P is $P = (1, 3, 2, 3, 4, 9, 4, 8, 4, 6, 5, 6, 7, 6, 4, 3)$.

Remark I: Extensive simulations show that the condition in Theorem 5.4 on the updating order P is sufficient, but not necessary. In fact, the algorithm is always observed to converge, regardless of the updating order of the nodes (see also Section 5.7). This is stated here as an observation since a proof is not yet available. However, choosing an updating order satisfying the condition in Theorem 5.4 usually results in a faster convergence for most of the nodes. This can be explained by the fact that this condition generally implies that nodes with many neighbors are updated more often. Since these nodes act as bottlenecks in the data diffusion, it is important that these are indeed updated frequently, whereas updates of nodes at the boundary of the network have less impact on the data flow in the rest of the network.

Remark II: The proof of convergence in Appendix 5.A shows that the algorithm is at least as fast as a centralized alternating optimization (AO) algorithm or block-coordinate descent method (a.k.a. nonlinear Gauss-Seidel iteration [20]), where alternating blocks of variables are optimized while the other variables are fixed. The T-DANSE$_K$ algorithm is usually even faster, since none of the variables are actually fixed, but rather constrained to a subspace. Since an AO method converges Q-linearly for strictly convex objective functions [21], we can conclude that T-DANSE$_K$ converges at least Q-linearly. The convergence proof in Appendix 5.A also shows that the MSE monotonically decreases at node k (when evaluated after each update of node k). When the network grows, convergence takes longer due to the sequential nature of the method, i.e. it takes more iterations to have a full round of updates. When nodes would update simultaneously, the convergence time usually scales much better with the network size, but convergence cannot be guaranteed anymore. Relaxation methods, similar to the ones applied in [14] may again yield convergence in this case, but this is beyond the scope of this paper.

Remark III: For the sake of an easy exposition, we have ignored clock- and transmission delays in the signal paths. However, if proper compensating delays are added at the right places in the signal path, the theoretical analysis above is not affected by this practical aspect. Section 5.6 also describes a data-driven computation of the sample estimates, which has the advantage that nodes do not need any information on the transmission delays.

Remark IV: In the special case where all the nodes estimate the same signal, i.e. $\mathbf{d}_k = \mathbf{d}$, $\forall\, k \in \mathcal{J}$, the convergence properties remain the same as in the scenario with node-specific estimation problems, i.e. the fact that different signals are estimated at each node does not affect the convergence speed of the algorithm. This straightforwardly follows from the proof of Theorem 5.4.

5.6 T-DANSE with Local Broadcasting

In this section, we describe how T-DANSE can operate in a WSN that allows local broadcasting to neighboring nodes, instead of using reserved communication links between node pairs. By describing the system as a data-driven feed-forward data flow, we will be able to transform the TFC procedure into another feedback cancellation procedure that exploits the higher communication bandwidth efficiency of a local broadcast communication protocol. However, these practical aspects have no effect on the iterations of the T-DANSE algorithm, and therefore the theoretical analysis of the previous section remains valid.

For reasons that will become clear, we will first group the different nodes of the network in subsets, which we will refer to as 'shells' of the network. Let h_k denote the maximum number of hops between node k and any other node in the network, and let $h_{\min} = \min_{k\in\mathcal{J}} h_k$ and $h_{\max} = \max_{k\in\mathcal{J}} h_k$. We define $N = h_{\max} - h_{\min}$. Let S_N denote the subset of nodes that form the outer shell of the network, i.e. $S_N = \{k : h_k = h_{\min} + N\}$. Similarly, we define the shells

$$S_n = \{k : h_k = h_{\min} + n\}\,,\ \ n = 0 \dots N\,. \tag{5.26}$$

The inner shell S_0 contains maximally 2 nodes, which we refer to as the root nodes. It is noted that the nodes in the outer shell S_N are leaf nodes, i.e. nodes with a single neighbor, but S_N does not necessarily contain all the leaf nodes of the network.

> ***Example:*** The shells of the network depicted in Fig. 5.5 are $S_2 = \{1, 2, 5, 7\}$ (white), $S_1 = \{3, 6, 8, 9\}$ (light grey) and $S_0 = \{4\}$ (dark grey).

In the sequel, we assume that each node knows the shell index to which it belongs, together with the shell indices of its neighboring nodes. This requires some upper layer protocol or coordination.

5.6.1 Data-Driven Computation of TFC-Signals

The T-DANSE algorithm uses the TFC-signals $\mathbf{z}_{kq}$ as defined in (5.17), which is an implicit definition. This signal definition is illustrated in Fig. 5.7 for node 3 of the graph depicted in Fig. 5.5. Because the network is assumed to have a tree topology, (5.17) can be easily solved for the $\mathbf{z}_{kq}$'s by backwards substition, where the leaf nodes act as starting points. Indeed, if k is a leaf node, then $\mathbf{z}_{kq} = \mathbf{W}_{kk}^H \mathbf{y}_k$, which does not contain contributions from any other node. This backwards substition defines causality constraints, and can be described by means of a data-driven signal flow graph, where elementary building blocks (as the one depicted in Fig. 5.7) are interconnected. Each internal operator is triggered when it has received a sample or a data packet on each of its input

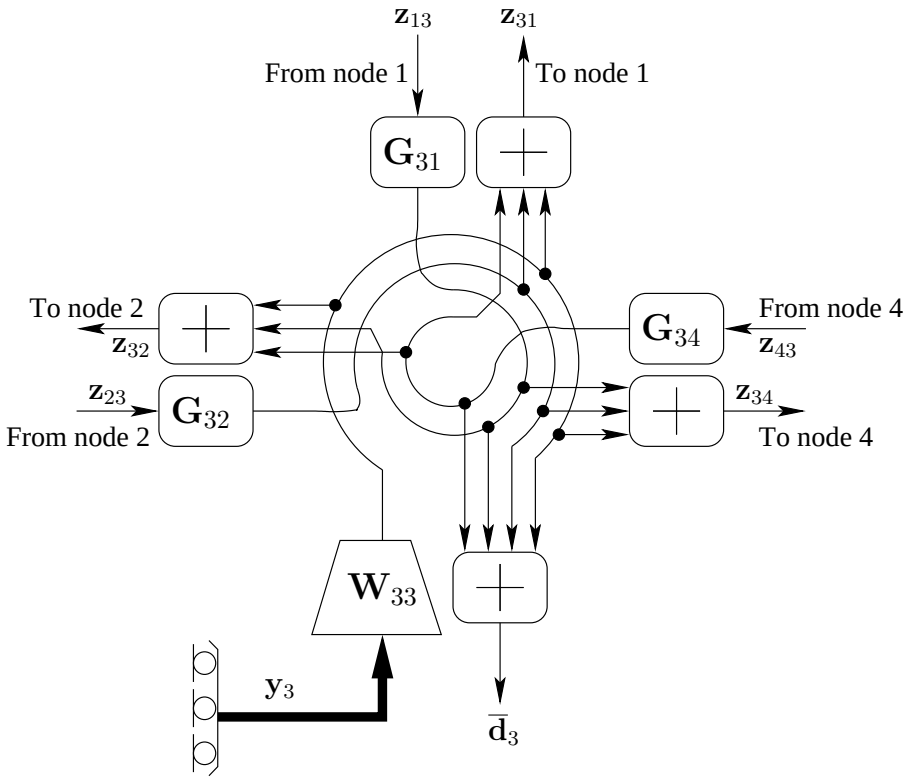


Figure 5.7: Illustration of the internal data flow in node 3 of the network depicted in Fig. 5.5.

lines, generating a new packet of data on its output line. This description can also be used in a practical implementation, and has the advantage that nodes do not need any information on the transmission delays (which is particularly interesting in communication networks with variable transmission delays).

Furthermore, the chain of computations in this data-driven procedure will show that the signal flow naturally decomposes in an inwards flow followed by an outwards flow. To this end, an important observation is that any non-root node k has only one neighbor q that is in a deeper shell, i.e. $\forall\, n \in \{1, \ldots, N\}, \forall\, k \in S_n : |\mathcal{N}_k \cap S_{n-1}| = 1$. The data flow is then decomposed as follows:

1. **Fusion flow** ($S_N \rightarrow S_0$): The fusion flow is initiated by the leaf nodes, who fire immediately after the collection of a new sensor observation. A non-leaf node fires when it has received data from all of its neighbors in the outer shell. At this event, the node fuses and forwards this data to the single neighbor in the deeper shell that has not fired yet. In this way,

the data travels inwards from S_N to S_0, where it is fused on the way with other local sensor data, and eventually arrives at the root node(s). If there are 2 root nodes, they have to exchange observations of the TFC signals, such that each of them has access to (fused) data that contains all sensor observations.

2. **Diffusion flow** ($S_0 \rightarrow S_N$): After arrival of the fusion flow in S_0, the root nodes initiate the diffusion flow by firing to all their neighbors in the outer shell, providing each with its node-specific TFC-signal. The neighbors at their turn do the same until the data is spread out over the entire network.

Remark I: In practice, the fusion flow and diffusion flow can run simultaneously, be it with a relative time lag. In other words, a leaf node k can fire each time it collects a new sensor observation $\mathbf{y}_k[t]$, even though the previous estimation sample $\overline{\mathbf{d}}_k[t-1]$ cannot be computed yet due to the fact that the required data is still on its way through the network.

Remark II: It is noted that in the fusion flow, each node transmits only one TFC-signal, whereas in the diffusion flow, a node k has to transmit $\mathcal{N}_k - 1$ TFC-signals (and a single root node k will transmit $\mathcal{N}_k$ TFC-signals). This will be exploited in the next subsection to reduce the communication bandwidth.

5.6.2 Receiver Feedback Cancellation

In Section 5.4.2, it is explained how direct feedback can be avoided by transmitter feedback cancellation. Although this is an efficient strategy in point-to-point communication protocols, TFC is very inefficient if the communication protocol allows local broadcasting. A better strategy would then be to let a node broadcast the same signal to all of its neighbors, and let the receiving nodes themselves remove their node-specific feedback component. We will refer to this strategy as 'receiver feedback cancellation' (RFC). The natural choice for the broadcast signal would be to use the $\mathbf{z}_k$ as defined in (5.12). A node q that receives the signal $\mathbf{z}_k$ from node k can then remove its own feedback component[10], using

$$\mathbf{z}_{kq} = \mathbf{z}_k - \mathbf{G}_{kq}^H \mathbf{z}_q \; . \tag{5.27}$$

However, (5.12) and (5.27) are implicit definitions, and it is not possible to compute the broadcast signals (5.12), $\forall\, k \in \mathcal{J}$, due to causality issues. Indeed, the computation of the sample $\mathbf{z}_k[t]$ based on (5.12) requires the samples $\mathbf{z}_q[t]$, $\forall\, q \in \mathcal{N}_k$, but $\mathbf{z}_q[t]$ cannot be computed by node q without the sample $\mathbf{z}_k[t]$, which results in a deadlock.

[10]Here it is assumed that $\mathbf{G}_{kq}$ is known at node q, which requires some minor additional information exchange between nodes, assuming that the sampling rate of the sensors is significantly larger than the update frequency of the estimation parameter at the different nodes.

To resolve this deadlock, we have to combine TFC with RFC, since the former is computable. To this end, we redefine the signals $\mathbf{z}_k$, $\forall k \in \mathcal{J}$, with an explicit definition based on the TFC-signals:

$$\begin{array}{ll} \forall\, k \in \mathcal{J}\backslash\mathcal{L}: & \mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k + \sum_{l\in\mathcal{N}_k} \mathbf{G}_{kl}^H \mathbf{z}_{lk} \\ \forall\, k \in \mathcal{L}: & \mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k \end{array} \tag{5.28}$$

where $\mathcal{L}$ denotes the set of leaf nodes. We denote the $\mathbf{z}_k$ signals (on the lefthand side) as RFC-signals. By comparing (5.28) with (5.17), we find that

$$\begin{array}{ll} \forall\, k \in \mathcal{J}\backslash\mathcal{L}, \forall\, q \in \mathcal{N}_k: & \mathbf{z}_{kq} = \mathbf{z}_k - \mathbf{G}_{kq}^H \mathbf{z}_{qk} \\ \forall\, k \in \mathcal{L}, \forall\, q \in \mathcal{N}_k: & \mathbf{z}_{kq} = \mathbf{z}_k\ . \end{array} \tag{5.29}$$

If a receiving node $q \in \mathcal{N}_k$ would have access to the signals $\mathbf{z}_k$ and $\mathbf{z}_{qk}$, and the parameters $\mathbf{G}_{kq}$, then node q itself can compute the TFC-signal $\mathbf{z}_{kq}$ by using expression (5.29). Therefore, some of the TFC-signals will have to be broadcast together with the RFC-signals. The natural question is then how to organize this. The answer straightforwardly follows from the decomposition of the signal flow in a fusion and a diffusion flow, as explained in the previous subsection.

In the fusion flow, each node provides only one neighbor with signal observations, i.e. its single neighbor in the deeper shell. In this case, RFC cannot provide any benefit, and therefore TFC-signals are transmitted. In the diffusion flow on the other hand, node k will provide $\mathcal{N}_k - 1$ nodes with signal observations. In this case, however, a root node k can compute observations of the RFC-signal $\mathbf{z}_k$ according to (5.28), since it has access to observations of all the signals on the righthand side (which are provided by the fusion flow). Node k then broadcasts observations of $\mathbf{z}_k$, and its neighbors in the outer shell can extract observations of their node-specific TFC-signal from the observations of $\mathbf{z}_k$, by using (5.29). A receiving node q, can then compute the observations of $\mathbf{z}_q$, similarly to (5.28), and broadcast this to its $\mathcal{N}_q - 1$ neighbors in the next shell. This is illustrated in Fig. 5.8 for a subgraph of the graph in Fig. 5.5.

When this RFC strategy is used, each node (except leaf nodes and single root nodes) transmits observations of 2 signals; a TFC-signal in the fusion flow and an RFC-signal in the diffusion flow. Hence, the amount of data that a node transmits is independent of the number of neighbors of that node.

5.7 Simulations

In this section, we provide batch mode simulation results for the T-DANSE$_3$ algorithm in networks with a tree topology. The first experiment is performed based on the network depicted in Fig.5.5. The network contains 9 nodes ($J =$

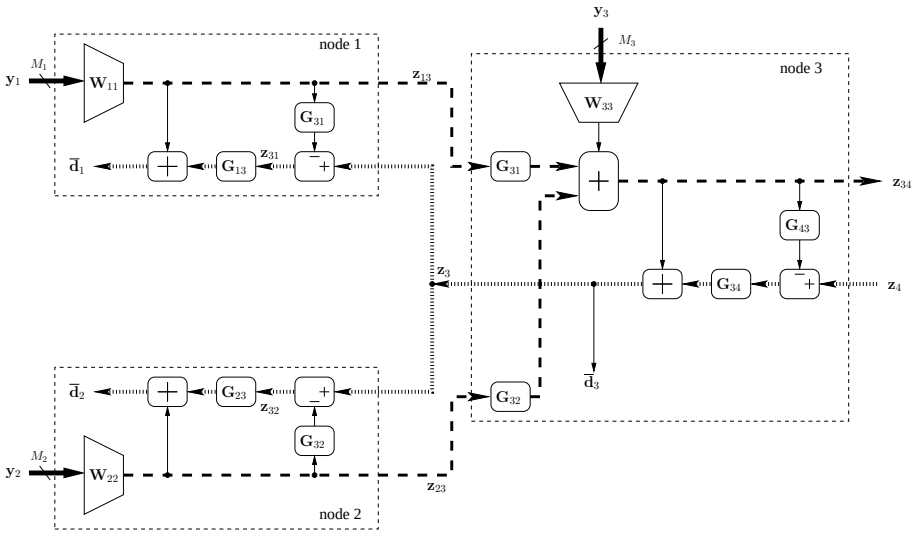
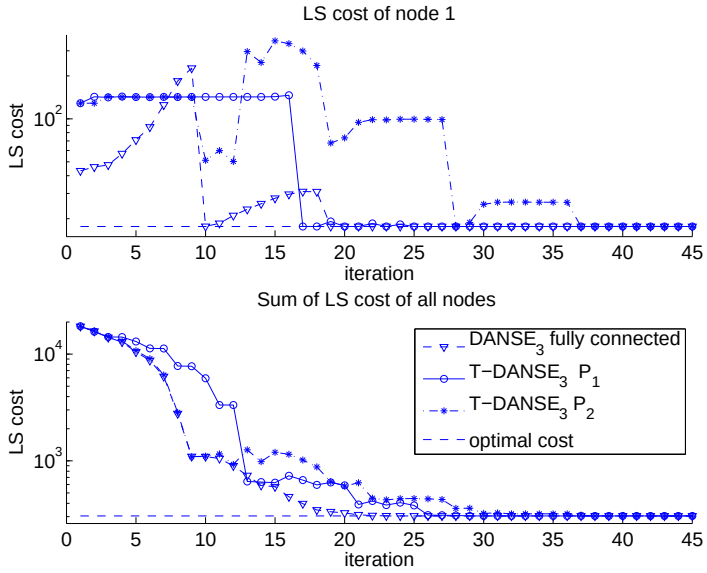


Figure 5.8: Illustration of the fusion flow (dashed line) and the diffusion flow (dotted line) between nodes 1, 2 and 3 for broadcast communication in the network depicted in Fig. 5.5.

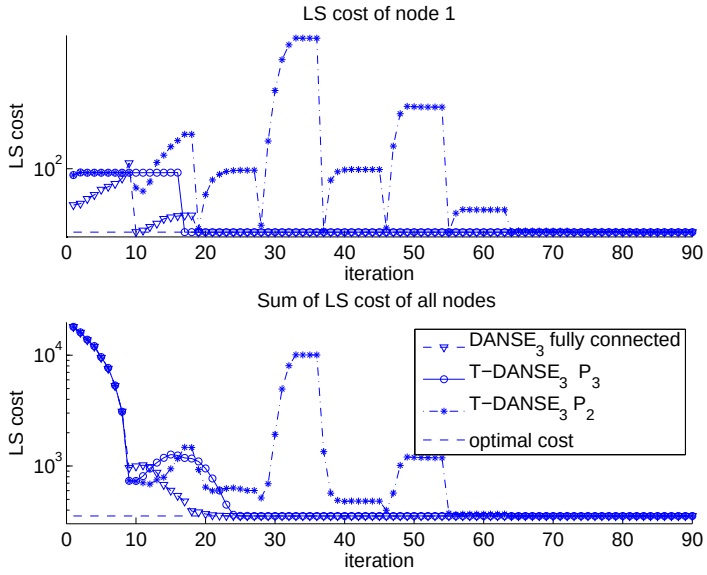
9), each having 10 sensors ($M = 90$). The dimension of the latent signal subspace defined by $\mathbf{d}$ is $K = 3$. All three signals in $\mathbf{d}$ are uniformly distributed random processes on the interval $[-0.5, 0.5]$ from which $N = 10000$ samples are generated. All sensor measurements correspond to a random linear combination of the three generated signals to which zero-mean white noise is added with half the power of the signals in $\mathbf{d}$. The $\mathbf{W}_{kk}$ variables are initialized randomly, whereas the $\mathbf{G}_{kq}$ variables are initialized as all-zero matrices. All MMSE cost functions are replaced by their related least-squares (LS) cost functions, i.e. (for node k)

$$J_k(\mathbf{W}_k) = \sum_{t=0}^{N} \|\mathbf{d}_k[t] - \mathbf{W}_k^H \mathbf{y}[t]\|^2 \ . \tag{5.30}$$

The results are given in Fig. 5.9(a), showing the LS cost of node 1 (above) and the summed LS cost of all the nodes (below) versus the iteration index i. Notice that one iteration corresponds to the time needed for a node to estimate the statistics of its inputs and to calculate the new parameter setting. Three different cases are simulated. In the first case, the network is assumed to be fully connected, and the DANSE$_3$ algorithm of [11] is applied where the updating order is round-robin. In the second and third case, the network has the tree topology shown in Fig. 5.5 and the T-DANSE$_3$ algorithm is applied. In case 2, the updating order is $P_1 = (1, 3, 2, 3, 4, 9, 4, 8, 4, 6, 5, 6, 7, 6, 4, 3)$, which satisfies the condition of Theorem 5.4, whereas in case 3 the updating order is



(a) Tree of Fig. 5.5



(b) Line topology

Figure 5.9: The LS cost of node 1 and summed LS cost versus the number of iterations (a) in the network depicted in Fig. 5.5 and (b) in a 9-node network with line topology.

$P_2 = (1, 2, \ldots, 9)$, i.e. round-robin, and so the condition of Theorem 5.4 is not satisfied.

Remarkably, the updating order P_1 yields a faster convergence than P_2 at node 1, despite the fact that the update rate of node 1 is higher in the latter. As mentioned already in Section 5.5.2, this holds for most of the nodes. If we only retain the iteration indices in $(1, 17, 33, \ldots,)$, i.e. the iteration steps in which node 1 updates its parameters, then the cost function J_1 decreases monotonically for updating order P_1, as indicated by (5.42) in the proof of Theorem 5.4. This does not hold in the round-robin case.

In the second experiment, T-DANSE$_3$ is applied in a 9-node network with a line topology, i.e. each node has exactly two neighbors, except for the 2 leaf nodes. The results are shown in Fig. 5.9(b). Here, we compare the updating order P_2 (round robin) with $P_3 = (1, 2, \ldots, 9, 8, 7, \ldots, 2)$, where the latter satisfies the conditions of Theorem 5.4. The difference in convergence speed between updating order P_2 and P_3 is even more significant in this case, where the latter converges much faster than the round-robin updating order P_2.

In both plots, it is observed that the LS cost function J_1 may increase significantly, even after it nearly reached the optimal level. This is due to the fact that the other nodes did not yet achieve optimal estimation performance, yielding significant changes in their estimation parameters. Since the estimators of all the nodes are intertwined due to (5.20), these changes have a significant effect on the local cost function J_1 at node 1, resulting in an increase. However, after a couple of iterations, an equilibrium state is reached where each node has optimal performance. The reason why the cost J_1 remains almost constant in the initial iterations, is due to the fact that node 1 chooses small entries for $\mathbf{G}_{13}$ in the first iteration, and therefore node 1 is basically cut off from the network until its next update.

5.8 Conclusions

In this paper, we have extended the DANSE$_K$ algorithm, introduced in [11, 12] for a fully connected sensor network, to the T-DANSE$_K$ algorithm which operates in a network with a tree topology. It is argued that feedback is to be avoided, when a straightforward modification of DANSE$_K$ is applied in a network that is not fully connected, since it harms the convergence and optimality properties of the algorithm. Direct feedback can be avoided easily, whereas indirect feedback is more difficult to remove in a network topology that has cycles. A tree topology is then a natural choice, since it has no cycles, for which the T-DANSE$_K$ algorithm can subsequently be derived. A condition is given on the updating order of the nodes to guarantee convergence to the optimal estimators. Simulations have shown that the condition on the updating

order of the nodes is sufficient but not necessary, although convergence is faster if the condition is satisfied.

Two different communication protocols have been considered, i.e. point-to-point transmission between node pairs and local broadcasts. For both protocols it is possible to remove direct feedback, and so from a theoretical point of view, there is no true difference. However, the local broadcast communication protocol is more efficient and scalable in terms of connectivity, i.e. the amount of data that is transmitted is independent of the number of neighbors at each node.

Appendix

5.A Proof of Theorem 5.4 (Convergence of T-DANSE$_K$)

Before proving Theorem 5.4, we first introduce some new concepts and lemmas. We will consider a partitioning $\mathcal{P}$, which is an ordered set of non-overlapping subsets of a set of nodes $\mathcal{J}$. The first subset of a partitioning $\mathcal{P}$ is referred to as the free subset, whereas the other subsets are referred to as the constrained subsets.

Example: For a 5-node network a possible partitioning is

$$\mathcal{P} = (\{2\}; \{1,4\}, \{3,5\}) \ .$$

The set $\{2\}$ is the free subset, and $\{1,4\}$ and $\{3,5\}$ are constrained subsets.

For a certain subset of nodes S, let $\mathbf{W}_{k|S}$ denote the stacked version of all the $\mathbf{W}_{kq}$'s (see Section 5.3) for which $q \in S$. Assume that the estimator $\mathbf{W}_k$ is initialized with a certain matrix $\mathbf{W}_k^0$. We consider an updating scheme that updates the values in $\mathbf{W}_k$ by a sequence of n alternating optimizations[11] (AO), defined by a sequence of partitionings $\left(\mathcal{P}^i\right)_{i=0\ldots n-1}$. In each step i of the AO sequence, MMSE optimization (5.6) is performed, but with constraints added to the variables in the constrained subsets of $\mathcal{P}^i$. In the i-th optimization step, the columns of $\mathbf{W}_{k|C}$ are constrained to the subspace defined by the columns

[11] The AO used in this paper is different from the AO algorithms (a.k.a. non-linear Gauss-Seidel algorithms) described in [20, 21], in which the optimization is done over a subset of the variables, while other variables are fixed to their current value. In this paper, the optimization in an AO step is performed over all variables, but constraints are added to certain variables in an alternating fashion.

of the current values of $\mathbf{W}^i_{k|C}$, where C is a constrained subset of $\mathcal{P}^i$. For $\mathcal{P}^i = \left(F^i; C^i_1, \ldots, C^i_{t_i}\right)$, the corresponding AO update step is then

$$\{\mathbf{W}^{i+1}_{k1}, ..., \mathbf{W}^{i+1}_{kJ}, \mathbf{C}_1, \ldots, \mathbf{C}_{t_i}\} =$$

$$\underset{\{\mathbf{W}_{k1},...,\mathbf{W}_{kJ},\mathbf{C}_1,...,\mathbf{C}_{t_i}\}}{\arg\min} E\{\|\mathbf{d}_k - \sum_{l=1}^{J} \mathbf{W}^H_{kl}\, \mathbf{y}_l\|^2\} \tag{5.31}$$

$$\text{s.t.} \begin{cases} \mathbf{W}_{k|C^i_1} = \mathbf{W}^i_{k|C^i_1}\mathbf{C}_1 \\ \vdots \\ \mathbf{W}_{k|C^i_{t_i}} = \mathbf{W}^i_{k|C^i_{t_i}}\mathbf{C}_{t_i} \end{cases}$$

where $\mathbf{C}_1$, ..., $\mathbf{C}_{t_i}$ are $K \times K$ matrices.

Example: The optimization in the AO-step defined by

$$\mathcal{P}^i = (\{2\}; \{1, 4\}, \{3, 5\})$$

is

$$\{\mathbf{W}^{i+1}_{k1}, ..., \mathbf{W}^{i+1}_{k5}, \mathbf{C}_1, \mathbf{C}_2\} = \underset{\{\mathbf{W}_{k1},...,\mathbf{W}_{k5}\}}{\arg\min} E\{\|\mathbf{d}_k - \sum_{l=1}^{5} \mathbf{W}^H_{kl}\, \mathbf{y}_l\|^2\}$$

$$\text{s.t.} \begin{bmatrix} \mathbf{W}_{k1} \\ \mathbf{W}_{k4} \end{bmatrix} = \begin{bmatrix} \mathbf{W}^i_{k1} \\ \mathbf{W}^i_{k4} \end{bmatrix} \mathbf{C}_1, \begin{bmatrix} \mathbf{W}_{k3} \\ \mathbf{W}_{k5} \end{bmatrix} = \begin{bmatrix} \mathbf{W}^i_{k3} \\ \mathbf{W}^i_{k5} \end{bmatrix} \mathbf{C}_2 \,.$$

At first, we will consider the hypothetical case where all sensor signal observations are available to all nodes, and a node k can compute any AO-step on its full unparametrized $\mathbf{W}_k$. Later, we will consider the AO procedure that is related to the actual network, by linking the sequence of partitionings $\left(\mathcal{P}^i\right)_{i=0\ldots n-1}$ to the network topology. To analyse this AO process, we will first analyse the convergence properties in a 2-node network where the AO steps of both nodes are linked, based on the sensors to which each node has access. Based on this results, we will prove the convergence of the T-DANSE$_K$ algorithm by making a hierarchical decomposition of the entire network into simpler 2-node networks.

In the sequel, we will refer to the MSE cost function of node k, as given in (5.31), with $J_k(\mathbf{W}_k) = J_k\left(\left[\mathbf{W}^T_{k1} \ldots \mathbf{W}^T_{kJ}\right]^T\right)$. Notice that for any AO-step performed on $J_k(\mathbf{W}_k)$:

$$J_k(\mathbf{W}^{i+1}_k) \leq J_k(\mathbf{W}^i_k)\,, \tag{5.32}$$

and because the optimization problem (5.31) is strictly convex[12]:

[12]Strict convexity is satisfied if the sensor measurements $\mathbf{y}_k$ are not perfectly correlated, which is always satisfied in practice due to sensor noise.

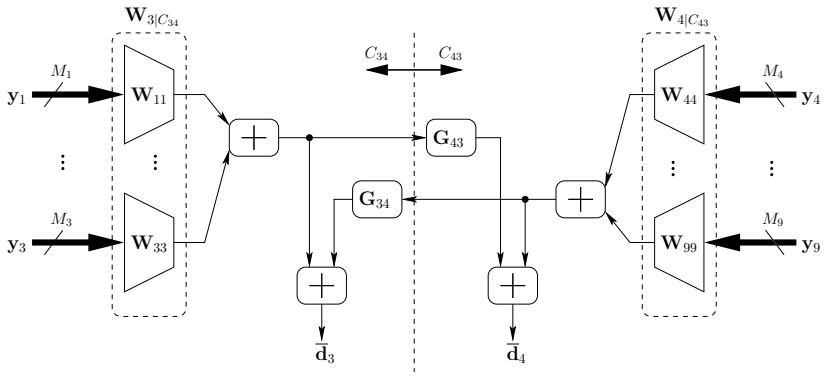


Figure 5.10: The hypothetical 2-node network for the edge $(3, 4)$ in the network depicted in Fig. 5.5.

$$\mathbf{W}_k^{i+1} \neq \mathbf{W}_k^i \Leftrightarrow J_k(\mathbf{W}_k^{i+1}) < J_k(\mathbf{W}_k^i) \; . \tag{5.33}$$

Lemma 5.5 *Consider an arbitrary AO sequence defined by a sequence of partitionings* $\left(\mathcal{P}^i\right)_{i=0\ldots n-1}$ *with* $\mathcal{P}^i = \{F^i; C_1^i, \ldots, C_{t_i}^i\}$. *This AO sequence is simultaneously applied to the cost function* $J_k(\mathbf{W}_k)$ *of node* k *to update* $\mathbf{W}_k^i$ *and to the cost function* $J_q(\mathbf{W}_q)$ *of node* q *to update* $\mathbf{W}_q^i$. *Assume that the two initial centralized estimators* $\mathbf{W}_k^0$ *and* $\mathbf{W}_q^0$ *are related through* $\mathbf{W}_{q|C_j^0}^0 = \mathbf{W}_{k|C_j^0}^0 \mathbf{G}_j$, $\forall j \in \{1, \ldots, t_0\}$, *where* $\mathbf{G}_j$ *is a non-singular* $K \times K$ *matrix. If (5.1) is satisfied, then the following holds for any* $i \in \{1, \ldots, n\}$*:*

$$\mathbf{W}_k^i = \mathbf{W}_q^i \mathbf{A}_{kq} \; , \tag{5.34}$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H} \mathbf{A}_k^H$.

Proof : See Appendix 5.B. □

Lemma 5.5 shows that all the (centralized) AO-steps at the different nodes (minimizing different cost functions) result in $\mathbf{W}_k$'s that have the same K-dimensional column subspace, if they are initialized properly.

In a network with a tree topology, there always exists a unique graph cut that cuts the edge (k, q) between two neighboring nodes k and q, and no other edges. This cut partitions the graph in two complementary sets of nodes: C_{kq} denotes the one that contains node k, and C_{qk} denotes the one that contains node q.

Consider a hypothetical 2-node network with nodes k and q, where node k has access to the sensor signal observations $\mathbf{y}_{C_{kq}}$ and node q has access to the sensor

signal observations $\mathbf{y}_{C_{qk}}$, where $\mathbf{y}_{C_{kq}}$ and $\mathbf{y}_{C_{qk}}$ denote the stacked version of all $\mathbf{y}_j$ for which $j \in C_{kq}$ and $j \in C_{qk}$, respectively. We now parametrize a subset of $\mathbf{W}_k$ and $\mathbf{W}_q$ as follows:

$$\mathbf{W}_{k|C_{qk}} = \mathbf{W}_{q|C_{qk}}\mathbf{G}_{kq} \tag{5.35}$$

$$\mathbf{W}_{q|C_{kq}} = \mathbf{W}_{k|C_{kq}}\mathbf{G}_{qk} \ . \tag{5.36}$$

It is assumed that node k has access to the variables $\mathbf{W}_{k|C_{kq}}$ and $\mathbf{G}_{kq}$, and that node q has access to the variables $\mathbf{W}_{q|C_{qk}}$ and $\mathbf{G}_{qk}$, as it is the case in the DANSE$_K$ algorithm with parametrization (5.9).

> ***Example:*** Consider the network in Fig. 5.5. Let $k = 3$ and $q = 4$, then $C_{34} = \{1, 2, 3\}$ and $C_{43} = \{4, 5, 6, 7, 8, 9\}$, and the 2-node network corresponding to (5.35)-(5.36) is shown in Fig. 5.10.

The parametrization of the hypothetical 2-node network as described above corresponds to partitionings $\mathcal{P}$ that satisfy a specific form. Indeed, node k can never freely manipulate the variables in $\mathbf{W}_{k|C_{qk}}$, and therefore any AO-step performed by node k must have partitionings of the form

$$\mathcal{P}_{k\leftarrow q} = \{(F; C_1, \ldots, C_t) : \exists\, n \in \{1, \ldots, t\} : C_{qk} \subseteq C_n\} \ . \tag{5.37}$$

Expression (5.37) implies that, during any optimization step at node k, the search space of the variables in $\mathbf{W}_{k|C_{qk}}$ is constrained to the current column space of $\mathbf{W}^i_{k|C_{qk}}$, which is due to (5.35) and the fact that node k can only change $\mathbf{W}_{k|C_{qk}}$ through $\mathbf{G}_{kq}$. Similarly, an AO-step performed by node q must have partitionings of the form $\mathcal{P}_{q\leftarrow k}$, which is also given by (5.37) (by switching k and q). It is noted that, even though it is assumed that node k can directly manipulate all entries in $\mathbf{W}_{k|C_{kq}}$, we do not assume that $\mathcal{P}_{k\leftarrow q} = (C_{kq}; C_{qk})$. The set C_{kq} can be divided in a free subset together with one or more constrained subsets. The latter can even be merged with C_{qk}.

> ***Example:*** Consider the network in Fig. 5.5. Let $k = 4$ and $q = 3$, then a possible partitioning $\mathcal{P}_{4\leftarrow 3}$ is $\mathcal{P}_{4\leftarrow 3} = (\{5, 6, 7\};\ \{1, 2, 3, 4\}, \{8, 9\})$. Notice that $C_{34} = \{1, 2, 3\}$ is a subset of one of the constrained subsets of $\mathcal{P}_{4\leftarrow 3}$, as defined by (5.37).

We will consider AO sequences that are computed by the two nodes in this hypothetical network as follows. If the partitioning is of the form $\mathcal{P}_{k\leftarrow q}$, then the AO step (5.31) is performed by node k where the optimization is with respect to cost function $J_k(\mathbf{W}_k)$. This node k has direct access to the vector $\mathbf{W}_{k|C_{kq}}$, although it may also be constrained by other constrained sets in $\mathcal{P}_{k\leftarrow q}$. The variables in $\mathbf{W}_{k|C_{qk}}$ are parametrized as in (5.35) and manipulated through

the $\mathbf{G}_{kq}$ matrix. Similarly, if the partitioning is of the form $\mathcal{P}_{q\leftarrow k}$ node q will optimize its parameters $\mathbf{W}_{q|C_{qk}}$ and $\mathbf{G}_{qk}$ with respect to cost function $J_q(\mathbf{W}_q)$. It is noted that that, due to (5.35)-(5.36), an update of $\mathbf{W}_{k|C_{kq}}$ also changes $\mathbf{W}_{q|C_{kq}}$ and an update of $\mathbf{W}_{q|C_{qk}}$ changes $\mathbf{W}_{k|C_{qk}}$.

Lemma 5.6 *Consider a network with tree topology, and consider the hypothetical 2-node network defined by the edge (k,q) as explained above, with the corresponding linked parametrization of $\mathbf{W}_k$ and $\mathbf{W}_q$ as defined by (5.35)-(5.36). Consider an AO sequence of n steps defined by a sequence of partitionings*

$$\left(\mathcal{P}^0_{k\leftarrow q}, \left(\mathcal{P}^i_{q\leftarrow k}\right)_{i=1\ldots(n-2)}, \mathcal{P}^{n-1}_{k\leftarrow q}\right)$$

where the first and last AO step are performed by node k and the others by node q. Assume that in the first and last AO step, the set C_{qk} is a constrained subset as such, without additional nodes. If (5.1) is satisfied, then the resulting $\mathbf{W}^n_k$ will be the same as if node k had access to all sensor signal observations and performed all optimizations in the AO sequence by itself with respect to its own cost function J_k.

Proof : This lemma is a straightforward consequence of Lemma 5.5, which shows that any $\mathbf{W}^i_q$ that results from an AO sequence with respect to $J_q(\mathbf{W}_q)$, is the same as $\mathbf{W}^i_k$ that results from the same AO sequence with respect to $J_k(\mathbf{W}_k)$, except for a transformation by the matrix $\mathbf{A}_{kq}$. The latter can be compensated by the $\mathbf{G}$'s in parametrization (5.35)-(5.36). Since node k performs the last AO step, and since C_{qk} is a constrained subset, $\mathbf{G}^n_{kq}$ indeed compensates for this transformation with respect to $\mathbf{W}^{n-1}_{q|C_{qk}}$. Since C_{qk} is a constrained subset in the initial AO step, the assumption in Lemma 5.5 on the initialization of the parameters is automatically satisfied by the parametrization (5.35)-(5.36). □

We now define a one-to-one correspondence between a node k and a partitioning $\mathcal{P}$ as follows:

$$k \leftrightarrow \mathcal{P} = (\{k\}; C_{\mathcal{N}_k}) \tag{5.38}$$

with $C_{\mathcal{N}_k}$ denoting the set containing all the sets C_{jk} for which $j \in \mathcal{N}_k$.

> ***Example:*** in the case of the network depicted in Fig. 5.5, we have that $4 \leftrightarrow (\{4\}; \{1,2,3\}, \{5,6,7\}, \{8\}, \{9\})$.

The correspondence (5.38) defines another correspondence between a path P over $n-1$ edges through the network, and an n-step AO procedure defined by the sequence of partitionings

$$P \leftrightarrow \left(\mathcal{P}^i\right)_{i=0\ldots n-1} . \tag{5.39}$$

In the sequel, we assume that $\mathbf{W}_k$, $\forall\, k \in \mathcal{J}$, is parametrized according to (5.20), and that the AO step with partitioning of the form (5.38) is applied to the cost function $J_k(\mathbf{W}_k)$. Notice that an update of node k by the T-DANSE$_K$ algorithm is equivalent to such an AO step. Indeed, the update of $\mathbf{W}_{kk}$ and $\mathbf{G}_{k-k}$ is defined by a constrained optimization over the variables in $\mathbf{W}_k$, where $\mathbf{W}_{kk}$ are unconstrained variables (free subset) and where the matrix $\mathbf{G}_{k-k}$ is used to manipulate the constrained variables in the constrained subsets of (5.38). The T-DANSE$_K$ algorithm thus performs the AO sequence based on the sequence of partitioning defined by (5.39), in which the actual cost function and optimization variables change along with the corresponding node that is actually updating. It is noted that, even though an AO step at node k is defined on the variable $\mathbf{W}_k$, the resulting update of $\mathbf{W}_{kk}$ and $\mathbf{G}_{k-k}$ has an indirect influence on the other variables $\mathbf{W}_q$ with $q \neq k$, since they are linked through parametrization (5.20).

Lemma 5.7 *Consider a network with a tree topology and a path $P_{k\to k}$ through this network with length $n-1$, that never passes through node k, except at the start and at the end. Consider the T-DANSE$_K$ updating sequence equivalent to the n-step AO sequence defined by the partitionings $\left(\mathcal{P}^i\right)_{i=0\ldots n-1} \leftrightarrow P_{k\to k}$. If (5.1) is satisfied, then the resulting $\mathbf{W}_k^n$, parameterized by (5.20), will be the same as if node k had access to all sensor signal observations and performed all optimizations in the AO sequence by itself with respect to its own cost function J_k.*

Proof : This lemma can be proven by recursively applying Lemma 5.6 on the different edges that are visited in the path $P_{k\to k}$. Let q be the second node that is visited in the path $P_{k\to k}$, then $P_{k\to k} = (k, P_{q\to q}, k)$. Notice that the new path $P_{q\to q}$ again starts and ends with the same edge. This is a consequence of the fact that the network has a tree topology. Since the path $P_{k\to k}$ eventually returns to node k through the edge (k,q), Lemma 5.6 can be applied at this edge, in which $k \leftrightarrow \mathcal{P}^0_{k\leftarrow q} = \mathcal{P}^{n-1}_{k\leftarrow q}$ and $P_{q\to q} \leftrightarrow \left(\mathcal{P}^i_{q\leftarrow k}\right)_{i=1\ldots(n-2)}$. Indeed, $\mathcal{P}^0_{k\leftarrow q}$ and $\mathcal{P}^{n-1}_{k\leftarrow q}$ both contain C_{qk} as a constrained subset, and the partitionings in $\left(\mathcal{P}^i_{q\leftarrow k}\right)_{i=1\ldots(n-2)}$ all contain C_{kq} in one of their constrained subset. This can then be viewed as a hypothetical 2-node network in which node q performs the AO sequence defined by $\left(\mathcal{P}^i_{q\leftarrow k}\right)_{i=2\ldots(n-1)}$, assuming that node q has access to all sensor signal observations in $\{\mathbf{y}_n : n \in C_{qk}\}$. Lemma 5.6 then states that the resulting $\mathbf{W}_k^n$ is the same as if node k performed the whole AO sequence defined by $\left(\mathcal{P}^i\right)_{i=0\ldots n-1}$ by itself.

Obviously, node q does not have direct access to all sensor signal observations in $\{\mathbf{y}_n : n \in C_{qk}\}$ and cannot perform the AO sequence $\left(\mathcal{P}^i_{q\leftarrow k}\right)_{i=1\ldots(n-2)}$. However, since the new path $P_{q\to q}$ also starts and ends with the same edge, we

can again apply Lemma 5.6 on the edge between node q and the node that is visited third in the path $P_{k\to k}$. This line of thought can be continued further until the problem has been recursively decomposed into a hierarchy of 2-node networks. Applying Lemma 5.6 backwards on all these recursive problems proves the theorem. □

Lemma 5.8 *Under the same assumptions of Lemma 5.7, the following holds:*

$$J_k(\mathbf{W}_k^n) = J_k(\mathbf{W}_k^1) \Rightarrow \left(\mathbf{W}_{k|C_{kq}}^n = \mathbf{W}_{k|C_{kq}}^1,\, \mathbf{G}_{kq}^n = \mathbf{A}_{kq},\, \mathbf{G}_{qk}^n = \mathbf{A}_{qk}\right) \quad (5.40)$$

where q is the neighboring node of k that is visited first and last but one in the path $P_{k\to k}$.

Proof : Assume hypothetically that node k performs all the updates of the AO sequence defined by the path $P_{k\to k}$, on its own cost function $J_k(\mathbf{W}_k)$. Since $J_k(\mathbf{W}_k^n) = J_k(\mathbf{W}_k^1)$, and because of (5.32), we find that $J_k(\mathbf{W}_k^i) = J_k(\mathbf{W}_k^1)$, $\forall\, i \in \{1,\ldots,n\}$. The latter, combined with (5.33), yields

$$\mathbf{W}_k^i = \mathbf{W}_k^1, \quad \forall\, i \in \{1,\ldots,n\}\,. \quad (5.41)$$

Keep in mind that (5.41) only holds if node k itself would perform all the AO steps, which is not the case. However, since $P_{k\to k}$ starts and ends in node k, the first and last AO step are performed by node k itself, and Lemma 5.7 then explains that $\mathbf{W}_k^1$ and $\mathbf{W}_k^n$ are equal to the result we would obtain if node k performed all updates by itself. Therefore, (5.41) does indeed hold for $i = n$, which proves the first part of the righthand side of (5.40).

If node q would perform all the updates of the AO sequence defined by the path $P_{k\to k}$, on its own cost function $J_q(\mathbf{W}_q)$, then (5.41) and Lemma 5.5 would imply that $\mathbf{W}_q^i = \mathbf{W}_k^1\mathbf{A}_{qk}$, $\forall\, i \in \{1,\ldots,n\}$. Since node q is the second to last node that is updated, the latter does indeed hold for $i = n-1$ (implied by Lemma 5.7). Therefore, $\mathbf{G}_{qk}^{n-1} = \mathbf{A}_{qk}$, and since $\mathbf{G}_{qk}$ is not updated in the last step of the AO sequence, $\mathbf{G}_{qk}^n = \mathbf{G}_{qk}^{n-1} = \mathbf{A}_{qk}$. The fact that $\mathbf{G}_{kq}^n = \mathbf{A}_{kq}$ can be proven with a similar argument. □

We can now prove the main theorem:

Proof : [Proof of Theorem 5.4] Consider the node $k = p_1$, i.e. the first node that is updated by the T-DANSE$_K$ algorithm. Consider the infinite path $\overline{P} = (P, P, \ldots)$, i.e. the periodic extension of path P, that defines the updating order of the T-DANSE$_K$ algorithm. Path $\overline{P}$ can be split into a sequence of subpaths $\left(P_{k\to k}^j\right)_{j\in\mathbb{N}}$ that all satisfy the conditions in Lemma 5.7. This lemma shows that, every time node k is updated, the result is as if node k performed

all optimizations in the AO sequence with respect to its own cost function J_k, even though the AO updates are performed by different nodes with respect to different cost functions. With (5.32), we therefore find that:

$$J_k(\mathbf{W}_k^{s_k^{j+1}}) \leq J_k(\mathbf{W}_k^{s_k^j}) \tag{5.42}$$

where the subsequence $\left(s_k^j\right)_{j\in\mathbb{N}} \subset (i)_{i\in\mathbb{N}}$ corresponds to the iteration indices at which node k updates the entries in $\mathbf{W}_{kk}^i$ and $\mathbf{G}_{k-k}^i$. By using a similar reasoning, it can be shown that expression (5.42) holds for any node k. This shows that all sequences $\left(J_k(\mathbf{W}_k^{s_k^j})\right)_{j\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$, are decreasing sequences, and since they have a lower bound, they converge. It can be shown that convergence of the sequence $\left(J_k(\mathbf{W}_k^{s_k^j})\right)_{j\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$, implies convergence of the sequences $\left(\mathbf{W}_{kk}^i\right)_{i\in\mathbb{N}}$ and $\left(\mathbf{G}_{k,-k}^i\right)_{i\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$, by applying Lemma 5.8 to each subpath of the form $\left(P_{k\to k}^j\right)_{j\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$. This proves convergence of the T-DANSE$_K$ algorithm.

Notice that, from Lemma 5.8, it follows that

$$\forall\, k \in \mathcal{J},\, \forall\, q \in \mathcal{N}_k :\; \mathbf{G}_{kq}^\infty = \mathbf{A}_{kq} \tag{5.43}$$

and therefore, $\mathbf{G}_{k\leftarrow l}^\infty = \mathbf{A}_{kl}$ for any k and l, since $\mathbf{A}_{nl}\mathbf{A}_{kn} = \mathbf{A}_{kl}$, for any k,l and n. Parametrization (5.20) then shows that

$$\forall\, k, q \in \mathcal{J} :\; \mathbf{W}_k^\infty = \mathbf{W}_q^\infty \mathbf{A}_{kq} \tag{5.44}$$

which satisfies the mutual property of the MMSE solutions given by (5.10). With this, we can show that $\mathbf{W}_k^\infty = \hat{\mathbf{W}}_k$, $\forall k \in \mathcal{J}$, by using the same arguments as in the optimality proof for the DANSE$_K$ algorithm in a fully connected network, as given in [11]. It is therefore omitted here. □

5.B Proof of Lemma 5.5

Proof : We will prove the following induction hypothesis:

$$\begin{gathered}\forall\, j \in \{1,\ldots,t_i\},\, \exists\, \mathbf{G}_j \in \left(\mathbb{C}^{K\times K}\right)^{-1} :\; \mathbf{W}_{k|C_j^i}^i = \mathbf{W}_{q|C_j^i}^i \mathbf{G}_j \\ \Downarrow \\ \mathbf{W}_k^{i+1} = \mathbf{W}_q^{i+1}\mathbf{A}_{kq}\end{gathered} \tag{5.45}$$

with $\left(\mathbb{C}^{K\times K}\right)^{-1}$ denoting the set of non-singular complex valued $K \times K$ matrices. The lefthand side of (5.45) states that any $\mathbf{W}_{k|C_j^i}^i$ has the same column

space as $\mathbf{W}^i_{q|C^i_j}$. This implies that the search space for $\mathbf{W}_k$ and $\mathbf{W}_q$ is the same in the i-th optimization step.

For notational convenience, we omit the superscript i in C^i_j and the subscript i in t_i. By substituting the constraints of (5.31) in the cost function, we obtain the unconstrained optimization problem:

$$\min_{\mathbf{V}_k} E\{\|\mathbf{d}_k - \mathbf{V}_k^H \tilde{\mathbf{y}}_k^i\|^2\} \tag{5.46}$$

with

$$\mathbf{V}_k^H = \left[\mathbf{W}^H_{k|F} \middle| \mathbf{C}^H_{1,k} \middle| \dots \middle| \mathbf{C}^H_{t,k}\right] , \quad \tilde{\mathbf{y}}_k^i = \left[\begin{array}{c} \mathbf{y}_F \\ \hline \mathbf{W}^{i\,H}_{k|C_1}\mathbf{y}_{C_1} \\ \hline \vdots \\ \hline \mathbf{W}^{i\,H}_{k|C_t}\mathbf{y}_{C_t} \end{array}\right] \tag{5.47}$$

where $\mathbf{y}_F$ and $\mathbf{y}_{C_j}$ denote the stacked vector of all $\mathbf{y}_l$ for which $l \in F$ and $l \in C_j$, respectively. The solution of (5.46) is:

$$\mathbf{V}_k = \left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}\right)^{-1} \mathbf{R}^i_{\tilde{y}_k d_k} \tag{5.48}$$

with $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k} = E\{\tilde{\mathbf{y}}_k^i \tilde{\mathbf{y}}_k^{i\,H}\}$ and $\mathbf{R}^i_{\tilde{y}_k d_k} = E\{\tilde{\mathbf{y}}_k^i \mathbf{d}_k^H\}$. Consider the optimization problem (5.46) for another node q, i.e. $\min_{\mathbf{V}_q} E\{\|\mathbf{d}_q - \mathbf{V}_q^H \tilde{\mathbf{y}}_q^i\|^2\}$. The solution of this optimization problem is

$$\mathbf{V}_q = \left(\mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}\right)^{-1} \mathbf{R}^i_{\tilde{y}_q d_q} . \tag{5.49}$$

If the lefthand side of (5.45) is satisfied, then we can write

$$\tilde{\mathbf{y}}_k^i = \mathbf{D}^H \tilde{\mathbf{y}}_q^i \tag{5.50}$$

with $\mathbf{D}$ a block diagonal matrix defined by $\mathbf{D}$ =blockdiag$\{\mathbf{I}, \mathbf{G}_1, \dots, \mathbf{G}_t\}$, where $\mathbf{I}$ denotes an identity matrix of appropriate dimensions. From (5.1) and (5.50) we find the following relations between correlation matrices:

$$\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k} = \mathbf{D}^H \mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}\mathbf{D} , \quad \mathbf{R}^i_{\tilde{y}_k d_k} = \mathbf{D}^H \mathbf{R}^i_{\tilde{y}_q d_q}\mathbf{A}_{kq} \tag{5.51}$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H}\mathbf{A}_k^H$. Comparison of (5.48) and (5.49), together with (5.51), yields

$$\mathbf{V}_k = \mathbf{D}^{-1}\mathbf{V}_q\mathbf{A}_{kq} . \tag{5.52}$$

By comparing the lefthand side and the righthand side of (5.52), we find that

$$\mathbf{W}^{i+1}_{k|F} = \mathbf{W}^{i+1}_{q|F}\mathbf{A}_{kq} \tag{5.53}$$

and

$$\forall\, j \in \{1, \dots, t\} : \mathbf{W}^{i+1}_{k|C_j} = \mathbf{W}^i_{k|C_j}\mathbf{C}_{j,k}$$

$$
\begin{aligned}
&= \mathbf{W}^{i}_{k|C_j} \mathbf{G}_j^{-1} \mathbf{C}_{j,q} \mathbf{A}_{kq} \\
&= \mathbf{W}^{i}_{q|C_j} \mathbf{C}_{j,q} \mathbf{A}_{kq} \\
&= \mathbf{W}^{i+1}_{q|C_j} \mathbf{A}_{kq} \qquad (5.54)
\end{aligned}
$$

which proves the induction hypothesis (5.45). Notice that

$$
\mathbf{W}^{i+1}_{k} = \mathbf{W}^{i+1}_{q} \mathbf{A}_{kq} \Rightarrow \forall\, j \in \{1, \ldots, t_{i+1}\} : \mathbf{W}^{i+1}_{k|C^{i+1}_j} = \mathbf{W}^{i+1}_{q|C^{i+1}_j} \mathbf{A}_{kq} \;, \quad (5.55)
$$

i.e. if the righthand side of (5.45) holds for $i = s$, then the lefthand side of (5.45) holds for $i = s + 1$. With (5.45) and (5.55), and since the lefthand side of (5.45) is satisfied for $i = 0$, the lemma is proven by induction. □

Bibliography

[1] 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, July 2008.

[2] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[3] S. Kar and J. Moura, "Distributed linear parameter estimation in sensor networks: Convergence properties," in *Proc. 42nd Asilomar Conference on Signals, Systems and Computers*, oct. 2008, pp. 1347 –1351.

[4] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, pp. 2320–2330, May 2011.

[5] B. Van Veen and K. Buckley, "Beamforming: a versatile approach to spatial filtering," *ASSP Magazine, IEEE*, vol. 5, no. 2, pp. 4 –24, apr. 1988.

[6] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, pp. 38–51, Jan. 2009.

[7] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435.

[8] A. Bertrand, J. Callebaut, and M. Moonen, "Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks," in *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC), Tel Aviv, Israel*, August 2010.

[9] A. Dogandzic and B. Zhang, "Distributed estimation and detection for sensor networks using hidden markov random field models," *IEEE Transactions on Signal Processing*, vol. 54, no. 8, pp. 3200 –3215, 2006.

[10] J. Fang and H. Li, "Distributed estimation of gauss - markov random fields with one-bit quantized data," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 449 –452, May 2010.

[11] 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, pp. 5277–5291, 2010.

[12] ——, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan*, April 2009.

[13] S. Doclo, T. 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," *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC), Paris, France*, Sep. 2006.

[14] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part II: simultaneous & asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5292–5306, 2010.

[15] ——, "Distributed adaptive node-specific MMSE signal estimation in sensor networks with a tree topology," *Proc. of the European signal processing conference (EUSIPCO), Glasgow - Scotland*, August 2009.

[16] A. Spriet, M. Moonen, and J. Wouters, "The impact of speech detection errors on the noise reduction performance of multi-channel wiener filtering and generalized sidelobe cancellation," *Signal Processing*, vol. 85, pp. 1073–1088, June 2005.

[17] A. Bertrand and M. Moonen, "Efficient sensor subset selection and link failure response for linear MMSE signal estimation in wireless sensor networks," in *Proc. of the European signal processing conference (EUSIPCO)*, Aalborg - Denmark, August 2010, pp. 1092–1096.

[18] H. Chen, A. Campbell, B. Thomas, and A. Tamir, "Minimax flow tree problems," *Networks*, vol. 54, pp. 117–129, March 2009.

[19] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artif. Intell.*, vol. 29, no. 3, pp. 241–288, 1986.

[20] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

[21] J. C. Bezdek and R. J. Hathaway, "Some notes on alternating optimization," in *Advances in Soft Computing*. Springer Berlin / Heidelberg, 2002, pp. 187–195.

Chapter 6

Linearly-Constrained DANSE

Distributed node-specific LCMV beamforming in wireless sensor networks

Alexander Bertrand and Marc Moonen

Submitted for publication, 2011.

Contributions of first author

- literature study
- co-development of the LC-DANSE$_K$ algorithm
- co-establishment of proof of convergence and optimality of LC-DANSE$_K$
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

In this paper, we consider the recently developed linearly constrained distributed adaptive node-specific signal estimation (LC-DANSE) algorithm, that generates an LCMV beamformer with node-specific constraints at each node of a wireless sensor network. The algorithm significantly reduces the number of signals that are exchanged between nodes, but obtains the node-specific LCMV beamformers as if each node has access to all the signals in the network. The LC-DANSE algorithm generalizes the DANSE algorithm for unconstrained linear MMSE signal estimation, by incorporating node-specific linear constraints in the estimation problems. We formally prove convergence and optimality of the LC-DANSE algorithm. We also consider the case where nodes update simultaneously instead of sequentially, and we demonstrate by means of simulations that applying relaxation is often required to obtain a converging algorithm in this case. We also provide simulation results that demonstrate the effectiveness of the algorithm in a realistic speech enhancement scenario.

6.1 Introduction

Many traditional spatial filtering or beamforming approaches assume a fixed sensor array with a limited number of wired sensors, where all sensor signal observations are gathered in a central processor. Therefore, the size of the array is often relatively small, resulting in only a local sampling of the spatial field, relatively large distances between the target source(s) and the array, and hence sensor signals with low signal-to-noise ratio (SNR) and low direct-to-reverberant ratio[1] (DRR). Recently, there has been a growing interest in distributed beamforming or signal estimation in wireless sensor networks (WSNs), where multiple sensor nodes are spatially distributed over an environment [1–6]. Each node consists of a small sensor array, a signal processing unit and a wireless communication link to communicate with other nodes. The advantage is that more sensors can be used to physically cover a wider area, and therefore there is a higher probability that a node is close to the target source, hence providing higher SNR signals.

One possibility is again to gather all the sensor signal observations in a dedicated device (the 'fusion center'), where an optimal beamformer can be computed. This approach is often referred to as centralized fusion or centralized estimation. Gathering all signal observations in a fusion center may however require a large communication bandwidth, a large transmission power at the individual nodes, and a significant computational power at the fusion center. Furthermore, in many sensor network applications, availability of a fusion center cannot be assumed. An alternative is a distributed approach where each

[1] A high DRR is important in, e.g., speech enhancement.

node has its own processing unit to exchange (possibly compressed) signal observations with other nodes. The nodes then cooperate in a distributed fashion to estimate a desired signal. This approach is preferred, especially so when it is scalable in terms of its communication bandwidth requirement and computational complexity.

In many beamforming applications, the estimation problem is defined by means of a local observation of the target source(s) in a local reference sensor (see, e.g., speech enhancement applications [7]). In a sensor network, this makes the estimation problem node-specific, since each node must choose its own local reference sensor. Furthermore, in some particular cases it may be required that the locally observed spatial information is still available in the estimate, e.g., if the estimation is followed by a localization task.

Distributed node-specific signal enhancement was first considered in a 2-node network, in the context of binaural hearing aids where it is important to preserve the spatial cues of the target source signals at both ears [8]. This technique relies on the speech-distortion-weighted multi-channel Wiener filter (SDW-MWF), and was referred to as distributed MWF (DB-MWF). In [9], distributed minimum variance distortionless response (DB-MVDR) beamforming was introduced for a similar binaural setting, which is essentially a limit case of the DB-MWF[2]. Both techniques assume a single target source to obtain convergence and optimality. In [1], a distributed adaptive node-specific signal estimation (DANSE) algorithm was introduced for fully connected WSNs, which generalizes DB-MWF to any number of nodes and multiple target sources. This has been extended to simply connected networks in [3], more robust versions of the algorithm [4], and versions with simultaneous and asynchronous node-updating [11].

In this paper, we consider distributed node-specific signal estimation, based on linearly constrained minimum variance (LCMV) beamforming. LCMV-beamforming is a well-known sensor array processing technique for noise reduction [12] where the goal is to minimize the output power of a multi-channel filter, under a set of linear constraints, e.g., to preserve target source signals and (fully or partially) cancel interferers. We refer to this distributed algorithm as linearly constrained distributed adaptive node-specific signal estimation (LC-DANSE), since it extends the DANSE algorithm, allowing to add node-specific linear constraints in the estimation problems of the different nodes. It can also be viewed as a generalization of the DB-MVDR algorithm from [9] to incorporate multiple sources (target sources and interferers), multiple node-specific constraints, and any number of nodes. However, it is noted that the LC-DANSE algorithm requires a completely different strategy to prove convergence compared to DB-MVDR, since the proof in [9] fully relies on a rank-1

[2]MVDR beamforming is equivalent to the SDW-MWF when the trade-off parameter $\mu \to 0$. When using a rank-1 model (in the case of a single target source), setting $\mu = 0$ in SDW-MWF gives exactly the same formula as MVDR beamforming [10].

data model and is not (straightforwardly) extended to more general models. We prove convergence of the LC-DANSE algorithm, and we show that the optimal linearly-constrained estimators are obtained as if each node has access to all the sensor signals in the network.

We consider an LCMV approach that operates without explicit knowledge of the array geometry or positions of the sources. However, this means that our approach is limited to scenarios that also lend themselves to subspace estimation of target sources and interferers. This is for example possible in speech enhancement, where both subspaces can be tracked based on non-stationarity and on-off behavior of the target source(s) [4, 9, 13, 14]. We also allow that the nodes solve node-specific LCMV problems, i.e. with different linear constraints. For example, a target source for one node may be an interfering source for another node. For the sake of an easy exposition, we only consider the case of fully connected networks. However, since the LC-DANSE algorithm has similar dynamics and parametrizations as the DANSE algorithm, it can also be applied in tree topology networks (see [3]). It is noted that the LC-DANSE algorithm has already been briefly introduced in [15]. In the present paper, we provide further details, i.e., we formally prove convergence and optimality of the LC-DANSE algorithm, we demonstrate by means of simulations that applying relaxation is often required to obtain a converging algorithm in the case where nodes update simultaneously, and we provide simulation results that demonstrate the effectiveness of the algorithm in a realistic speech enhancement scenario.

The outline of this paper is as follows. In Section 6.2, the estimation problem for each node in the network is described, and the centralized LCMV solution is explained. In Section 6.3, we explain how this centralized solution can be obtained in an iterative distributed fashion, by means of the LC-DANSE algorithm. We provide a convergence and optimality proof for the LC-DANSE algorithm in Section 6.4. In Section 6.5, we explain how relaxation can be applied to the LC-DANSE algorithm to obtain convergence in the case of simultaneous instead of sequential node-updating. Section 6.6 describes an application of the LC-DANSE algorithm, i.e., speech enhancement in a wireless acoustic sensor network, and provides simulation results for a realistic scenario. Conclusions are drawn in Section 6.7.

6.2 Centralized LCMV Beamforming

We consider a network with J sensor nodes where the set of nodes is denoted by $\mathcal{J} = \{1, ..., J\}$. Node k collects observations from a complex-valued[3] M_k-channel sensor signal $\mathbf{y}_k[t]$, where t is the time index which will be omitted in

[3]We assume that all signals are complex valued to incorporate frequency domain descriptions, e.g. in the short-time Fourier transform (STFT) domain.

the sequel. All $\mathbf{y}_k$'s are stacked in an M-channel signal $\mathbf{y}$ with $M = \sum_{k\in\mathcal{J}} M_k$. We assume that there are K relevant sources, i.e., target sources and interferers (we consider a source as relevant, if there is at least one node that uses this source in the linear constraints of its estimation problem, as explained later). We assume that $\mathbf{y}$ is generated by the linear model

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{6.1}$$

where $\mathbf{s}$ is a stacked signal vector containing the K relevant source signals, $\mathbf{H}$ is an $M \times K$ steering matrix from the K relevant sources to the M sensors, and $\mathbf{n}$ is a noise component. Sources that are not used in the linear constraints of any node (see below) are incorporated in $\mathbf{n}$.

In this section, we consider centralized LCMV beamforming, so we assume that each node k effectively has access to all channels of $\mathbf{y}$. Let $\mathcal{I}_k^d$ denote the set of indices that correspond to the N_k target source signals from $\mathbf{s}$ that node k aims to preserve in its node-specific estimation. The other $P_k = K - N_k$ source signals from $\mathbf{s}$ are assumed to be interferer signals, and their indices define the set $\mathcal{I}_k^n$. Similarly to [13], the goal for node k is to estimate the mixture of the N_k target source signals from $\mathbf{s}$ as they are observed by one of node k's sensors, referred to as the reference sensor (assume w.l.o.g. that this is the first sensor, i.e. y_{k1}).

It should be noted that we do not necessarily intend to unmix the sources in $\mathcal{I}_k^d$, since this would require to estimate the individual steering vector of each target source separately, which is often difficult or impossible. For example, in the case of speech enhancement, a voice activity detector (VAD) is required to estimate the speech subspace [1, 4, 13]. In a multiple speaker scenario, to estimate the steering vectors of each speaker separately, the VAD must be able to distinguish between different speakers (e.g. as in [14]). However, common VAD's are triggered by any (nearby) speakers, and therefore indeed only the joint subspace can be identified.

Let $\mathbf{Q}_k^d$ denote an $M \times N_k$ matrix with its columns defining a unitary basis for the target source subspace spanned by the columns of $\mathbf{H}$ with indices in $\mathcal{I}_k^d$. Similarly, let $\mathbf{Q}_k^n$ denote an $M \times P_k$ matrix containing a unitary basis for the interferer subspace corresponding to $\mathcal{I}_k^n$. Although it is usually impossible to estimate the individual columns of $\mathbf{H}$, the matrices $\mathbf{Q}_k^d$ and $\mathbf{Q}_k^n$ can often be estimated from the sensor signals $\mathbf{y}$ (see e.g. [13]). In the sequel, we assume that these matrices are indeed available.

Node k will apply a linear M-dimensional estimator $\mathbf{w}_k$ to the M-channel signal $\mathbf{y}$ to compute the signal $d_k = \mathbf{w}_k^H \mathbf{y}$ where H denotes the conjugate transpose operator. To this end, it will choose the $\mathbf{w}_k$ that minimizes the variance of d_k, while preserving the target source signals in $\mathcal{I}_k^d$. If required, other constraints can be added, e.g. to (fully or partially) block the interferers in $\mathcal{I}_k^n$. More

specifically, node k then solves the following centralized LCMV problem:

$$\min_{\mathbf{w}_k} E\{|\mathbf{w}_k^H \mathbf{y}|^2\} \tag{6.2}$$

s.t.

$$\mathbf{Q}_k^H \mathbf{w}_k = \mathbf{f}_k \tag{6.3}$$

with

$$\mathbf{Q}_k = \left[\begin{array}{cc} \mathbf{Q}_k^d & \mathbf{Q}_k^i \end{array} \right] \tag{6.4}$$

$$\mathbf{f}_k = \left[\begin{array}{c} \alpha \mathbf{q}_k^d(1) \\ \epsilon \mathbf{q}_k^n(1) \end{array} \right] \tag{6.5}$$

where $E\{.\}$ denotes the expected value operator, where $\mathbf{q}_k^d(1)$ and $\mathbf{q}_k^n(1)$ denote the first column of $\mathbf{Q}_k^{d\,H}$ and $\mathbf{Q}_k^{n\,H}$ respectively (corresponding to the reference sensor of node k), and where α and ϵ are user-defined gains[4]. The solution of this problem is given by [12]:

$$\hat{\mathbf{w}}_k = \mathbf{R}_{yy}^{-1} \mathbf{Q}_k \left(\mathbf{Q}_k^H \mathbf{R}_{yy}^{-1} \mathbf{Q}_k \right)^{-1} \mathbf{f}_k \tag{6.6}$$

with $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$. It can be shown [13] that the source signal components in the output $\hat{d}_k = \hat{\mathbf{w}}_k^H \mathbf{y}$ appear with the same mixing coefficients as in the reference sensor (except for the scaling by α or ϵ), i.e., we obtain

$$\hat{d}_k = \alpha \sum_{l \in \mathcal{I}^d} h_{1l} s_l + \epsilon \sum_{l \in \mathcal{I}^n} h_{1l} s_l + \hat{\mathbf{w}}_k^H \mathbf{n} \tag{6.7}$$

with h_{ij} denoting the entry in the i-th row and j-th column of $\mathbf{H}$. It is observed that this procedure indeed yields a distortionless response, which is not the case in SDW-MWF based beamforming techniques [8]. However, the constraints that enforce this distortionless response remove some degrees of freedom, yielding less noise reduction in the residual $\hat{\mathbf{w}}_k^H \mathbf{n}$.

6.3 Linearly Constrained DANSE (LC-DANSE)

In this section, we revisit the linearly constrained DANSE (LC-DANSE) algorithm, as introduced in [15]. In the next section, we will prove its convergence and optimality. For the sake of an easy exposition, we describe the algorithm for a fully connected network, but all results can be extended to simply connected tree topology networks, similar to [3]. Also for the sake of an easy exposition,

[4] Usually $\epsilon = 0$ to fully cancel the interferers. However in some cases it may be important to retain some undistorted residual noise, e.g. for hearing aid users to be able to reconstruct the acoustic environment. Note that α and ϵ can be chosen differently at different nodes k, but we did not add a subscript k, for the sake of an easier notation.

we describe a batch mode version of the algorithm, i.e. all iterations are performed on the full signal. However, in practice, iterations can be spread out over different signal observations, such that the same signal observations never need to be re-estimated nor retransmitted (we will discuss this later).

The compression of the sensor signal observations, and therefore the required bandwidth for LC-DANSE, directly depends on the number of relevant sources. In the case where there are K relevant sources, the nodes of LC-DANSE will exchange K-channel signal observations, yielding a compression with a factor of M_k/K at node k. We therefore assume[5] that $M_k > K$.

First, we define $K-1$ auxiliary estimation problems at node k, which are basically the same as (6.2)-(6.3) but with different choices for $\mathbf{f}_k$. This means that node k computes K different beamformer outputs $\mathbf{d}_k = \mathbf{W}_k^H\mathbf{y}$, defined by an $M \times K$ linear estimator $\mathbf{W}_k$ that solves

$$\min_{\mathbf{W}_k} E\{\|\mathbf{W}_k^H\mathbf{y}\|^2\} \tag{6.8}$$

s.t.

$$\mathbf{Q}_k^H\mathbf{W}_k = \mathbf{F}_k \tag{6.9}$$

where $\mathbf{F}_k$ is chosen as a full rank $K \times K$ matrix. The reason for adding these auxiliary estimation problems, is to obtain an estimator $\mathbf{W}_k$ that captures the full K-dimensional signal subspace defined by $\mathbf{s}$. The solution of (6.8)-(6.9) is

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^{-1}\mathbf{Q}_k\left(\mathbf{Q}_k^H\mathbf{R}_{yy}^{-1}\mathbf{Q}_k\right)^{-1}\mathbf{F}_k\ . \tag{6.10}$$

The first column of $\mathbf{F}_k$ corresponds to the constraints of node k, as in (6.5). The last $K-1$ columns of $\mathbf{F}_k$ are filled with random entries, as these define auxiliary problems. Therefore, the matrix $\mathbf{F}_k$ has the form

$$\mathbf{F}_k = \begin{bmatrix} \alpha\mathbf{q}_k^d(1) & v_{1,1} & \dots & v_{1,K-1} \\ \epsilon\mathbf{q}_k^n(1) & v_{2,1} & \dots & v_{2,K-1} \end{bmatrix} \tag{6.11}$$

where $v_{i,j}$ is a random number. However, the last $K-1$ columns of $\mathbf{F}_k$ may also be filled with constraints that define other estimation problems for node k that use the same subspace partitioning of the two subspaces $\mathbf{Q}_k^d$ and $\mathbf{Q}_k^n$:

$$\mathbf{F}_k = \begin{bmatrix} \alpha\mathbf{q}_k^d(1) & \alpha_1\mathbf{q}_k^d(m_1) & \dots & \alpha_{K-1}\mathbf{q}_k^d(m_{K-1}) \\ \epsilon\mathbf{q}_k^n(1) & \epsilon_1\mathbf{q}_k^n(n_1) & \dots & \epsilon_{K-1}\mathbf{q}_k^n(n_{K-1}) \end{bmatrix} \tag{6.12}$$

where m_j, $n_j \in \{1,\dots,M_k\}$ and where α_j,$\epsilon_j \in \mathbb{C}$ are chosen such that $\mathbf{F}_k$ is full rank (note that these can be chosen differently for different nodes k, even

[5] In the fully connected case, LC-DANSE only has a compression benefit if $M_k > K$, i.e., in this case the number of exchanged signals can be reduced without harming the optimality results. In the case of a simply connected topology (see [3]), there is still a compression benefit compared to the scenario where all signals are relayed, even if $M_k < K$.

though we did not add a subscript k). Either (6.11) or (6.12) can be used in the sequel.

In the LC-DANSE algorithm, $\mathbf{y}_k$ is linearly compressed to a K-channel signal $\mathbf{z}_k$ (the compression rule will be defined later, namely in formula (6.18)), of which observations are then broadcast to the remaining $J-1$ nodes [15]. We define the $(K(J-1))$-channel signal $\mathbf{z}_{-k} = \left[\mathbf{z}_1^T \dots \mathbf{z}_{k-1}^T \mathbf{z}_{k+1}^T \dots \mathbf{z}_J^T\right]^T$. Node k then collects observations of its own sensor signals in $\mathbf{y}_k$ together with observations of the signals in $\mathbf{z}_{-k}$ obtained from the other nodes in the network. Similarly to the centralized LCMV approach, node k can then compute the $(M_k + K(J-1))$-channel LCMV beamformer $\mathbf{U}_k$ with respect to these input signals, i.e. the solution of

$$\min_{\mathbf{U}_k} \; E\{\|\mathbf{U}_k^H \tilde{\mathbf{y}}_k\|^2\} \tag{6.13}$$

s.t.

$$\widetilde{\mathbf{Q}}_k^H \mathbf{U}_k = \widetilde{\mathbf{F}}_k \tag{6.14}$$

where

$$\tilde{\mathbf{y}}_k = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k} \end{bmatrix} \tag{6.15}$$

and with $\widetilde{\mathbf{Q}}_k$ denoting the equivalent to $\mathbf{Q}_k$, but now with respect to the modified steering vectors corresponding to node k's input signals, i.e., $\tilde{\mathbf{y}}_k$. These will be linearly compressed versions of the steering vectors in $\mathbf{H}$, due to the linear compression rules that generate the $\mathbf{z}_k$'s. Again, we assume that the subspaces spanned by the steering vectors, i.e. $\widetilde{\mathbf{Q}}_k$, can be estimated from the input signals. $\widetilde{\mathbf{F}}_k$ is constructed similarly to (6.11) or (6.12), but now based on the columns of $\widetilde{\mathbf{Q}}_k^H$ instead of $\mathbf{Q}_k^H$ (the random entries in (6.11) do not change).

The problem (6.13)-(6.14) is equivalent to the centralized LCMV problem described in Section 6.2 (but with fewer signals), and its solution can be computed in exactly the same way.

We now define the partitioning

$$\mathbf{U}_k = \left[\mathbf{W}_{kk}^T | \mathbf{G}_{k,-k}^T\right]^T \tag{6.16}$$

$$= \left[\mathbf{W}_{kk}^T | \mathbf{G}_{k1}^T | \dots | \mathbf{G}_{k,k-1}^T | \mathbf{G}_{k,k+1}^T | \dots | \mathbf{G}_{kJ}^T\right]^T \tag{6.17}$$

where $\mathbf{W}_{kk}$ contains the first M_k rows of $\mathbf{U}_k$ (which are applied to node k's own sensor signals $\mathbf{y}_k$) and where $\mathbf{G}_{kq}$ is the part of $\mathbf{U}_k$ that is applied to the K-channel signal $\mathbf{z}_q$ obtained from node q. We can now also define the compression rule to generate the broadcast signal $\mathbf{z}_k$ as

$$\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k \; . \tag{6.18}$$

A schematic illustration of this is shown in Fig. 6.1, for a network with $J = 3$ nodes. It should be noted that $\mathbf{W}_{kk}$ both acts as a compressor and as a part of

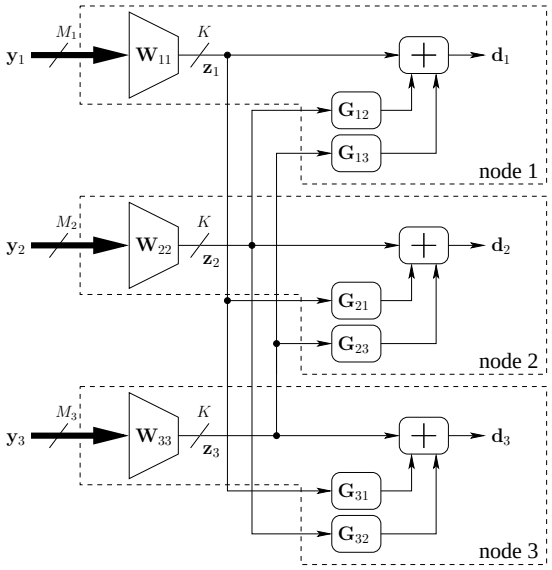


Figure 6.1: The LC-DANSE scheme with 3 nodes ($J = 3$). Each node k computes an LCMV beamformer using its own M_k-channel sensor signal observations, and 2 K-channel signals broadcast by the other two nodes.

the estimator $\mathbf{W}_k$. Based on Fig. 6.1, it can be seen that the parametrization of $\mathbf{W}_k$ effectively applied at node k, to generate $\mathbf{d}_k = \mathbf{U}_k^H \tilde{\mathbf{y}}_k = \mathbf{W}_k^H \mathbf{y}$, is then

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{kJ} \end{bmatrix} \tag{6.19}$$

where we assume that $\mathbf{G}_{kk} = \mathbf{I}_K$ with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix. This is exactly the same parametrization as used in the DANSE algorithm [1]. If we define the partitioning $\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_{k1}^T & \dots & \mathbf{W}_{kJ}^T \end{bmatrix}^T$, where $\mathbf{W}_{kq}$ is the part of $\mathbf{W}_k$ that is applied to the sensor signals of node q, i.e. $\mathbf{y}_q$, then (6.19) is equivalent to

$$\mathbf{W}_{kq} = \mathbf{W}_{qq}\mathbf{G}_{kq}, \; \forall\, k, q \in \mathcal{J} \; . \tag{6.20}$$

Expression (6.19) or (6.20) defines a solution space for all $\mathbf{W}_k$, $k \in \mathcal{J}$, simultaneously, where node k can only control the parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_{k,-k}$. This solution space captures all optimal LCMV solutions (for every node), as was proven in [15].

The LC-DANSE algorithm iteratively updates the parameters in (6.19), by letting each node k compute (6.13)-(6.14), $\forall k \in \mathcal{J}$, in a sequential round robin fashion:

The LC-DANSE Algorithm

1. Initialize $i \leftarrow 0$, $k \leftarrow 1$, and initialize all $\mathbf{U}_q^0 = \left[\mathbf{W}_{qq}^{0\,T} | \mathbf{G}_{q,-q}^{0\,T}\right]^T$, $\forall\, q \in \mathcal{J}$, with random entries.
2. Set $\mathbf{z}_q^i = \mathbf{W}_{qq}^{i\,H}\mathbf{y}_q$, $\forall\, q \in \mathcal{J}$.
3. Update $\widetilde{\mathbf{Q}}_k^i$ by computing a unitary basis for the desired and interferer subspace with respect to the inputs at node k (i.e. the channels of $\tilde{\mathbf{y}}_k^i$).
4. Update $\widetilde{\mathbf{F}}_k^i$ according to either
$$\widetilde{\mathbf{F}}_k^i = \begin{bmatrix} \alpha\tilde{\mathbf{q}}_k^d(1) & v_{1,1} & \dots & v_{1,K-1} \\ \epsilon\tilde{\mathbf{q}}_k^n(1) & v_{2,1} & \dots & v_{2,K-1} \end{bmatrix} \tag{6.21}$$
or
$$\widetilde{\mathbf{F}}_k^i = \begin{bmatrix} \alpha\tilde{\mathbf{q}}_k^d(1) & \alpha_1\tilde{\mathbf{q}}_k^d(m_1) & \dots & \alpha_{K-1}\tilde{\mathbf{q}}_k^d(m_{K-1}) \\ \epsilon\tilde{\mathbf{q}}_k^n(1) & \epsilon_1\tilde{\mathbf{q}}_k^n(n_1) & \dots & \epsilon_{K-1}\tilde{\mathbf{q}}_k^n(n_{K-1}) \end{bmatrix} \tag{6.22}$$
where $\tilde{\mathbf{q}}_k^d(j)$ and $\tilde{\mathbf{q}}_k^n(j)$ denote the j-th column of $\widetilde{\mathbf{Q}}_k^{d\,H}$ and $\widetilde{\mathbf{Q}}_k^{n\,H}$ respectively.
5. Update $\mathbf{U}_k^i$ to $\mathbf{U}_k^{i+1} = \left[\mathbf{W}_{kk}^{i+1\,T} | \mathbf{G}_{k,-k}^{i+1\,T}\right]^T$ according to the solution of (6.13)-(6.14), while the other nodes do not perform any updates, i.e. $\mathbf{U}_q^{i+1} = \mathbf{U}_q^i = \left[\mathbf{W}_{qq}^{i\,T} | \mathbf{G}_{q,-q}^{i\,T}\right]^T$, $\forall\, q \in \mathcal{J}\backslash\{k\}$.
6. $i \leftarrow i+1$ and $k \leftarrow (k \mod J) + 1$.
7. Return to step 2.

It is noted that we are generally not interested in $\mathbf{W}_k$, but rather in the first channel of its output signal $\mathbf{d}_k$. The sample $d_k[t]$ in node k at any point in the iterative process is computed as

$$d_k^i[t] = \mathbf{w}_{kk}^{i\,H}\mathbf{y}_k[t] + \sum_{q\neq k} \mathbf{g}_{kq}^{i\,H}\mathbf{z}_q^i[t] \tag{6.23}$$

where $\mathbf{w}_{kk}^i$ and $\mathbf{g}_{kq}^i$ denote the first column of $\mathbf{W}_{kk}^i$ and $\mathbf{G}_{kq}^i$.

Remark I: The iterative nature of the LC-DANSE algorithm may suggest that the same sensor signal observations are compressed and broadcast multiple times, i.e. once after every iteration. However, we emphasize that the algorithm is in fact a distributed beamforming approach, usually only exploiting spatial information, and iterations can therefore be spread out over time (over different observations). This means that only the local fusion rules are iteratively updated. In other words, if $\mathbf{W}_{kk}^i$ is updated to $\mathbf{W}_{kk}^{i+1}$ at time t_0, this updated version is only used to produce the observations $\mathbf{z}_k[t]$ and $d_k[t]$

for $t > t_0$, while previous observations for $t \leq t_0$, are neither recompressed nor retransmitted. Effectively, each sensor signal observation is compressed and transmitted only once. The non-batch description of LC-DANSE is similar to the non-batch description of DANSE in [1], and is omitted here.

Remark II: Using (6.11) instead of (6.12) often yields a better conditioned system, i.e. the cross-correlation matrix $E\{\mathbf{z}_k^i \mathbf{z}_k^{i\,H}\}$ is better conditioned, which may yield better results in practical scenarios [4]. As explained in Section 6.4, the output signal (6.23) of each node will still be equal to the optimal LCMV beamformer output (6.7) after convergence of the algorithm, but the last $K-1$ columns of the $\mathbf{W}_k$'s will not be equal to the corresponding LCMV-beamformers of the centralized optimization problem in this case. This is however not a problem, since these are auxiliary estimators. If (6.12) is used, the last $K-1$ columns of the $\mathbf{W}_k$'s will be equal to the corresponding centralized LCMV-beamformers defined by the solution of (6.8)-(6.9).

Remark III: In the beginning of this section, we assumed that $M_k > K$, $\forall\, k \in \mathcal{J}$. However, if there would be a node k where $M_k \leq K$, it can broadcast its raw sensor signals to the other nodes, i.e., $\mathbf{z}_k = \mathbf{y}_k$. Another node q will incorporate these in its local node-specific estimation problem, by means of a non-square $M_k \times K$ matrix $\mathbf{G}_{qk}$. This will not affect convergence or optimality of the LC-DANSE algorithm, but there is no compression at node k.

Remark IV: It is noted that the local constraints in each node of the LC-DANSE algorithm change over different iterations. However, we will prove that, despite these changing constraints, the algorithm still converges to the output of the centralized LCMV beamformer as if the full constraints in (6.3) are applied at each node k.

6.4 Convergence and Optimality of LC-DANSE

The following theorems guarantee convergence and optimality of LC-DANSE:

Theorem 6.1 *If $\mathbf{F}_k$ is full rank, $\forall k \in \mathcal{J}$, and if $\widetilde{\mathbf{F}}_k$ is updated based on (6.22), then all parameters of the LC-DANSE algorithm converge. Furthermore, if $i \to \infty$, the output signal d_k^i defined in (6.23) is equal to the output signal of the centralized algorithm defined in (6.7), $\forall\, k \in \mathcal{J}$, and the estimator $\lim_{i\to\infty} \mathbf{W}_k^i$ parametrized by (6.19) is equal to $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$.*

Theorem 6.2 *If $\mathbf{F}_k$ is full rank, $\forall k \in \mathcal{J}$, and if $\widetilde{\mathbf{F}}_k$ is updated based on (6.21), then all parameters of the LC-DANSE algorithm converge. Furthermore, if $i \to \infty$, the output signal d_k^i defined in (6.23) is equal to the output signal of the centralized algorithm defined in (6.7), $\forall\, k \in \mathcal{J}$, and the first column of the*

estimator $\lim_{i\to\infty} \mathbf{W}_k^i$ *parametrized by (6.19) is equal to the first column of* $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$.

We will only elaborate on the proof of Theorem 6.1. As explained in Subsection 6.4.2, the proof of Theorem 6.1 can be easily modified to also prove Theorem 6.2, i.e., the case where the $\mathbf{F}_k$'s satisfy the more general structure (6.11).

6.4.1 Proof of Theorem 6.1

We consider the estimation problem (6.8)-(6.9) of node k. Instead of solving the problem directly by means of expression (6.10), we solve it iteratively with the following (centralized) Gauss-Seidel type procedure[6], where it is assumed that node k has access to all the channels of $\mathbf{y}$:

1. Initialize $i \leftarrow 0$, $q \leftarrow 1$ and initialize $\mathbf{W}_k^0$ with random entries.
2. Obtain $\mathbf{W}_k^{i+1}$ as the solution of the following constrained optimization problem:
$$\mathbf{W}_k^{i+1} = \arg\min_{\mathbf{W}_k} E\{\|\mathbf{W}_k^H \mathbf{y}\|^2\} \tag{6.24}$$
s.t.
$$\mathbf{Q}_k^H \mathbf{W}_k = \mathbf{F}_k \tag{6.25}$$
$$\forall\, j \in \mathcal{J}\backslash\{q\},\ \exists\, \mathbf{C}_j \in \mathbb{C}^{K\times K} :\ \mathbf{W}_{kj} = \mathbf{W}_{kj}^i \mathbf{C}_j\ . \tag{6.26}$$
3. $i \leftarrow i+1$ and $q \leftarrow (q \mod J) + 1$.
4. Return to step 2.

In each step of the algorithm, an LCMV beamformer $\mathbf{W}_k$ is computed under additional linear constraints, i.e. certain blocks of $\mathbf{W}_k$ are constrained to a K-dimensional subspace defined by the columns of the current block. This procedure will always converge to the optimal LCMV-beamformer (6.10), which follows from the strict convexity of the optimization problem in each iteration and the monotonic decrease of the cost function, i.e., $E\{\|\mathbf{W}_k^{i+1\,H}\mathbf{y}\|^2\} \leq E\{\|\mathbf{W}_k^{i\,H}\mathbf{y}\|^2\}$. We will explain how the LC-DANSE algorithm mimics this iterative procedure, despite the fact that the latter is a centralized algorithm. Convergence and optimality of the Gauss-Seidel procedure will then imply convergence and optimality of the LC-DANSE algorithm.

We first transform the centralized Gauss-Seidel procedure to a form that is more closely related to the LC-DANSE algorithm. By parametrizing the optimization problem (6.24)-(6.26), the constraints in (6.26) can be eliminated. The problem (6.24)-(6.26) is then equivalent to computing the $(M_1+(J-1)K)\times K$ matrix $\mathbf{U}_k$ that solves
$$\min_{\mathbf{U}_k} E\{\|\mathbf{U}_k^H \mathcal{W}_{k,q}^i \mathbf{y}\|^2\} \tag{6.27}$$

[6]Strictly speaking, this procedure is not a Gauss-Seidel procedure, since none of the variables are actually fixed. Instead, some of them are constrained to a certain subspace.

s.t.

$$\overline{\mathbf{Q}}_k^{i\,H}\mathbf{U}_k = \mathbf{F}_k \tag{6.28}$$

where

$$\overline{\mathbf{Q}}_k^i = \mathcal{W}_{k,q}^i \mathbf{Q}_k \tag{6.29}$$

$$\mathcal{W}_{k,q}^i = \text{blockdiag}\left\{\mathbf{W}_{k1}^{i\,H}, \ldots, \mathbf{W}_{k,q-1}^{i\,H}, \mathbf{I}_{M_q}, \mathbf{W}_{k,q+1}^{i\,H}, \ldots, \mathbf{W}_{kJ}^{i\,H}\right\} \tag{6.30}$$

with $\mathbf{I}_{M_q}$ denoting an $M_q \times M_q$ identity matrix and with blockdiag$\{.\}$ denoting the operator that creates a block diagonal matrix with the entries in its argument as the diagonal blocks. The solution $\mathbf{W}_k^{i+1}$ of (6.24)-(6.26) is then given by $\mathbf{W}_k^{i+1} = \mathcal{W}_{k,q}^{i\,H}\mathbf{U}_k$. It is noted that this solution does not change when we transform the constraint space to a unitary basis, i.e. when we replace the constraint (6.28) by

$$\widetilde{\mathbf{Q}}_k^{i\,H}\mathbf{U}_k = \widetilde{\mathbf{F}}_k^i \tag{6.31}$$

where

$$\widetilde{\mathbf{Q}}_k^{i\,H} = \mathbf{T}_k^i \overline{\mathbf{Q}}_k^{i\,H}, \quad \widetilde{\mathbf{F}}_k^i = \mathbf{T}_k^i \mathbf{F}_k \tag{6.32}$$

and where $\mathbf{T}_k^i$ is a $K \times K$ matrix that makes the rows of $\widetilde{\mathbf{Q}}_k^{i\,H}$ unitary (i.e., $\widetilde{\mathbf{Q}}_k^{i\,H}\widetilde{\mathbf{Q}}_k^i = \mathbf{I}_K$). Therefore, the above Gauss-Seidel iteration can be replaced by the following equivalent procedure:

1. Initialize $i \leftarrow 0$, $q \leftarrow 1$ and initialize $\mathbf{W}_k^0$ with random entries.
2.
   - Compute $\overline{\mathbf{Q}}_k^i$ according to (6.29).
   - Transform $\overline{\mathbf{Q}}_k^i$ to obtain $\widetilde{\mathbf{Q}}_k^i$, such that $\widetilde{\mathbf{Q}}_k^{i\,H}\widetilde{\mathbf{Q}}_k^i = \mathbf{I}_K$, and compute a corresponding $\widetilde{\mathbf{F}}_k^i$, as in (6.32).
   - Compute $\mathbf{U}_k$ as the solution of (6.27) s.t. (6.31).
   - Update $\mathbf{W}_k^{i+1} = \mathcal{W}_{k,q}^{i\,H}\mathbf{U}_k$.
3. $i \leftarrow i+1$ and $q \leftarrow (q \mod J) + 1$.
4. Return to step 2.

We refer to this procedure as centralized Gauss-Seidel$_k$ or CGS$_k$, where the subscript k indicates that it solves the estimation problem of node k. A careful look at a single iteration of this CGS$_k$ procedure shows that it is basically what an updating node q solves in the LC-DANSE algorithm at an iteration where $q = k$. However, in other iterations where $q \neq k$, this is not the case, since CGS$_k$ then requires that node k has access to observations of the full signal vector $\mathbf{y}$, and not only to its own sensor signals in $\mathbf{y}_k$. Let us now assume that this CGS$_k$ procedure is performed for all $k \in \mathcal{J}$ in parallel but independently from each other. We then have the following lemma:

Lemma 6.3 *Assume that all the CGS$_k$ procedures are performed in parallel (for all $k \in \mathcal{J}$) and that $\mathbf{F}_k$ has full rank, $\forall\, k \in \mathcal{J}$, then the following holds:*

If for any $k, q \in \mathcal{J}$, there exists a full rank $K \times K$ matrix $\mathbf{A}_{kq}^i$ such that

$$\mathbf{W}_k^i = \mathbf{W}_q^i \mathbf{A}_{kq}^i \tag{6.33}$$

then there exists a full rank $K \times K$ matrix $\mathbf{A}_{kq}^{i+1}$ such that

$$\mathbf{W}_k^{i+1} = \mathbf{W}_q^{i+1} \mathbf{A}_{kq}^{i+1} \tag{6.34}$$

Proof : See Appendix 6.A. □

This is a key result to prove convergence of LC-DANSE. If the CGS_k procedures (for all $k \in \mathcal{J}$) are initialized with the same matrix $\mathbf{W}_k^0 = \mathbf{W}^0$, $\forall\, k \in \mathcal{J}$, then Lemma 6.3 basically says that their intermediate solutions (observed at each iteration) will always be equal up to a $K \times K$ transformation, even though they solve optimization problems with different constraints.

At this point, let us return to the LC-DANSE algorithm, where each node k, $\forall\, k \in \mathcal{J}$, initializes its $\mathbf{W}_{kk}^0$ with random entries. For now, we assume that all $\mathbf{G}_{kq}^0$, $\forall\, k, q \in \mathcal{J}$, are initialized with an identity matrix $\mathbf{I}_K$ (we will return to the general case later). Based on the parametrization (6.19), this means that $\widetilde{\mathbf{W}}_k^0 = \mathbf{W}^0$, $\forall\, k \in \mathcal{J}$, where

$$\mathbf{W}^0 = \left[\mathbf{W}_{11}^{0\,T} | \ldots | \mathbf{W}_{JJ}^{0\,T}\right]^T \tag{6.35}$$

(we added a tilde to refer to the LC-DANSE estimators ($\widetilde{\mathbf{W}}_k$), whereas the $\mathbf{W}_k$'s without a tilde refer to the CGS_k estimators). We also define

$$\mathbf{G}_k = \left[\mathbf{G}_{k1}^T | \ldots | \mathbf{G}_{kJ}^T\right]^T \ . \tag{6.36}$$

Notice that the $\mathbf{G}_{k,-k}$ as defined in (6.17) then corresponds to $\mathbf{G}_k$ with $\mathbf{G}_{kk}$ omitted. It is noted that the parameter $\mathbf{G}_{kk}$ is a variable that is not explicitly used in the LC-DANSE algorithm, since it is always assumed to be an identity matrix. We will now run the LC-DANSE algorithm in parallel with all the CGS_k procedures (one for each node), and we will compare the updates of LC-DANSE (iteration per iteration) with those of the centralized algorithms. We assume that the CGS_k procedures are all initialized with the same matrix $\mathbf{W}^0$, and therefore (6.33) will be satisfied in every iteration, for any pair $k, q \in \mathcal{J}$. We then have the following corollary from Lemma 6.3, which couples the LC-DANSE solutions to the CGS_k solutions in a particular way:

Corollary 6.4 *Assume that we are in iteration i of the LC-DANSE algorithm in which node k updates its parameters, and assume that the CGS_k estimator has the same column space as the LC-DANSE estimator at node k, i.e., $\mathbf{W}_k^i = \widetilde{\mathbf{W}}_k^i \mathbf{C}_k^i$ with $\mathbf{C}_k^i$ a full rank $K \times K$ matrix. Furthermore, assume that, for any*

$q \in \mathcal{J}$, the CGS$_q$ estimator satisfies (6.33). Then there exists a full rank $K \times K$ matrix $\mathbf{B}_{qk}^{i+1}$ such that, if node q would update its $\mathbf{G}_q^i$ parameters according to[7]

$$\mathbf{G}_q^{i+1} = \mathbf{G}_k^{i+1}\mathbf{B}_{qk}^{i+1} \tag{6.37}$$

then

$$\mathbf{W}_q^{i+1} = \widetilde{\mathbf{W}}_q^{i+1} \, . \tag{6.38}$$

Proof : Since node k updates at iteration i in LC-DANSE, and since $\mathbf{W}_k^i = \widetilde{\mathbf{W}}_k^i \mathbf{C}_k^i$, it holds that

$$\mathbf{W}_k^{i+1} = \widetilde{\mathbf{W}}_k^{i+1} \tag{6.39}$$

since both optimization procedures have the same constraint set. Furthermore, if (6.37) holds, we know that

$$\widetilde{\mathbf{W}}_q^{i+1} = \widetilde{\mathbf{W}}_k^{i+1}\mathbf{B}_{qk}^{i+1} \, . \tag{6.40}$$

Because (6.33) holds, we also know from Lemma 6.3 that

$$\mathbf{W}_q^{i+1} = \mathbf{W}_k^{i+1}\mathbf{A}_{qk}^{i+1} \, . \tag{6.41}$$

By substituting (6.39) in (6.40), and by choosing $\mathbf{B}_{qk}^{i+1} = \mathbf{A}_{qk}^{i+1}$, we obtain

$$\widetilde{\mathbf{W}}_q^{i+1} = \mathbf{W}_k^{i+1}\mathbf{A}_{qk}^{i+1} \, . \tag{6.42}$$

Comparing (6.41) and (6.42) yields (6.38). □

Notice that the conditions of Corollary 6.4 are satisfied in the initial phase (iteration 1), since all algorithms are initialized with the same matrix $\mathbf{W}^0$. Now assume that LC-DANSE would be (hypothetically) modified such that each node q performs the extra update (6.37) (assuming that $\mathbf{B}_{qk}^{i+1}$ is known) at every update (where k changes according to the updating node in the LC-DANSE algorithm). We will refer to this modified LC-DANSE algorithm as hypothetical LC-DANSE (H-LC-DANSE). Corollary 6.4 then basically says that the CGS estimators $\{\mathbf{W}_k^i\}_{k\in\mathcal{J}}$ are always equal to the H-LC-DANSE estimators $\{\widetilde{\mathbf{W}}_k^i\}_{k\in\mathcal{J}}$. Since the CGS procedures converge to the optimal estimators, the H-LC-DANSE estimators will therefore also converge to the optimal estimators.

To prove convergence of the actual LC-DANSE algorithm, observe that the values of the $\mathbf{G}_k$'s have no impact at all on the updates of the $\mathbf{W}_{kk}$'s or the $\mathbf{U}_k$'s in general[8]. Indeed, if node q updates its $\mathbf{U}_q^i$ at iteration i, this is not influenced by the choices of $\mathbf{G}_k^i$ at other nodes $k \neq q$, since the $\mathbf{z}_k$ signals

[7] Note that G_{qq}^{i+1} does not necessarily have to be equal to the identity matrix here.

[8] And therefore, initializing LC-DANSE with $\mathbf{G}_{kq}^0 = \mathbf{I}_K$, $\forall\, k, q \in \mathcal{J}$, does not affect the generality of the proof, i.e. they may as well be initialized with random entries.

only depend on the $\mathbf{W}_{kk}$'s. Therefore, the set $\{\mathbf{W}_{kk}^i\}_{k\in\mathcal{J}}$ at any iteration i will always be the same for the H-LC-DANSE and the actual LC-DANSE algorithm (when initialized with the same set $\{\mathbf{W}_{kk}^0\}_{k\in\mathcal{J}}$). Since the former converges to the optimal estimators, the set $\{\mathbf{W}_{kk}^i\}_{k\in\mathcal{J}}$ of LC-DANSE must also converge to the same (optimal) values when $i \to \infty$. Convergence and optimality of the parameters $\{\mathbf{W}_{kk}^i\}_{k\in\mathcal{J}}$ straightforwardly implies convergence and optimality of the parameters $\{\mathbf{G}_{k,-k}^i\}_{k\in\mathcal{J}}$ of LC-DANSE. This proves full convergence and optimality of the LC-DANSE algorithm.

Remark: It is noted that, at every iteration where a particular node k updates, its estimator $\widetilde{\mathbf{W}}_k^{i+1}$ will be equal to the CGS_k estimator $\mathbf{W}_k^{i+1}$ (this also holds for LC-DANSE, not only for H-LC-DANSE, as long as the CGS procedures are initialized with the same $\mathbf{W}^0$ as the LC-DANSE algorithm.). This reveals that the LC-DANSE algorithm is as fast as the centralized Gauss-Seidel algorithm described in the proof. This also implies that the cost function $J_k(\mathbf{W}_k) = E\{\|\mathbf{W}_k^H\mathbf{y}\|^2\}$ decreases monotonically at each node k, when evaluated after each update at node k. This is similar to the monotonic decrease of the unconstrained DANSE algorithm of [1].

6.4.2 Proof of Theorem 6.2

The proof of Theorem 6.1 basically also proves Theorem 6.2. Even though (6.32) is not fully satisfied anymore, since the last $K-1$ columns in $\widetilde{\mathbf{F}}_k^i$ never change over the iterations, the resulting $\mathbf{z}_k^i$'s still span the same K-dimensional signal subspaces as in the case where $\widetilde{\mathbf{F}}_k^i$ changes according to (6.32). Since the $\mathbf{G}_{qk}$'s can compensate for this, the other nodes are still able to obtain the same estimator as if $\widetilde{\mathbf{F}}_k^i$ changes according to (6.32). However, since the last $K-1$ columns of $\widetilde{\mathbf{F}}_k^i$ do not change along with the input signals over the different iterations, only the first column of the resulting estimator will be equal to $\hat{\mathbf{W}}_k$.

6.5 LC-DANSE with Simultaneous Node-Updating

The LC-DANSE algorithm, as described in Section 6.3 assumes that the nodes update in a sequential round-robin fashion. However, due to this sequential updating scheme, only one node at a time can estimate the statistics of its input signals and perform an update of its parameters. Since every such parameter update at node k changes the statistics of the node's broadcast signal $\mathbf{z}_k$, it takes some time before the next node has collected sufficient data to compute a reliable estimate of the modified signal statistics and then update its parameters. As a result, even though the LC-DANSE algorithm converges in a small number of iterations, it may converge slowly in time, especially so when the number of nodes is large.

If alternatively, nodes would perform their updates simultaneously, the algorithm can adapt more swiftly, and all nodes can then estimate the signal statistics in parallel. Similar to the results in [11] for unconstrained DANSE, convergence can no longer be guaranteed for this case, as will be demonstrated by means of simulations in Section 6.6. In [11], it is suggested to relax the parameter updates of the unconstrained DANSE algorithm, to again obtain a converging algorithm. A similar procedure can be applied to LC-DANSE to let the algorithm converge under simultaneous node-updating.

The relaxed LC-DANSE algorithm with simultaneous node-updating is then defined as follows.

Relaxed LC-DANSE Algorithm with Simultaneous Node-Updating

1. Initialize $i \leftarrow 0$, $k \leftarrow 1$, and initialize all $\mathbf{U}_q^0 = \left[\mathbf{W}_{qq}^{0\,T} | \mathbf{G}_{q,-q}^{0\,T}\right]^T$, $\forall\, q \in \mathcal{J}$, with random entries.
2. Set $\mathbf{z}_q^i = \mathbf{W}_{qq}^{i\,H} \mathbf{y}_q$, $\forall\, q \in \mathcal{J}$.
3. For all $k \in \mathcal{J}$ simultaneously:
   - Update $\widetilde{\mathbf{Q}}_k^i$ by computing a unitary basis for the desired and interferer subspace with respect to the inputs at node k (i.e. the channels of $\tilde{\mathbf{y}}_k^i$).
   - Update $\widetilde{\mathbf{F}}_k^i$ according to either (6.21) or (6.22).
   - Compute the solution of (6.13)-(6.14) and store it in $\overline{\mathbf{U}}_k = \left[\overline{\mathbf{W}}_{kk}^T | \overline{\mathbf{G}}_{k,-k}^T\right]^T$.
   - Choose a relaxation stepsize $\alpha^i \in (0, 1]$.
   - Perform the relaxed update

$$\mathbf{W}_{kk}^{i+1} = (1 - \alpha^i)\mathbf{W}_{kk}^i + \alpha^i \overline{\mathbf{W}}_{kk} \tag{6.43}$$

$$\mathbf{G}_{k,-k}^{i+1} = \overline{\mathbf{G}}_{k,-k} \tag{6.44}$$

4. $i \leftarrow i + 1$ and $k \leftarrow (k \mod J) + 1$.
5. Return to step 2.

Usually, a fixed relaxation stepsize is chosen, e.g. $\alpha^i = 0.5, \forall i \in \mathbb{N}$. Simulations demonstrate that the algorithm converges if α is chosen sufficiently small. This is stated here as an observation, based on extensive simulation results, since a formal proof is not available. However, the intuitive explanation why relaxation helps to let the algorithm converge is the same as in [11]. Relaxation can also be applied to obtain convergence in the case of LC-DANSE with asynchronous node-updating (see [11]), i.e., the case where nodes can decide for themselves

when and how often they update their parameters.

6.6 Application: Noise reduction in an Acoustic Sensor Network

In [15], the convergence and optimality of LC-DANSE has been briefly addressed by means of a numerical simulation on a toy scenario, where the data model (6.1) was perfectly satisfied. In this paper, we provide simulations of the LC-DANSE algorithm in a more realistic scenario, i.e. for speech enhancement in a wireless acoustic sensor network. Since we consider convolutive mixtures, the problem will be solved in the short-time Fourier transform (STFT) domain, where the data model (6.1) is only approximately satisfied.

6.6.1 The Acoustic Scenario

A multi-source acoustic scenario is simulated using the image method [16]. Fig. 6.2 shows a schematic illustration of the scenario. The room is rectangular (5m $\times$ 5m $\times$ 3m) with a reflection coefficient of 0.4 for the floor, the ceiling and for every wall. According to Sabine's formula this corresponds to a reverberation time of $T_{60} = 0.3$ s. There are two speakers (A and B), who produce speech sentences from the HINT ('Hearing in Noise Test') database [17]. There are two localized noise sources that produce (mutually uncorrelated) babble noise with the same power as the speech sources. In addition to the localized noise sources, all microphone signals have an uncorrelated noise component which consist of white noise with power that is 20% of the power of the (superimposed) speech signals in the first microphone of node 1. The acoustic sensor network is fully connected and consists of $J = 4$ nodes, each having $M_k = 4$ omnidirectional microphones with a spacing of 3 cm. All nodes and all sound sources are in the same horizontal plane, 2 m above ground level. The sampling frequency is 16 kHz in all experiments.

6.6.2 Problem Statement

The goal for each node is to obtain an undistorted estimate of one speech source as it impinges on one of the node's microphones, with full suppression of the other (interfering) speech source, while reducing as much background noise as possible. Since there are 2 relevant sources (speakers A and B), we choose $K = 2$. To obtain the aforementioned goal at node k, it chooses a matrix $\mathbf{F}_k$ as in (6.12) with $\alpha = 1$, $\epsilon = 0$, $\alpha_1 = 0$ and $\epsilon_1 = 1$. If $\mathcal{I}_k^d$ contains speaker A, and $\mathcal{I}_k^n$ contains speaker B, then the first column of $\mathbf{W}_k$ will estimate speaker A while suppressing speaker B (and the second column will estimate speaker B while

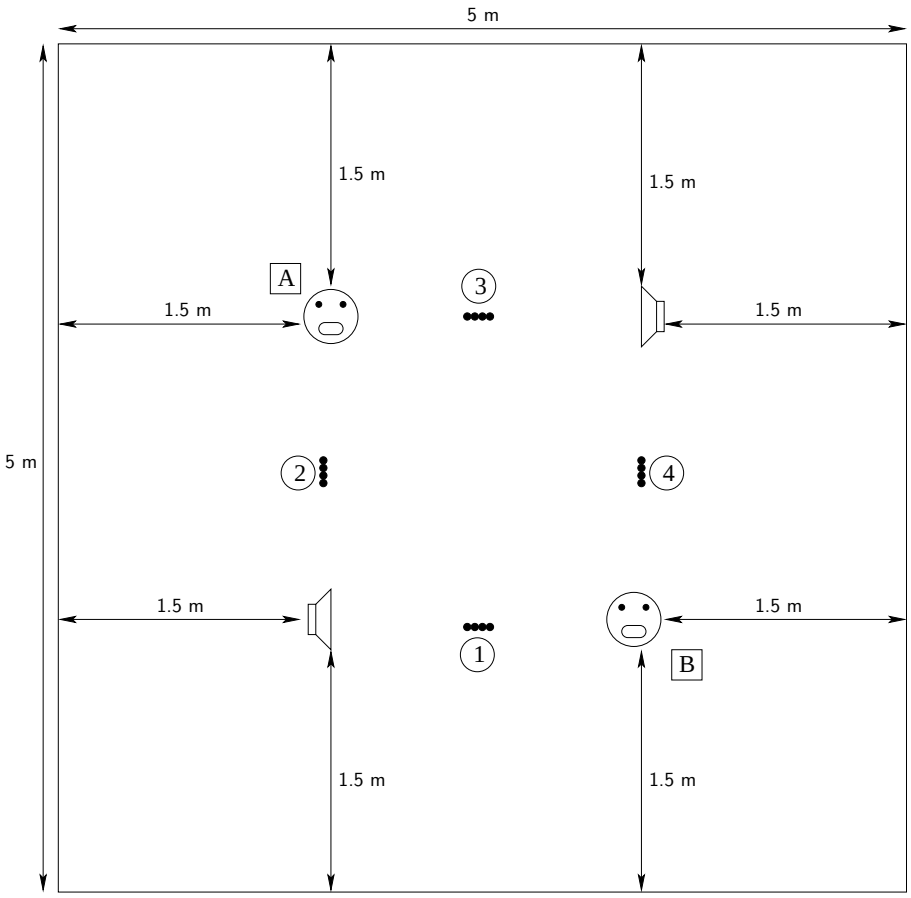


Figure 6.2: The acoustic scenario used in the simulations. There are two speech sources, two babble noise sources, and 4 nodes with 4 microphones each.

suppressing speaker A), so each node will estimate an undistorted (unmixed) version of both speakers.

Since the microphone signals consist of convolutive mixtures of multiple signals, we transform the problem to the frequency domain to satisfy the instantaneous mixture model (6.1). To this aim, we use an STFT with a DFT size equal to $L = 512$ if not specified otherwise. The LC-DANSE algorithm is then applied to each frequency bin separately. This decouples the problem into L smaller problems that approximately[9] satisfy the data model (6.1). Notice that L is representative for the length of the time domain filters that are implicitly applied to the microphone signals.

It is noted that speech signals are not stationary, whereas the convergence of the LC-DANSE algorithm is based on stationarity. However, by taking long-term time averages, we mostly incorporate spatial characteristics, which are indeed invariant in time.

It should be noted that we do not intend to provide a fully practical speech enhancement implementation here. The goal of this experiment is to validate the convergence and optimality of the LC-DANSE algorithm in a realistic scenario with convolutive mixtures. Therefore, to isolate the effects of estimation errors, all experiments are performed in batch mode where the correlation matrices are computed by time averaging over the full signal, and both matrices $\mathbf{Q}_k^d$ and $\mathbf{Q}_k^n$ (and their local LC-DANSE versions) are computed as the eigenvector corresponding to the dominant eigenvalue of the clean speech covariance matrices of both speakers. The latter isolates errors introduced by any practical estimation approach (such as, e.g., [13] or techniques based on [14]). The covariance matrices (for each frequency bin) are computed in the STFT domain, by means of time averaging.

6.6.3 Performance Measures

We use three performance measures to assess the quality of the LC-DANSE based noise reduction algorithm, namely the broadband signal-to-noise ratio (SNR), the signal-to-distortion ratio (SDR), and the mean squared error (MSE) between the coefficients of the optimal (centralized) LCMV filters $\hat{\mathbf{w}}_k$ and the filters (6.19) obtained by the LC-DANSE algorithm (after transformation to the time-domain). In particular, the SNR and SDR at node k in iteration i are defined as

$$\mathrm{SNR}^i = 10 \log_{10} \frac{E\left\{d_k^i[t]^2\right\}}{E\left\{n_k^i[t]^2\right\}} \tag{6.45}$$

[9] The STFT transform always introduces some leakage between frequency bins, and therefore the data model (6.1) is only approximately satisfied. The larger the choice for L, the better (6.1) holds.

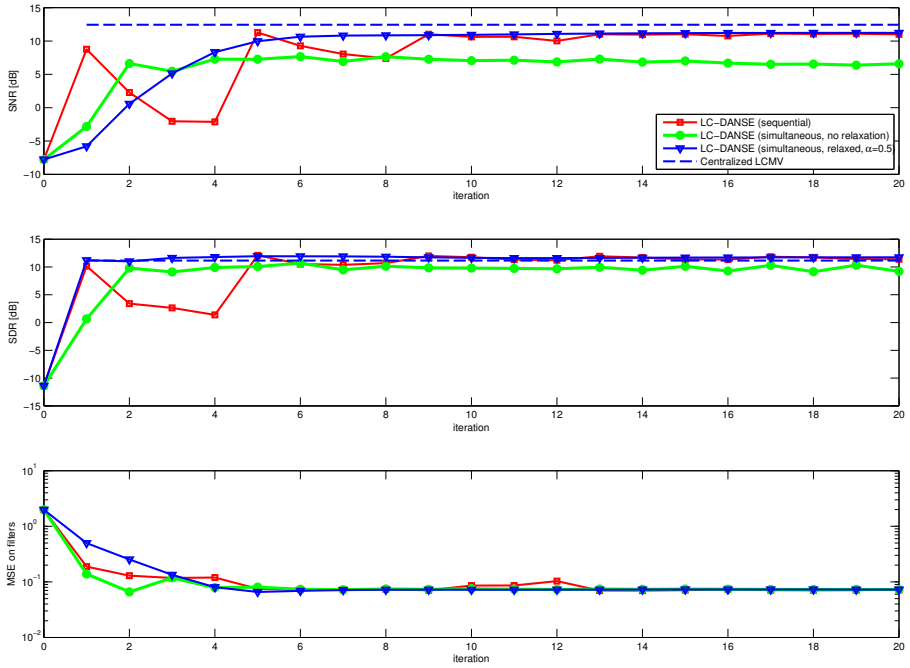


Figure 6.3: The SNR, SDR and MSE at the filter output of node 1 versus the iteration index of the LC-DANSE algorithm, applied to the scenario of Fig. 6.2. The DFT size is $L = 512$.

$$\mathrm{SDR}^i = 10\log_{10}\frac{E\left\{x_{k1}[t]^2\right\}}{E\left\{\left(x_{k1}[t]-d_k^i[t]\right)^2\right\}} \tag{6.46}$$

with $n_k^i[t]$ and $d_k^i[t]$ denoting the time domain noise component and desired speech component respectively at the beamformer output at node k in iteration i, and $x_{k1}[t]$ denoting the desired time domain clean speech component as observed by the reference microphone of node k. The noise component $n_k^i[t]$ also contains the residual interfering speech component. The MSE at node k is defined as

$$\mathrm{MSE}^i = \|\hat{\mathbf{w}}_k - \mathbf{w}_k^i\|^2 \tag{6.47}$$

with $\hat{\mathbf{w}}_k$ defined by (6.6), and $\mathbf{w}_k^i$ denoting the first column of $\mathbf{W}_k$ in (6.19) at iteration i. The filters in (6.47) are transformed to the time-domain.

6.6.4 Results

Fig. 6.3 shows the convergence of the LC-DANSE algorithm at node 1, where a DFT size of $L = 512$ is used. If nodes update sequentially, the algorithm gets

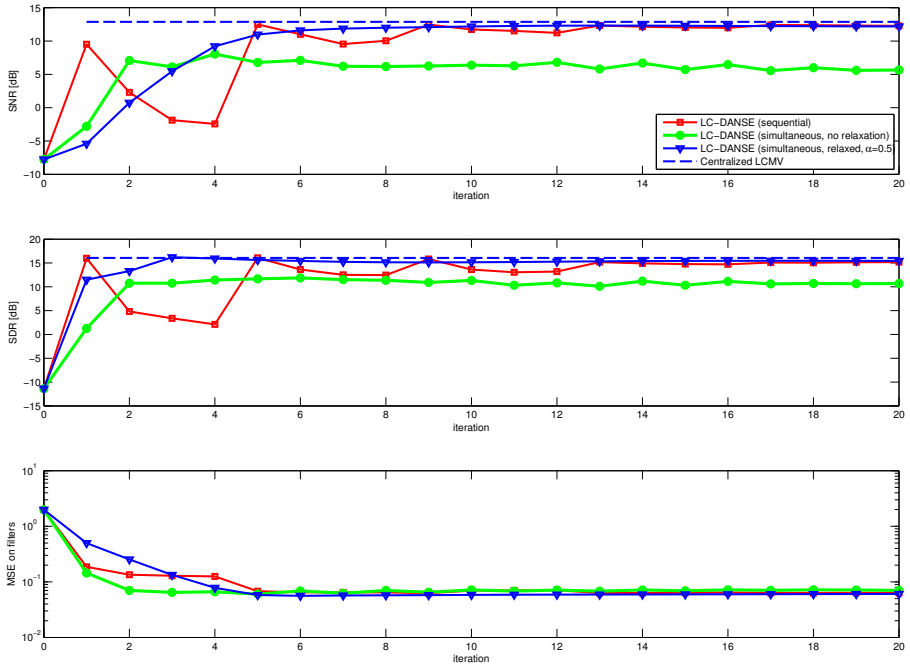


Figure 6.4: The SNR, SDR and MSE at the filter output of node 1 versus the iteration index of the LC-DANSE algorithm, applied to the scenario of Fig. 6.2. The DFT size is $L = 1024$.

close to the optimal performance of the centralized LCMV beamformer for the beamforming problem at node 1. However, since the data model (6.1) is only approximately satisfied due to the finite DFT size, the optimal performance is not reached. If nodes update simultaneously, the algorithm does not converge, but gets stuck in a limit cycle instead[10], yielding a loss of approximately 5 dB in SNR and 2 dB in SDR. This is similar to the behavior of the simultaneous DANSE (S-DANSE) algorithm in [11] and [4]. By applying relaxation (with $\alpha^i = 0.5$, $\forall\, i \in \mathbb{N}$), the algorithm with simultaneous node-updating converges to the same solution as with sequential node-updating.

Fig. 6.4 shows the results when a DFT size of $L = 1024$ is used. It is observed that the SNR curve better approaches the SNR curve of the centralized algorithm, compared to the case where the DFT size was $L = 512$. This is not surprising: the larger the DFT size, the better the data model (6.1) is satisfied, and hence the closer LC-DANSE approaches the centralized solution. It is noted that simultaneous node-updating reduces the SNR with more than 6

[10]The limit cycle is not clearly visible in Fig. 6.3. However, it is slightly visible in the SDR curve during the last 5 iterations.

dB, and the SDR with more than 4 dB in this experiment. Again, relaxation yields a converging algorithm, yielding the optimal LCMV beamformers.

6.7 Conclusions

In this paper, we have revisited the linearly constrained distributed adaptive node-specific signal estimation algorithm, referred to as LC-DANSE. The algorithm changes the local constraints of the nodes in every iteration and significantly compresses the sensor signal observations that are shared between nodes. Nevertheless it obtains the optimal node-specific LCMV beamformers, corresponding to the original constraints, as if all sensor signals are available to each node. The LC-DANSE algorithm is closely related to the DANSE algorithm for unconstrained linear MMSE signal estimation, and we have pointed out that previously developed extensions for DANSE also hold for LC-DANSE. Convergence and optimality of the algorithm has been formally proven. We have experimentally verified that convergence can be obtained when nodes update simultaneously, if relaxation is applied, similar to the unconstrained DANSE algorithm. We have provided simulation results that demonstrate the effectiveness of the algorithm for speech enhancement (and separation) in a wireless acoustic sensor network.

Appendix

6.A Proof of Lemma 6.3:

To simplify notation, we will often omit the superscript i in the sequel, unless we explicitly want to denote a specific iteration number. Similar to (6.10), the solution of (6.27), s.t. (6.31) is given by

$$\mathbf{U}_k = \mathbf{R}_{k,q}^{-1}\widetilde{\mathbf{Q}}_k \left(\widetilde{\mathbf{Q}}_k^H \mathbf{R}_{k,q}^{-1}\widetilde{\mathbf{Q}}_k\right)^{-1} \widetilde{\mathbf{F}}_k \tag{6.48}$$

$$= \mathbf{R}_{k,q}^{-1}\overline{\mathbf{Q}}_k \left(\overline{\mathbf{Q}}_k^H \mathbf{R}_{k,q}^{-1}\overline{\mathbf{Q}}_k\right)^{-1} \mathbf{F}_k \ . \tag{6.49}$$

with

$$\mathbf{R}_{k,q} = \mathcal{W}_{k,q}\mathbf{R}_{yy}\mathcal{W}_{k,q}^H \ . \tag{6.50}$$

Since the columns of every $\mathbf{Q}_k$ in (6.25), $\forall k \in \mathcal{J}$, span the same K-dimensional subspace as the columns of $\mathbf{H}$, the following holds for any $k, m \in \mathcal{J}$:

$$\exists\, \mathbf{B}_{km} \in \left(\mathbb{C}^{K\times K}\right)^{-1} :\ \mathbf{Q}_k = \mathbf{Q}_m\mathbf{B}_{km} \tag{6.51}$$

where $\left(\mathbb{C}^{K\times K}\right)^{-1}$ denotes the set of all non-singular $K\times K$ matrices. We define

$$\mathcal{A}_{km,q} = \text{blockdiag}\left\{\mathbf{A}_{km}^H,\ldots,\mathbf{A}_{km}^H,\mathbf{I}_{M_q},\mathbf{A}_{km}^H,\ldots,\mathbf{A}_{km}^H\right\} \tag{6.52}$$

where $\mathbf{I}_{M_q}$ is the q-th argument of the blockdiag{.} operator, and since we assume that (6.33) holds,

$$\mathcal{W}_{k,q} = \mathcal{A}_{km,q}\mathcal{W}_{m,q}\ . \tag{6.53}$$

Combining (6.50) with (6.53), we obtain

$$\mathbf{R}_{k,q} = \mathcal{A}_{km,q}\mathbf{R}_{m,q}\mathcal{A}_{km,q}^H\ . \tag{6.54}$$

Combining (6.51), (6.53) and (6.29), we obtain

$$\overline{\mathbf{Q}}_k = \mathcal{A}_{km,q}\overline{\mathbf{Q}}_m\mathbf{B}_{km}\ . \tag{6.55}$$

By inserting (6.54) and (6.55) in (6.49), and using the fact that all $\mathbf{F}_k$, $\forall k\in\mathcal{J}$, are full rank, some straightforward algebraic manipulation shows that

$$\mathbf{U}_k = \mathcal{A}_{km,q}^{-H}\mathbf{U}_m\mathbf{L}_{km} \tag{6.56}$$

with

$$\mathbf{L}_{km} = \mathbf{F}_m^{-1}\mathbf{B}_{km}^{-H}\mathbf{F}_k\ . \tag{6.57}$$

Since $\mathbf{W}_k^{i+1} = \mathcal{W}_{k,q}^{i\,H}\mathbf{U}_k$, it follows from (6.53) and (6.56) that

$$\mathbf{W}_k^{i+1} = \mathbf{W}_m^{i+1}\mathbf{L}_{km}\ . \tag{6.58}$$

By setting $\mathbf{A}_{km}^{i+1} = \mathbf{L}_{km}$, we obtain (6.34).

Bibliography

[1] 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, pp. 5277–5291, 2010.

[2] H. Ochiai, P. Mitran, H. Poor, and V. Tarokh, "Collaborative beamforming for distributed wireless ad hoc sensor networks," *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4110 – 4124, 2005.

[3] A. Bertrand and M. Moonen, "Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology," *IEEE Transactions on Signal Processing*, vol. 59, pp. 2196–2210, May 2011.

[4] ——, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009.

[5] M. F. A. Ahmed and S. A. Vorobyov, "Collaborative beamforming for wireless sensor networks with gaussian distributed sensor nodes," *IEEE Trans. Wireless. Comm.*, vol. 8, pp. 638–643, February 2009.

[6] A. Bertrand, J. Callebaut, and M. Moonen, "Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks," in *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC), Tel Aviv, Israel*, Aug. 2010.

[7] 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, Sept. 2002.

[8] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, pp. 38–51, Jan. 2009.

[9] S. Markovich Golan, S. Gannot, and I. Cohen, "A reduced bandwidth binaural MVDR beamformer," in *Proc. of the International Workshop on Acoustic Echo and Noise Control (IWAENC)*, Tel-Aviv, Israel, Aug. 2010.

[10] M. Souden, J. Benesty, and S. Affes, "On optimal frequency-domain multichannel linear filtering for noise reduction," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 2, pp. 260 –276, 2010.

[11] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part II: simultaneous & asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5292–5306, 2010.

[12] B. Van Veen and K. Buckley, "Beamforming: a versatile approach to spatial filtering," *ASSP Magazine, IEEE*, vol. 5, no. 2, pp. 4 –24, apr. 1988.

[13] S. Markovich Golan, S. Gannot, and I. Cohen, "Subspace tracking of multiple sources and its application to speakers extraction," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas USA, Mar. 2010, pp. 201 –204.

[14] A. Bertrand and M. Moonen, "Energy-based multi-speaker voice activity detection with an ad-hoc microphone array," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas USA, March 2010, pp. 85–88.

[15] ——, "Distributed LCMV beamforming in wireless sensor networks with node-specific desired signals," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011.

[16] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," vol. 65, pp. 943–950, Apr. 1979.

[17] M. Nilsson, S. Soli, and A. Sullivan, "Development of the Hearing in Noise Test for the measurement of speech reception thresholds in quiet and in noise," *Journal of the Acoustical Society of America*, vol. 95, pp. 1085–1099, Feb. 1994.

Part III

Distributed Parameter Estimation Techniques

Chapter 7

Distributed Total Least Squares

Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks

Alexander Bertrand and Marc Moonen

Published in *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2320 - 2330, May 2011.

Contributions of first author

- literature study
- co-derivation and co-analysis of the D-TLS algorithm
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

Total least squares (TLS) is a popular solution technique for overdetermined systems of linear equations, where both the right hand side and the input data matrix are assumed to be noisy. We consider a TLS problem in an ad hoc wireless sensor network, where each node collects observations that yield a node-specific subset of linear equations. The goal is to compute the TLS solution of the full set of equations in a distributed fashion, without gathering all these equations in a fusion center. To facilitate the use of the dual based subgradient algorithm (DBSA), we transform the TLS problem to an equivalent convex semidefinite program (SDP), based on semidefinite relaxation (SDR). This allows us to derive a distributed TLS (D-TLS) algorithm, that satisfies the conditions for convergence of the DBSA, and obtains the same solution as the original (unrelaxed) TLS problem. Even though we make a detour through SDR and SDP theory, the resulting D-TLS algorithm relies on solving local TLS-like problems at each node, rather than computationally expensive SDP optimization techniques. The algorithm is flexible and fully distributed, i.e. it does not make any assumptions on the network topology and nodes only share data with their neighbors through local broadcasts. Due to the flexibility and the uniformity of the network, there is no single point of failure, which makes the algorithm robust to sensor failures. Monte-Carlo simulation results are provided to demonstrate the effectiveness of the method.

7.1 Introduction

An ad hoc wireless sensor network (WSN) [1] consists of spatially distributed sensor nodes that share data with each other through wireless links to perform a certain task, e.g. the estimation of a signal or parameter. Generally, the goal is to implement an estimator that is equal (or close) to an optimal estimator that has access to all observations acquired by all the nodes in the network. The latter corresponds to a centralized approach, where all observations are gathered in a fusion center to calculate an optimal estimate. A WSN, on the other hand, requires distributed algorithms that allow for in-network processing and only rely on local collaboration between neighboring nodes. It is likely that future data acquisition, control and physical monitoring, will heavily rely on this type of networks.

Distributed estimation in a linear least squares (LLS) framework has been widely studied in the sensor network literature (see e.g. [2–10]). The LLS framework is applied for linear regression problems, and provides a solution for an overdetermined system of linear equations, i.e. $\mathbf{U}\mathbf{w} = \mathbf{d}$ with $\mathbf{U}$ an $M \times P$ data matrix and $\mathbf{d}$ an M-dimensional data vector with $M \geq P$. The aim is then to find a vector $\mathbf{w}$ that minimizes the squared error between the left-hand

and the right-hand side, i.e.

$$\mathbf{w}_{LLS} = \arg\min_{\mathbf{w}} \|\mathbf{U}\mathbf{w} - \mathbf{d}\|_2^2 \tag{7.1}$$

where $\|.\|_2$ denotes the Euclidean norm (L^2-norm). In a WSN, nodes can either have access to subsets of the columns of $\mathbf{U}$ [2–4], e.g. for distributed signal enhancement and beamforming [11], or to subsets of the rows of $\mathbf{U}$ (and $\mathbf{d}$) [5–10], e.g. for distributed system identification. Despite this common cost function, both problems are tackled in very different ways. In the former case, each node estimates a node-specific subvector of the global estimator $\mathbf{w}$, whereas in the latter case, each node estimates the full estimator $\mathbf{w}$, which allows for consensus-based approaches. Here, we focus on the latter case, i.e. where each node has access to a node-specific subset of the equations. Distributed LLS estimation amounts to computing the network-wide LLS solution in a distributed way, where the nodes have to reach a consensus on a common network-wide parameter vector $\mathbf{w}$. A motivation for this type of distributed estimation problems, different types of algorithms, and possible applications, can be found in [5–10] and references therein.

In a stochastic framework, the solution (7.1) yields the best linear unbiased estimate (BLUE), if $\mathbf{d}$ is corrupted by additive zero-mean white noise [12]. However, in many practical situations, the input data matrix $\mathbf{U}$ is also noisy. For example, in adaptive filtering, this corresponds to the case where there is additive noise at both the input and output of the filter that is estimated. In [13], it is shown that the resulting LLS estimate (7.1) is biased in this case. A natural generalization of LLS estimation is total least squares (TLS) estimation, where both $\mathbf{U}$ and $\mathbf{d}$ are indeed assumed to be noisy (cf. [14] for an extensive overview). For the particular case of recursive TLS in adaptive filtering theory, we refer to [13], where it is shown that TLS yields an unbiased estimate if the additive noise at both the input and output of the filter is zero-mean and white.

In this paper, we propose an algorithm to compute the network-wide TLS solution in a distributed way, where the nodes of a WSN have to reach a consensus on a common network-wide parameter vector $\mathbf{w}$. As opposed to the LLS problem, the TLS problem is non-convex, which makes the derivation of a distributed algorithm non-trivial. To facilitate the use of the dual based subgradient algorithm (DBSA) [15], we transform the TLS problem to an equivalent convex semidefinite program (SDP), based on a technique called semidefinite relaxation (SDR) [16]. This allows us to derive a consensus-based distributed TLS (D-TLS) algorithm, that satisfies the conditions for convergence of the DBSA. Even though we make a detour through SDR and SDP theory, the resulting D-TLS algorithm does not rely on computationally expensive SDP optimization techniques. Instead, we obtain an iterative algorithm that solves local TLS-like problems at each node, which enables the use of robust TLS solvers. Furthermore, even though we solve a relaxed optimization problem, the D-TLS algorithm still obtains the solution of the original TLS problem,

which is available at each node (after convergence).

The D-TLS algorithm is flexible and fully distributed, i.e. nodes only share data with their neighbors through local broadcasts (single-hop communication) and the algorithm does not assume a specific network topology or a so-called Hamiltonian cycle (as often assumed in incremental strategies, e.g. [5]), which makes it robust to sensor and link failures. Due to the flexibility and the uniformity of the network (i.e. all nodes perform identical tasks), there is no single point of failure and the network is self-healing.

The outline of the paper is as follows. In section 7.2, we briefly review the TLS problem statement and we explain how the TLS solution can be computed. Furthermore, we describe the TLS problem in a distributed context for WSNs. In section 7.3, we review the dual based subgradient algorithm, which forms the backbone of the D-TLS algorithm. In section 7.4, we derive the D-TLS algorithm, based on a semi-definite relaxation of the original TLS problem, and we address its convergence properties. In section 7.5, we provide simulation results. Conclusions are drawn in section 7.6.

7.2 Problem Statement

7.2.1 The Total Least Squares Problem (TLS)

We consider an overdetermined system of linear equations in P unknowns (stacked in a vector $\mathbf{w}$), given by

$$\mathbf{U}\mathbf{w} = \mathbf{d} \tag{7.2}$$

with $\mathbf{U}$ an $M \times P$ noisy data matrix and $\mathbf{d}$ an M-dimensional noisy data vector, with $M \geq P$. Since this is an overdetermined system of equations, its solution set is usually empty. The goal is then to find the TLS solution, i.e. to solve the constrained optimization problem

$$\min_{\mathbf{w}, \triangle\mathbf{U}, \triangle\mathbf{d}} \|\triangle\mathbf{U}\|_F^2 + \|\triangle\mathbf{d}\|_2^2 \tag{7.3}$$

$$\text{s.t. } (\mathbf{U} + \triangle\mathbf{U})\,\mathbf{w} = (\mathbf{d} + \triangle\mathbf{d}) \tag{7.4}$$

where $\|.\|_F$ denotes the Frobenius norm. In section 7.5.1, we will demonstrate that the TLS estimate can be significantly better than the LLS estimate in cases where the input data matrix $\mathbf{U}$ is noisy. In [13], it is shown that the TLS estimate provides an unbiased estimate when both $\mathbf{U}$ and $\mathbf{d}$ are contaminated with white noise (whereas the LLS solution is biased in this case). In order to compute the solution of (7.3)-(7.4), we define the matrix

$$\mathbf{U}_+ = [\mathbf{U}\ \mathbf{d}] \tag{7.5}$$

and the matrix

$$\mathbf{R} = \mathbf{U}_+^T \mathbf{U}_+ \, . \tag{7.6}$$

Let $\mathbf{x}$ denote the eigenvector of $\mathbf{R}$ corresponding to the smallest eigenvalue, where $\mathbf{x}$ is scaled such that

$$\mathbf{x} = \begin{bmatrix} \mathbf{w}^* \\ -1 \end{bmatrix} \, . \tag{7.7}$$

In [17], it is shown that there exists a unique solution for the TLS problem (7.3)-(7.4) if the $(P+1)$-th singular value of $\mathbf{U}_+$ is strictly smaller than the P-th singular value (where the singular values are sorted in decreasing order). In this case, the first P entries of $\mathbf{x}$, denoted by $\mathbf{w}^*$, are the solution of the TLS problem (7.3)-(7.4) [14, 17]. The eigenvector $\mathbf{x}$ can be computed in a robust and efficient way (e.g. by means of an eigenvalue or singular value decomposition [17]).

7.2.2 Total Least Squares in Ad Hoc Wireless Sensor Networks

Consider an ad hoc WSN with the set of nodes $\mathcal{J} = \{1, \ldots, J\}$ and with a random (connected) topology, where neighboring nodes can exchange data through a wireless link. We denote the set of neighbor nodes of node k as $\mathcal{N}_k$, i.e. all the nodes that can share data with node k, node k excluded. Node k collects observations of a data matrix $\mathbf{U}_k$ and a data vector $\mathbf{d}_k$. The goal is then to solve a network-wide TLS problem, i.e. to compute a vector $\mathbf{w}$ from

$$\min_{\mathbf{w}, \triangle\mathbf{U}_1, \ldots, \triangle\mathbf{U}_J, \triangle\mathbf{d}_1, \ldots, \triangle\mathbf{d}_J} \sum_{k \in \mathcal{J}} \left(\|\triangle\mathbf{U}_k\|_F + \|\triangle\mathbf{d}_k\|_2 \right) \tag{7.8}$$

$$\text{s.t.} \;\; (\mathbf{U}_k + \triangle\mathbf{U}_k)\, \mathbf{w} = (\mathbf{d}_k + \triangle\mathbf{d}_k) \, , \; k = 1, \ldots, J \, . \tag{7.9}$$

We will refer to this problem as the distributed total least squares (D-TLS) problem, since it corresponds to the TLS problem (7.3)-(7.4), where each node has access to a subset of the rows of $\mathbf{U}$ and a subvector of $\mathbf{d}$. The goal is to compute $\mathbf{w}$ in a distributed fashion, without gathering all data in a fusion center.

We will develop a consensus-based algorithm, based on dual decomposition of the optimization problem (7.8)-(7.9), which we will refer to as the distributed TLS algorithm (D-TLS). The dual based subgradient algorithm (DBSA) [15], which forms the backbone of the D-TLS algorithm, is described in its general form in section 7.3. Even though many of the required conditions for DBSA are violated in the case of D-TLS, we will explain in section 7.4 how DBSA can still be applied, without affecting its convergence results.

Remark I: In dynamic scenarios, an adaptive algorithm is required to track changes in the environment. In this case, extra observations become available

over time at each node, and old observations may become invalid. This can be easily included in the problem statement described above and in the D-TLS algorithm described in section 7.4, by applying recursive updating and downdating schemes. However, for the sake of an easy exposition, we assume in the sequel that the matrices $\mathbf{U}_k$ do not change over time.

Remark II: The different nodes usually observe data at a different signal-to-noise ratio (SNR), depending on their position (e.g. the distance to a localized source). Therefore, it is often desirable to give a different weight to each term in (7.8), or even to each specific row of $\mathbf{U}_k$, depending on the local SNR. This is often referred to as generalized total least squares (GTLS) [14]. However, since the algorithmic treatment of GTLS and TLS are the same, and for the sake of an easy exposition, we do not consider GTLS in this paper.

7.3 Dual Based Subgradient Algorithm (DBSA)

In this section, we briefly review DBSA, as applied in [18] on the following optimization problem[1] for a connected network with a set of nodes $\mathcal{J}$:

$$\min_{\mathbf{x}} \sum_{k\in\mathcal{J}} f_k(\mathbf{x}) \tag{7.10}$$

$$\text{s.t. } \mathbf{x} \in \mathcal{X} \tag{7.11}$$

where $\mathcal{X}$ is a convex set with non-empty interior, $\mathbf{x}$ is an N- dimensional vector, and where f_k, $\forall\, k \in \mathcal{J}$, are convex functions. In this problem, it is assumed that f_k can only be evaluated at node k. The goal is then to solve (7.10)-(7.11) in a distributed fashion, where each node eventually obtains an optimal vector $\hat{\mathbf{x}}$ from the solution set of (7.10)-(7.11).

One possible way to solve the above problem is to apply dual composition. The main idea is to introduce local copies of $\mathbf{x}$ in each node, and to optimize and update these variables until a consensus is reached amongst all neighbor nodes as follows:

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_J} \sum_{k\in\mathcal{J}} f_k(\mathbf{x}_k) \tag{7.12}$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X},\ \forall\, k \in \mathcal{J} \tag{7.13}$$

$$\mathbf{x}_k = \mathbf{x}_q,\ \forall\, k \in \mathcal{J},\ \forall\, q \in \mathcal{N}_k,\ q < k \tag{7.14}$$

[1] The problem described in [18] incorporates private variables, i.e. variables that are not common between the nodes, and assumes that the consensus variable is 1-dimensional. For later purpose, we extend the problem here for N-dimensional consensus variables. For the sake of an easy exposition, we omit the private variables.

which has the same solution as (7.10)-(7.11), if the network is connected. The constraints (7.14) are denoted as consensus constraints. The condition $q < k$ removes redundancy in the constraints, such that there is a single consensus constraint for each individual link[2]. We denote the set of network links $\mathcal{L} = \{1, \ldots, L\}$, where each link contains a pair of nodes in $\mathcal{J}$. We introduce the $NL \times NJ$ linking matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \ldots & \mathbf{C}_{1J} \\ \vdots & & \vdots \\ \mathbf{C}_{L1} & \ldots & \mathbf{C}_{LJ} \end{bmatrix} \tag{7.15}$$

where the $N \times N$ submatrix $\mathbf{C}_{lk}$ is defined as

$$\mathbf{C}_{lk} = \begin{cases} \mathbf{I}_N, & \text{if } l = \{k, q\} \text{ and } k < q \\ -\mathbf{I}_N, & \text{if } l = \{k, q\} \text{ and } q < k \\ \mathbf{O}_N, & \text{otherwise} \end{cases} \tag{7.16}$$

with $\mathbf{I}_N$ denoting the $N \times N$ identity matrix and $\mathbf{O}_N$ denoting an all-zero $N \times N$ matrix. With this, we can rewrite problem (7.12)-(7.14) as

$$\min_{\mathbf{x}_1, \ldots, \mathbf{x}_J} \sum_{k \in \mathcal{J}} f_k(\mathbf{x}_k) \tag{7.17}$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X}, \, \forall \, k \in \mathcal{J} \tag{7.18}$$

$$\mathbf{C}\overline{\mathbf{x}} = \mathbf{0} \tag{7.19}$$

where $\overline{\mathbf{x}} = [\mathbf{x}_1^T \ \ldots \ \mathbf{x}_J^T]^T$. This problem is not separable due to the consensus constraints (7.19). However, by applying dual decomposition [15, 19], the problem can be solved iteratively in a fully distributed fashion. Consider the dual function

$$d(\boldsymbol{\lambda}) = \min_{\overline{\mathbf{x}}} L(\boldsymbol{\lambda}, \overline{\mathbf{x}}) \tag{7.20}$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X}, \, \forall \, k \in \mathcal{J} \tag{7.21}$$

where $L(\boldsymbol{\lambda}, \overline{\mathbf{x}})$ denotes the so called Lagrangian defined by

$$L(\boldsymbol{\lambda}, \overline{\mathbf{x}}) = \sum_{k \in \mathcal{J}} f_k(\mathbf{x}_k) + \boldsymbol{\lambda}^T \mathbf{C}\overline{\mathbf{x}} \tag{7.22}$$

and where the variables in the (NL)-dimensional vector $\boldsymbol{\lambda}$ are so called Lagrange multipliers. We denote $\boldsymbol{\lambda}_l$ as the subvector of $\boldsymbol{\lambda}$ that is applied to the rows of $\mathbf{C}$ corresponding to submatrices $\mathbf{C}_{lk}$, $\forall \, k \in \mathcal{J}$, i.e. the Lagrange multipliers corresponding to the consensus constraints at link l. If strong duality

[2]Note that there is still a lot of redundancy left, since many links can be removed without harming the overall consensus between the nodes.

holds [19], then the optimal value of (7.12)-(7.14) is the same as the optimal value of the dual problem

$$\max_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}) \; . \tag{7.23}$$

Here, the unconstrained optimization problem (7.23) is referred to as the 'master problem', and the optimization problem in (7.20)-(7.21) is referred to as the 'slave problem'. An important observation is the fact that the slave problem is fully separable:

$$d(\boldsymbol{\lambda}) = \sum_{k \in \mathcal{J}} d_k(\boldsymbol{\lambda}) \tag{7.24}$$

with

$$d_k(\boldsymbol{\lambda}) = \min_{\mathbf{x}_k} f_k(\mathbf{x}_k) + \sum_{l \in \mathcal{L}_k} \boldsymbol{\lambda}_l^T \mathbf{C}_{lk} \mathbf{x}_k \tag{7.25}$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X} \tag{7.26}$$

where $\mathcal{L}_k$ denotes the set of links that contain node k. Since the dual function $d(\boldsymbol{\lambda})$ is not differentiable in general, the master problem is solved by a subgradient method. A subgradient of $d(\boldsymbol{\lambda})$ in the point $\boldsymbol{\lambda}$ is given by

$$\mathbf{g}(\boldsymbol{\lambda}) = \mathbf{C}\overline{\mathbf{x}}^*(\boldsymbol{\lambda}) \tag{7.27}$$

where $\overline{\mathbf{x}}^*(\boldsymbol{\lambda})$ denotes an $\overline{\mathbf{x}}$ that is in the solution set of the slave problem (7.20)-(7.21) for the given $\boldsymbol{\lambda}$ [15]. It is noted that, if $d(\boldsymbol{\lambda})$ is differentiable in $\boldsymbol{\lambda}$, the subgradient $\mathbf{g}(\boldsymbol{\lambda})$ is equal to the actual gradient $\nabla d(\boldsymbol{\lambda})$.

We can now solve (7.12)-(7.14) with the following algorithm, which is known as the dual based subgradient algorithm (DBSA):

1. Let $i = 1$ and $\boldsymbol{\lambda}^0 = \mathbf{0}_N$, where $\mathbf{0}_N$ denotes an all-zero N-dimensional vector.
2. Each node $k \in \mathcal{J}$ solves (7.25)-(7.26) to get $\mathbf{x}_k^i$.
3. Each node $k \in \mathcal{J}$ transmits $\mathbf{x}_k^i$ to the nodes in $\mathcal{N}_k$.
4. Each node $k \in \mathcal{J}$ updates the subvectors $\boldsymbol{\lambda}_l^i$, $\forall\, l \in \mathcal{L}_k$, according to

$$\boldsymbol{\lambda}_l^{i+1} = \boldsymbol{\lambda}_l^i + \mu \left(\mathbf{C}_{lk} \mathbf{x}_k^i + \mathbf{C}_{lq} \mathbf{x}_q^i \right) \tag{7.28}$$

   where q is the node that is connected to node k by link l, and with stepsize $\mu > 0$.
5. $i \leftarrow i + 1$.
6. return to step 2.

In [15], it is shown that, if the stepsize μ is sufficiently small, the distance of the $\mathbf{x}_k^i$'s to the optimal solution set is reduced in each iteration if the following conditions are satisfied:

1. The functions f_k are convex, $\forall\, k \in \mathcal{J}$.

2. The set $\mathcal{X}$ is convex with non-empty interior.
3. Strong duality holds for (7.17)-(7.19).
4. All subgradients of $d(\boldsymbol{\lambda})$ are bounded for all values of $\boldsymbol{\lambda}$, i.e. $\exists\, B \in \mathbb{R}, \forall\, \boldsymbol{\lambda}:\ \|\mathbf{g}(\boldsymbol{\lambda})\|_2 \leq B$.

More specifically, the following theorems hold under the above assumptions [15]:

Theorem 7.1 *If $\boldsymbol{\lambda}^i$ is not optimal, then for every dual optimal solution $\boldsymbol{\lambda}^*$, we have*

$$\|\boldsymbol{\lambda}^{i+1} - \boldsymbol{\lambda}^*\| < \|\boldsymbol{\lambda}^i - \boldsymbol{\lambda}^*\| \tag{7.29}$$

for all stepsizes μ such that

$$0 < \mu < \frac{2\left(d(\boldsymbol{\lambda}^*) - d(\boldsymbol{\lambda}^i)\right)}{\|\mathbf{g}(\boldsymbol{\lambda}^i)\|_2^2}\,. \tag{7.30}$$

Theorem 7.2 *For a fixed stepsize μ, there is at least asymptotic convergence to a neighborhood of $d(\boldsymbol{\lambda}^*)$, i.e.*

$$\limsup_{i\to\infty} d(\boldsymbol{\lambda}^i) \geq d(\boldsymbol{\lambda}^*) - \frac{\mu B^2}{2} \tag{7.31}$$

where B is defined in condition 4.

In other words, there is at least asymptotic convergence to a neighborhood of the optimal solution, where the size of this neighborhood shrinks with μ.

7.4 Distributed Total Least Squares (D-TLS)

In this section, we demonstrate how the DBSA algorithm can be applied to the D-TLS problem (7.8)-(7.9), even though the latter is a non-convex optimization problem. As explained in section 7.2.1, the solution of the TLS problem (7.3)-(7.4) can be found by means of an eigenvalue or a singular value decomposition, i.e. the computation of the eigenvector of $\mathbf{R}$ corresponding to the smallest eigenvalue. We use $\mathbf{x}$ to denote this eigenvector, and we use N to denote the dimension of this vector, i.e. $N = P + 1$. The solution of the TLS problem is then given by the first P entries of $\mathbf{x}$ when scaled such that the last (the $(P+1)$-th) entry is -1.

The eigenvector $\mathbf{x}$ corresponding to the smallest eigenvector of $\mathbf{R}$ is the solution of the following quadratically constrained quadratic program[3] (QCQP)

$$\min_{\mathbf{x}} \mathbf{x}^T\mathbf{R}\mathbf{x} \tag{7.32}$$

[3]The stationary points of the Lagrangian of (7.32)-(7.33) are the eigenvectors of $\mathbf{R}$. Therefore, the eigenvector corresponding to the smallest eigenvalue is the global minimum of (7.32)-(7.33).

$$\text{s.t. } \|\mathbf{x}\|_2^2 = 1 \,. \tag{7.33}$$

In the case of D-TLS, this becomes

$$\min_{\mathbf{x}} \sum_{k\in\mathcal{J}} \mathbf{x}^T \mathbf{R}_k \mathbf{x} \tag{7.34}$$

$$\text{s.t. } \|\mathbf{x}\|_2^2 = 1 \tag{7.35}$$

where $\mathbf{R}_k = \mathbf{U}_{k+}^T \mathbf{U}_{k+}$, with $\mathbf{U}_{k+} = [\mathbf{U}_k \; \mathbf{d}_k]$.

7.4.1 Transformation into a Convex Problem

It is noted that (7.34)-(7.35) has the same form as (7.10)-(7.11) to which the DBSA was applied in section 7.3. However, we cannot straightforwardly apply the DBSA to (7.34)-(7.35) due to the norm constraint, which defines a non-convex constraint set with empty interior. This violates condition 2 for application of the DBSA, which states that the set $\mathcal{X}$ must be convex with non-empty interior. Furthermore, condition 3 assumes strong duality after addition of the consensus constraints, which is not guaranteed for (7.34)-(7.35), again due to the non-convex norm constraint. However, since (7.34)-(7.35) is a QCQP, we can apply semidefinite relaxation (SDR) to transform the original problem into a convex optimization problem, which has the solution of the original QCQP in its solution set (see [16] for an overview article). Since the new problem is convex, DBSA can be readily applied, and its convergence results will then also hold for the derived algorithm. Remarkably, it will turn out that the SDR yields an algorithm that solves local TLS problems at each node[4] (see section 7.4.2), which enables the use of robust TLS solvers. Furthermore, even though we solve a relaxed problem, we eventually obtain the solution of the original D-TLS problem (7.34)-(7.35).

SDR is based on the observation that $\mathbf{x}^T\mathbf{R}_k\mathbf{x} = \text{tr}(\mathbf{x}^T\mathbf{R}_k\mathbf{x}) = \text{tr}(\mathbf{R}_k\mathbf{x}\mathbf{x}^T)$, where $\text{tr}(\mathbf{Q})$ denotes the trace of the matrix $\mathbf{Q}$. By applying the substitution $\mathbf{X} = \mathbf{x}\mathbf{x}^T$, we transform (7.34)-(7.35) into the equivalent problem

$$\min_{\mathbf{X}} \sum_{k\in\mathcal{J}} \text{tr}\left(\mathbf{R}_k\mathbf{X}\right) \tag{7.36}$$

$$\text{s.t. } \text{tr}\left(\mathbf{X}\right) = 1 \tag{7.37}$$

$$\mathbf{X} \succeq 0 \tag{7.38}$$

$$\text{rank}(\mathbf{X}) = 1 \tag{7.39}$$

[4] If DBSA would be directly applied to the original QCQP (7.34)-(7.35), this would result in a different distributed algorithm that is not guaranteed to converge, and it cannot rely on robust TLS solvers, since each local cost function will then have linear terms due to the consensus constraints.

where $\mathbf{X} \succeq 0$ is used to denote that $\mathbf{X}$ is symmetric and positive (semi)definite. It is noted that the rank constraint (7.39) is the only non-convex constraint. The SDR approach relaxes (7.36)-(7.39) to a convex optimization problem by removing this rank constraint, i.e.

$$\min_{\mathbf{X}} \sum_{k \in \mathcal{J}} \operatorname{tr}(\mathbf{R}_k \mathbf{X}) \tag{7.40}$$

$$\text{s.t. } \operatorname{tr}(\mathbf{X}) = 1 \tag{7.41}$$

$$\mathbf{X} \succeq 0 \,. \tag{7.42}$$

This type of problem is known as a semidefinite program (SDP), which is studied extensively in literature [19, 20]. Obviously, the SDP resulting from the SDR usually yields a new problem which is not equivalent to the original QCQP. However, there is a known upper bound on the rank of the matrix $\mathbf{X}^*$ with lowest rank that is in the solution set of any feasible SDP with an $N \times N$ matrix variable and m linear constraints [16]:

$$\operatorname{rank}(\mathbf{X}^*) \leq \frac{\sqrt{8m+1}-1}{2} \,. \tag{7.43}$$

Furthermore, this matrix $\mathbf{X}^*$ can easily be found [21]. It is noted that if $m = 1$, as it is the case for (7.40)-(7.42), the rank of $\mathbf{X}^*$ reduces to 1. This is a very important observation, since it implies that the optimal solution of (7.36)-(7.39), and consequently of the D-TLS problem (7.34)-(7.35), can be found by solving the relaxed (convex) problem (7.40)-(7.42).

We can now apply the DBSA to the problem (7.40)-(7.42). Condition 1 for convergence of the DBSA is straightforwardly satisfied. Condition 4 is also satisfied due to the boundedness of the constraint set. To satisfy condition 2, the constraint set needs to be convex with a non-empty interior, but the latter is not satisfied in (7.40)-(7.42). However, if we would replace the constraint (7.41) with $1 + \epsilon \geq \operatorname{tr}(\mathbf{X}) \geq 1$ where $\epsilon > 0$, the solution set of (7.40)-(7.42) remains the same (due to the minimization), but the constraint set has now a non-empty interior (and it remains bounded and convex). Therefore, condition 2 is also satisfied[5]. Condition 3 is satisfied due to the following theorem[6] from [22]:

Theorem 7.3 *If the solution set of an SDP is non-empty and bounded, then strong duality holds.*

[5]The resulting DTLS algorithm as described in subsection 7.4.2 will be exactly the same, whether we use the constraint (7.41), or the constraint $1 + \epsilon \geq \operatorname{tr}(\mathbf{X}) \geq 1$, even though only the latter satisfies condition 2. Therefore, we keep the constraint (7.40) in the sequel without affecting the convergence results of DBSA.

[6]It is noted that strong duality still holds if (7.41) is replaced with $1 + \epsilon \geq \operatorname{tr}(\mathbf{X}) \geq 1$, due to convexity of the problem, and the fact that Slater's condition holds.

This theorem holds for the SDP (7.40)-(7.42) (also after including the consensus constraints), because there is at least one solution (corresponding to the solution of (7.34)-(7.35)) and boundedness is guaranteed due to the fact that the constraint set is bounded.

7.4.2 The Distributed Total Least Squares Algorithm

In the previous section, we have explained how the D-TLS problem can be transformed into an SDP which has the same solution, and satisfies the conditions for convergence of the DBSA. In this section, we derive the formulas that describe the D-TLS algorithm at each node, based on the algorithm description of the DBSA in section 7.3.

The SDP (7.40)-(7.42) yields the following node-specific DBSA slave problems (to be compared to the general formulation (7.25)-(7.26)):

$$d_k(\mathbf{\Lambda}) = \min_{\mathbf{X}_k} \operatorname{tr}(\mathbf{R}_k \mathbf{X}_k) + \sum_{l \in \mathcal{L}_k} c_{lk} \operatorname{tr}\left(\mathbf{\Lambda}_l^T \mathbf{X}_k\right) \tag{7.44}$$

$$\text{s.t. } \operatorname{tr}(\mathbf{X}_k) = 1 \tag{7.45}$$

$$\mathbf{X}_k \succeq 0 \tag{7.46}$$

where c_{lk} is defined as the scalar version of (7.16), i.e. $c_{lk} = \frac{1}{N}\operatorname{tr}(\mathbf{C}_{lk})$. The $\mathbf{\Lambda}_l$'s are $N \times N$ matrices that contain the Lagrange multipliers corresponding to the consensus constraints $\mathbf{X}_k = \mathbf{X}_q, \forall\, k \in \mathcal{J}, \forall\, q \in \mathcal{N}_k$.

The slave problem (7.44)-(7.46) is solved by each node $k \in \mathcal{J}$ (step 2 in the DBSA algorithm), followed by an exchange of these solutions between neighboring nodes. After receiving its neighbors' solutions, the Lagrange multipliers at node k are updated as follows (step 4 in the DBSA algorithm):

$$\forall\, l \in \mathcal{L}_k : \ \mathbf{\Lambda}_l^{i+1} = \mathbf{\Lambda}_l^i + \mu\left(c_{lk}\mathbf{X}_k^i + c_{lq}\mathbf{X}_q^i\right) \tag{7.47}$$

where q is the node that is connected to node k by link l. It is noted that the nodes communicate $N \times N$ matrices $\mathbf{X}_k^i$, which is not very efficient. Eventually, we are interested in the N-dimensional vector $\mathbf{x}$ that solves the original QCQP (7.34)-(7.35) corresponding to the distributed TLS problem (7.8)-(7.9). However, based on SDR theory, the problem (7.44)-(7.46) must have a rank-1 solution since it is the SDR of the QCQP[7]

$$d_k(\mathbf{\Lambda}) = \min_{\mathbf{x}_k} \mathbf{x}_k^T \left(\mathbf{R}_k + \sum_{l \in \mathcal{L}_k} c_{lk}\mathbf{\Lambda}_l^i\right) \mathbf{x}_k \tag{7.48}$$

[7]Note that, since the $\mathbf{X}_k^i$'s are solutions of an SDP, they are symmetric. Therefore the matrices $\mathbf{\Lambda}_l^i$ are also symmetric due to (7.47), assuming that they are initialized with a symmetric matrix.

$$\text{s.t. } \|\mathbf{x}_k\|_2^2 = 1 \tag{7.49}$$

where we used the substitution $\mathbf{X}_k = \mathbf{x}_k\mathbf{x}_k^T$. We can then solve this QCQP instead of (7.44)-(7.46), to obtain the rank-1 solution. Conveniently, the Lagrange multipliers now appear in the quadratic term of the cost function, rather than in a separate linear term as it is the case in (7.25). Therefore, the above constrained optimization problem again corresponds to computing the eigenvector corresponding to the smallest eigenvalue[8] of a matrix, i.e. $\left(\mathbf{R}_k + \sum_{l\in\mathcal{L}_k} c_{lk}\mathbf{\Lambda}_l^i\right)$. This also implies that

$$d_k\left(\mathbf{\Lambda}\right) = \lambda_{\min}\left(\mathbf{R}_k + \sum_{l\in\mathcal{L}_k} c_{lk}\mathbf{\Lambda}_l^i\right) \tag{7.50}$$

where $\lambda_{\min}\left(\mathbf{Q}\right)$ denotes the smallest eigenvalue of $\mathbf{Q}$. The nodes now only share N-dimensional vectors with their neighbors. This yields the following algorithm, which we refer to as the distributed total least squares (D-TLS) algorithm:

The D-TLS Algorithm

1. Let $i = 1$ and $\mathbf{\Lambda}_l^0 = \mathbf{O}_{N\times N}$, $\forall\, l \in \mathcal{L}$.
2. Each node $k \in \mathcal{J}$ computes the eigenvector $\mathbf{x}^i$ corresponding to the smallest eigenvalue of $\overline{\mathbf{R}}_k^i$ defined by

$$\overline{\mathbf{R}}_k^i = \mathbf{R}_k + \sum_{l\in\mathcal{L}_k} c_{lk}\mathbf{\Lambda}_l^i \tag{7.51}$$

   where $\mathbf{x}_k^i$ is scaled such that $\|\mathbf{x}_k^i\|_2 = 1$.
3. Each node $k \in \mathcal{J}$ transmits $\mathbf{x}_k^i$ to the nodes in $\mathcal{N}_k$.
4. Each node $k \in \mathcal{J}$ updates the Lagrange multipliers $\mathbf{\Lambda}_l^i$, $\forall\, l \in \mathcal{L}_k$ according to

$$\mathbf{\Lambda}_l^{i+1} = \mathbf{\Lambda}_l^i + \mu\left(c_{lk}\mathbf{x}_k^i\mathbf{x}_k^{i\,T} + c_{lq}\mathbf{x}_q^i\mathbf{x}_q^{i\,T}\right) \tag{7.52}$$

   where q is the node that is connected to node k by link l, and with stepsize $\mu > 0$.
5. $i \leftarrow i + 1$.
6. return to step 2.

In each iteration of the D-TLS algorithm, each node computes the eigenvector corresponding to the smallest eigenvalue of a local symmetric matrix. Hence,

[8] It is noted that this eigenvalue can be negative.

although we took a detour through SDR and SDP theory, the resulting D-TLS algorithm eventually can still rely on robust TLS computations, i.e. an eigenvalue decomposition, and there is no need for computationally expensive iterative interior-point algorithms, as mostly used to solve SDP's. Furthermore, if $N \gg |\mathcal{N}_k|$, the $\mathbf{x}_k^{i+1}$ can be efficiently computed from the previous solution $\mathbf{x}_k^i$ by exploiting the rank-1 updates of the $\mathbf{\Lambda}_l^i$'s [23, 24]. Similar procedures can be used to update (or downdate) the $\mathbf{R}_k$ matrices when new observations are included or when old observations are removed (e.g. in adaptive scenarios).

Remark I: The multiplier update (7.52) requires that each node has a unique index to determine the c_{lk}'s, which requires some extra coordination. Furthermore, each node stores a separate $N \times N$ multiplier matrix for each different neighbor. We can eliminate this need by using the substitution

$$\mathbf{\Theta}_k^i = \sum_{l \in \mathcal{L}_k} c_{lk} \mathbf{\Lambda}_l^i \tag{7.53}$$

which corresponds to a node-specific variable for node k. Since $c_{lk}^2 = 1$ and $c_{lk}c_{lq} = -1$ for link l that connects nodes k and q, we readily find from (7.52) and (7.53) that

$$\mathbf{\Theta}_k^{i+1} = \mathbf{\Theta}_k^i + \mu \left(|\mathcal{N}_k| \mathbf{x}_k^i \mathbf{x}_k^{i\,T} - \sum_{q \in \mathcal{N}_k} \mathbf{x}_q^i \mathbf{x}_q^{i\,T} \right) \tag{7.54}$$

where we use $|\mathcal{S}|$ to denote the cardinality of the set $\mathcal{S}$. It is noted that each node now only has to store a single multiplier matrix (instead of multiple $\mathbf{\Lambda}_l$'s, i.e. one for each link). This yields a simplified version of the D-TLS algorithm where the multiplier update (7.52) is replaced with (7.54) where $\mathbf{\Theta}_k^0 = \mathbf{O}_{N \times N}$, $\forall\, k \in \mathcal{J}$, and where $\overline{\mathbf{R}}_k^i$ is redefined as $\overline{\mathbf{R}}_k^i = \mathbf{R}_k + \mathbf{\Theta}_k^i$.

Remark II: It is noted that D-TLS is an adaptive algorithm where each node performs a similar task. This uniformity and adaptivity guarantees that there is no single point of failure. This means that the algorithm is self-healing in case of permanent link or sensor failures, assuming that the network graph remains connected. If link l between nodes k and q is permanently removed, both nodes need to remove $\mathbf{\Lambda}_l$ from (7.51), and the network will still converge to the solution of (7.34)-(7.35). If node k fails permanently, its neighbors must remove the $\mathbf{\Lambda}_l$ from (7.51), $\forall\, l \in \mathcal{L} : k \in l$. The network will then adapt to find the new solution, i.e. the solution of (7.34)-(7.35) with the k-th term removed from (7.34). It is noted that the simplified version of the D-TLS algorithm (see Remark I) does not have this self-healing property, since the $\mathbf{\Lambda}_l$'s are not stored separately[9]. However, both versions of the algorithm can handle temporary link failures, as long as none of the links or nodes are permanently removed. We will demonstrate in section 7.5.7 that the algorithm still performs

[9]Note that the sum of all $\mathbf{\Theta}_k$'s must be zero at all time.

well in random graphs, where the links between nodes fail in each iteration with a certain probability $p < 1$.

Remark III: We have transformed the original QCQP (7.34)-(7.35) to another problem (7.44)-(7.46) that has the original solution in its solution set and for which strong duality holds. However, this does not necessarily imply that strong duality also holds for (7.34)-(7.35). Nevertheless, strong duality of the relaxed problem (7.44)-(7.46) is sufficient for convergence of the derived D-TLS algorithm to the optimal solution of (7.44)-(7.46). This is because the D-TLS algorithm is essentially DBSA applied[10] to (7.44)-(7.46), and not to (7.34)-(7.35). However, since we select the rank-1 solution out of the solution set of (7.44)-(7.46), we eventually obtain the solution of the original D-TLS problem (7.34)-(7.35) or (7.8)-(7.9). It is noted that, when DBSA is applied to the original QCQP (7.34)-(7.35), this would result in a different algorithm, which will probably not converge due to a non-zero duality gap. Furthermore, this algorithm cannot rely on TLS solvers, since linear terms will appear in the local cost functions due to the linear consensus constraints.

7.4.3 Convergence

The convergence of the D-TLS algorithm follows straightforwardly from the convergence of the DBSA. For a fixed stepsize μ, we know from Theorem 7.2 that there is at least asymptotic convergence to a neighborhood of the optimal solution, where the size of this neighborhood shrinks with μ. Furthermore, from Theorem 7.1, we know that each new iteration of D-TLS gets closer to the optimal solution if μ satisfies (7.30). In particular, for the case of D-TLS, we find that (to be compared with (7.30)):

$$d(\boldsymbol{\lambda}^*) = \lambda_{\min}\left(\sum_{k\in\mathcal{J}} \mathbf{R}_k\right) \tag{7.55}$$

$$d(\boldsymbol{\lambda}^i) = \sum_{k\in\mathcal{J}} \lambda_{\min}\left(\overline{\mathbf{R}}_k^i\right) \tag{7.56}$$

$$\|\mathbf{g}(\boldsymbol{\lambda}^i)\|_2^2 = \sum_{k\in\mathcal{J}} \sum_{q\in\mathcal{N}_k, q<k} \|\mathbf{X}_k^i - \mathbf{X}_q^i\|_F^2 \tag{7.57}$$

where $\mathbf{X}_k^i = \mathbf{x}_k^i \mathbf{x}_k^{i\,T}$. Here, (7.55) follows from strong duality, and (7.56) follows from (7.50) and (7.51). Expression (7.57) follows from (7.27) and the fact that D-TLS implicitly solves an SDP with substitution $\mathbf{X}_k^i = \mathbf{x}_k^i \mathbf{x}_k^{i\,T}$. Using the fact that $\mathbf{X}_k^i = \mathbf{x}_k^i \mathbf{x}_k^{i\,T}$ and $\|\mathbf{x}_k^i\|_2 = 1$, $\forall\, k \in \mathcal{J}$, we obtain

$$\|\mathbf{X}_k^i - \mathbf{X}_q^i\|_F^2 = \mathrm{tr}\left((\mathbf{X}_k^i - \mathbf{X}_q^i)^2\right)$$

[10] Even though D-TLS eventually solves a QP at each node, the solution of this QP is also a solution of (7.44)-(7.46) (i.e. the rank-1 solution), hence we implicitly solve (7.44)-(7.46).

$$
\begin{aligned}
&= \mathrm{tr}\left(\mathbf{X}_k^{i\,2}\right) + \mathrm{tr}\left(\mathbf{X}_q^{i\,2}\right) - 2\mathrm{tr}\left(\mathbf{X}_k^i \mathbf{X}_q^i\right) \\
&= \|\mathbf{x}_k^i\|_2^4 + \|\mathbf{x}_q^i\|_2^4 - 2(\mathbf{x}_k^{i\,T}\mathbf{x}_q^i)^2 \\
&= 2 - 2(\mathbf{x}_k^{i\,T}\mathbf{x}_q^i)^2 \,. \qquad (7.58)
\end{aligned}
$$

Based on Theorem 7.1, this results in the following bound for the stepsize:

$$
0 < \mu^i < \frac{\lambda_{\min}\left(\sum_{k\in\mathcal{J}} \mathbf{R}_k\right) - \sum_{k\in\mathcal{J}} \lambda_{\min}\left(\overline{\mathbf{R}}_k^i\right)}{|\mathcal{L}| - \sum_{k\in\mathcal{J}} \sum_{q\in\mathcal{N}_k, q<k} (\mathbf{x}_k^{i\,T}\mathbf{x}_q^i)^2} \,. \qquad (7.59)
$$

Here, the numerator is equal to the difference between the current value of the dual function and its optimal value, which reduces to zero if and only if all $\mathbf{x}_k$ are equal to the solution of (7.34)-(7.35). The denominator (7.57) is the squared consensus error summed over all the links of the network. It is noted that the denominator heavily depends on the number of links $|\mathcal{L}|$, i.e. strongly connected networks require a smaller μ. However, this does not necessarily result in slower convergence, since information diffuses much faster over the network if it is strongly connected (simulations in Section 7.5.3 confirm this). The numerator, on the other hand, mainly depends on the number of nodes, i.e. $\lambda_{\min}\left(\sum_{k\in\mathcal{J}} \mathbf{R}_k\right)$ increases when $|\mathcal{J}|$ increases (this follows from the fact that all $\mathbf{R}_k$'s are positive definite).

7.4.4 Choice of Stepsize μ

A small stepsize μ yields a more accurate estimate of the TLS solution, but generally yields slow convergence. Therefore, it is desirable to adapt μ in each iteration so that it is close to the upper bound (7.59). This is often not possible in practice since the nodes usually do not have access to most of the variables in (7.59). However, the second term of both the numerator and the denominator are summations over variables that are locally available in different nodes of the network (in fact, the computation of $|\mathcal{L}|$ can also be viewed as the sum of locally available variables, i.e. $|\mathcal{L}| = \frac{1}{2}\sum_{k\in\mathcal{J}} |\mathcal{N}_k|$). Therefore, these summations can be iteratively computed by so called 'consensus averaging' algorithms (see e.g. [25]). These procedures compute the average (or sum) of local observations in an iterative fashion, and this average is then eventually known to each node. If N is large, this will not significantly increase the required communication bandwidth that is used by D-TLS. Alternative fast gossip type algorithms for computing the sum of local quantities can be found in [26]. The value $\lambda_{\min}\left(\sum_{k\in\mathcal{J}} \mathbf{R}_k\right)$ cannot be computed, and therefore an estimate should be known a priori (or at least a tight lower bound should be known).

It is noted that the computation of (7.59) becomes less robust when the algorithm closely approximates the optimal solution, since both the numerator and the denominator then become very small. In this case, there will be large fluctuations in the stepsizes of subsequent iterations, and extremely large values may be obtained when the denominator approaches zero. Therefore, it

is better to use (7.59) in combination with an a priori fixed upper bound, to avoid instability. An alternative approach is to only use an upper bound for the denominator instead of the full expression, i.e.

$$0 < \mu^i < \frac{\lambda_{\min}\left(\sum_{k\in\mathcal{J}} \mathbf{R}_k\right) - \sum_{k\in\mathcal{J}} \lambda_{\min}\left(\overline{\mathbf{R}}_k^i\right)}{|\mathcal{L}|} \tag{7.60}$$

or the less conservative bound

$$0 < \mu^i < \frac{\lambda_{\min}\left(\sum_{k\in\mathcal{J}} \mathbf{R}_k\right) - \sum_{k\in\mathcal{J}} \lambda_{\min}\left(\overline{\mathbf{R}}_k^i\right)}{\epsilon + |\mathcal{L}| - \sum_{k\in\mathcal{J}} \sum_{q\in\mathcal{N}_k, q<k} (\mathbf{x}_k^{i\,T} \mathbf{x}_q^i)^2} \tag{7.61}$$

with $\epsilon > 0$.

If there is no knowledge available on any of the variables in (7.59), convergence to the optimal solution can be guaranteed when a variable stepsize is used that satisfies [15]:

$$\sum_{i=1}^{\infty} \mu^i = \infty \tag{7.62}$$

$$\sum_{i=1}^{\infty} (\mu^i)^2 < \infty \,. \tag{7.63}$$

A possible (but conservative) choice is $\mu^i = \frac{1}{i}$. However, in tracking applications, (7.62)-(7.63) cannot be used, and then a fixed stepsize is a better alternative. The latter requires some parameter tuning, and it introduces a trade-off between the speed of convergence (or adaptation speed) and the accuracy of the final solution, as given by Theorem 7.2.

7.5 Simulations

In this section, we provide numerical simulation results that demonstrate the convergence properties of the D-TLS algorithm. To illustrate the general behavior, we show results that are averaged over multiple Monte-Carlo (MC) runs. In each MC run, the following process is used to generate the network and the sensor observations (unless stated otherwise):

1. Construct a P-dimensional vector $\mathbf{w}$ where the entries are drawn from a zero-mean normal distribution with unit variance.
2. Create a random[11] connected network with J nodes, with 3 neighbors per node (on average).

[11] Unless stated otherwise, we start from a random tree to guarantee that the network is connected. Links are then randomly added until the average number of links per node equals 3.

3. For each node $k \in \mathcal{J}$:
   - Construct a $2P \times P$ input data matrix $\mathbf{U}_k$ where the entries are drawn from a zero-mean normal distribution with unit variance. The matrix $\mathbf{U}_k$ is then scaled by a random factor drawn from a uniform distribution on the unit interval (this models the different observation SNR at each node).
   - Compute $\mathbf{d}_k = \mathbf{U}_k\mathbf{w}$.
   - Add zero-mean white Gaussian noise to the entries in $\mathbf{U}_k$ and $\mathbf{d}_k$, with a standard deviation of 0.5.

In each experiment, we choose $J = 20$, $N = P+1 = 10$, and we run the D-TLS algorithm for 400 iterations (unless stated otherwise). It is noted that both algorithms (i.e. D-TLS and its simplified version, as described in Remark I in section 7.4.2) are exactly equivalent in each of the experiments in the sequel, except for the experiment in subsection 7.5.7.

To assess the convergence and optimality of the algorithm, we use the error between the centralized TLS solution and the local estimate, averaged over the J nodes in the network:

$$\frac{1}{|\mathcal{J}|}\sum_{k\in\mathcal{J}} \|\mathbf{x}_k - \mathbf{x}_{\text{TLS}}\|_2 \tag{7.64}$$

where $\mathbf{x}_{\text{TLS}}$ is the solution of (7.34)-(7.35).

7.5.1 TLS versus LLS

To motivate the use of D-TLS, we first compare different techniques to estimate $\mathbf{w}$. The results are given in Fig. 7.1, showing the exact entries of the 9-dimensional vector $\mathbf{w}$, together with the following estimates:

- The centralized TLS solution.
- The D-TLS solution at node 1 (with fixed stepsize $\mu = 1$).
- The local TLS solution at node 1, without sharing any data with neighboring nodes.
- The centralized LLS solution.

Fig. 7.1 demonstrates that the TLS procedure indeed provides a significantly better estimate than the LLS procedure, since the latter ignores the fact that the input data matrix is noisy. Furthermore, it is demonstrated that the solution of the D-TLS algorithm is very close the centralized TLS solution. A last observation is that nodes indeed benefit from sharing their data amongst each other, since only using the data of node 1 yields a very poor TLS estimate.

7.5.2 Influence of Stepsize μ

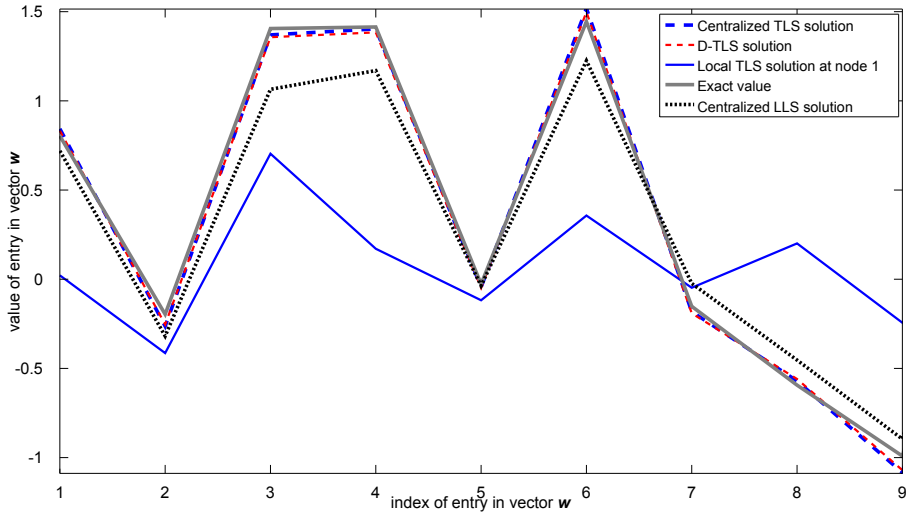


Figure 7.1: Comparison of different techniques for the estimation of $\mathbf{w}$.

For the convergence properties of the D-TLS algorithm, we compare the following strategies to choose the stepsize μ:

- Strategy 1: a fixed stepsize $\mu = 1$.
- Strategy 2: an adaptive stepsize, based on upper bound (7.59), but clipped to always stay smaller than 5.
- Strategy 3: a more conservative adaptive stepsize, based on upper bound (7.60).

For each strategy, 1000 MC runs are performed, and the mean values (over all MC runs) of the error curves are shown in Fig. 7.2. The gray-colored areas cover one standard deviation of the error curves over all MC runs (on both sides of the mean curve). It is observed that the stepsize based on (7.60) (strategy 3) has a fast initial convergence, but becomes extremely slow when it gets close to the optimal solution. This can be explained by the fact that the denominator is fixed in (7.60), and becomes very large compared to the numerator when reaching the optimal solution, yielding an extremely conservative stepsize μ. The fixed stepsize and the adaptive stepsize based on (7.59) (strategies 1 and 2), have a similar convergence speed.

In the next experiment, we use different values for the fixed stepsize. The results are shown in Fig. 7.3 (the standard deviation of the MC runs is given for $\mu = 1$ and $\mu = 2.5$). Fig. 7.3 shows that $\mu = 1$ was actually a lucky guess. Indeed, the convergence of the D-TLS algorithm heavily depends on the stepsize. The stepsizes that are smaller than 1 all yield slower convergence. For the stepsizes larger than 1 ($\mu = 2.5$ and $\mu = 5$), convergence becomes a vague

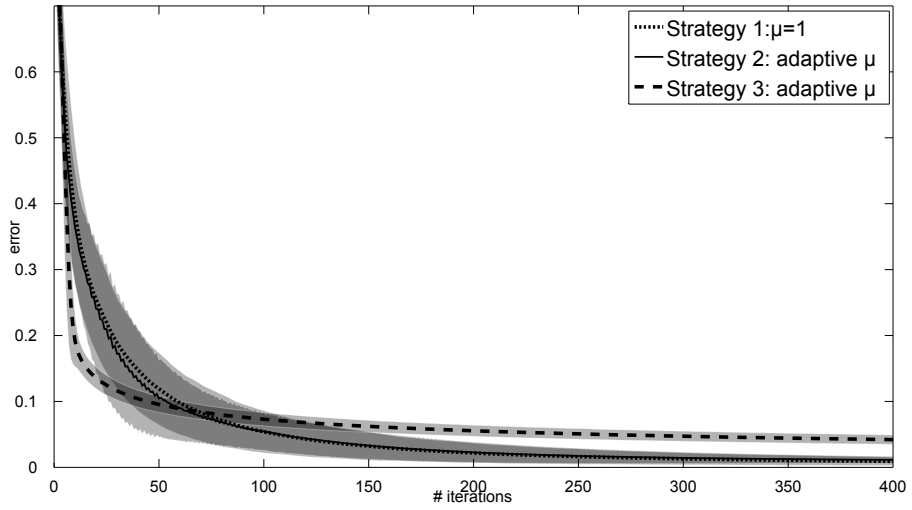


Figure 7.2: Convergence properties of D-TLS for different strategies in choosing μ, averaged over 1000 Monte-Carlo runs.

concept, due to the large excess error, i.e. the $\mathbf{x}_k^i$'s vary significantly over the different iterations (this causes the large standard deviation in the experiments with $\mu = 2.5$). The adaptive stepsize based on (7.59) seems to provide a good convergence speed when prior tuning of the stepsize is not possible, as it is almost identical to the $\mu = 1$ experiment (compare with Fig. 7.2).

7.5.3 Influence of Connectivity of the Network Graph

In this experiment, we investigate the influence of the connectivity of the network, where the number of nodes is fixed to $J = 20$. Two extremes are of interest: a network with a ring topology (i.e. each node has 2 neighbors) and a fully connected network (i.e. each node has $J - 1$ neighbors). We also investigate networks in between those extremes, by adding links between random node pairs. In particular, we simulated (random) networks with $|\mathcal{L}| \in \{20, 25, \ldots, 40, 50, \ldots, 100, 190\}$.

200 MC runs are performed for each type of network (the links are chosen differently in each run). The stepsize is fixed, but depends on the number of links, i.e. $\mu = \frac{10}{|\mathcal{L}|}$, inspired by expression (7.60), as the denominator increases linearly with the number of links. The results are shown in Fig. 7.4. Not surprisingly, it is observed that increasing the number of links increases the convergence speed, even though a smaller μ is used. However, this effect becomes less significant for large $|\mathcal{L}|$. In the case of a ring topology, the convergence speed can be greatly improved by only adding a few extra links.

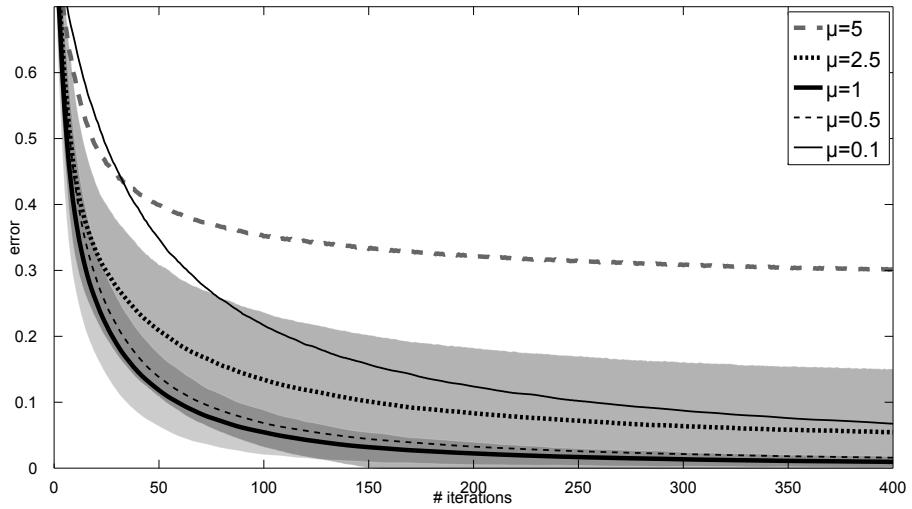


Figure 7.3: Convergence properties of D-TLS for different values of the fixed stepsize μ, averaged over 1000 Monte-Carlo runs.

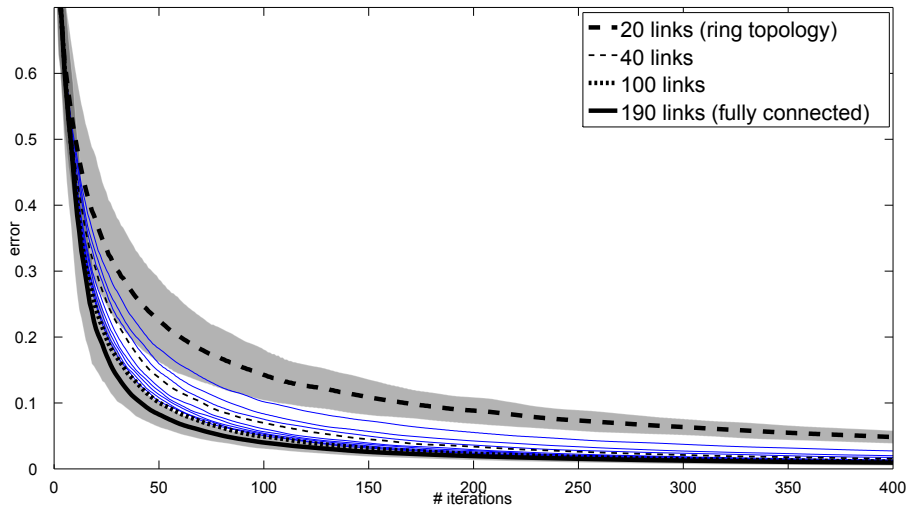


Figure 7.4: Convergence properties of D-TLS for different degrees of connectivity, averaged over 200 Monte-Carlo runs.

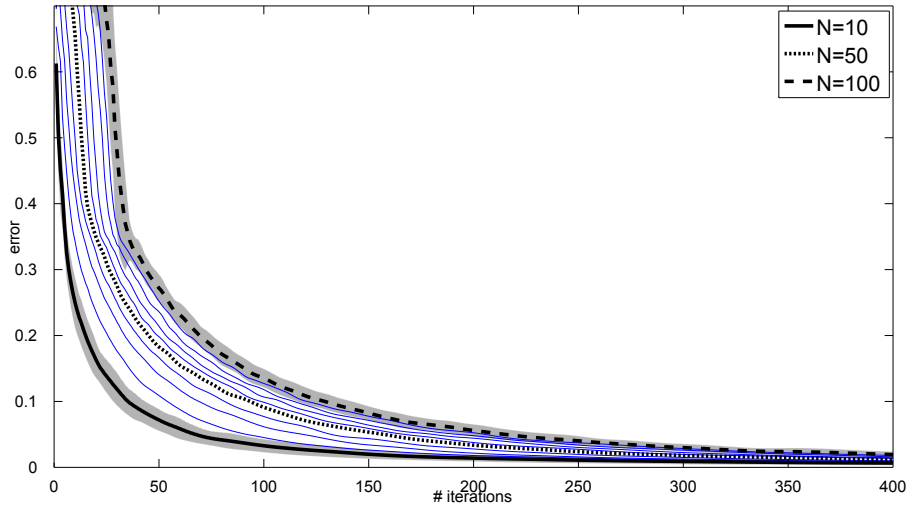


Figure 7.5: Convergence properties of D-TLS for different dimensions N, averaged over 100 Monte-Carlo runs.

7.5.4 Influence of Dimension N

In this experiment, we increase the dimension N of the vector $\mathbf{x}$. When N is large, the ratio between the smallest and second smallest eigenvalue of the matrix $\mathbf{R}_k$ might be close to one, especially at nodes with very low SNR. This makes the eigenvalue problem in step 2 of the D-TLS algorithm ill-conditioned, which may affect the stability or the convergence time of the D-TLS algorithm[12]. Therefore, the input data matrix $\mathbf{U}$ is not scaled with a random variable in this experiment, to avoid nodes with very low SNR.

In particular, we simulated 100 MC runs for each value of $N \in \{10, 20, \ldots, 100\}$. The stepsize is fixed, but depends on the dimension N, i.e. $\mu = \frac{N}{10}$, inspired by expression (7.60), as the numerator increases linearly with N. The results[13] are shown in Fig. 7.5. It is observed that the value of N significantly influences the convergence speed of the algorithm.

7.5.5 Influence of Size of the Network

In this experiment, we increase the number of nodes J, but the average links per node remains fixed to 3. We simulated 100 MC runs for a network with

[12] In practice, low SNR nodes should therefore be removed from the network when using D-TLS for high-dimensional regression problems.

[13] The reason why the variance over the different MC runs is smaller than in previous experiments, is the fact that the SNR is equal in every node and in every MC run.

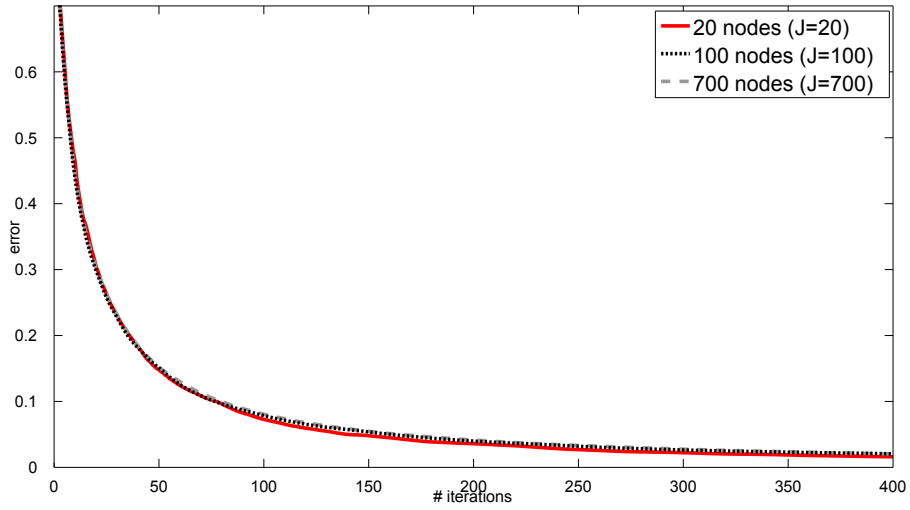


Figure 7.6: Convergence properties of D-TLS for different sizes of the network, averaged over 100 Monte-Carlo runs.

$J \in \{20, 100, 700\}$ nodes, and with the stepsize fixed to $\mu = 1$. The result is shown in Fig. 7.6. It is observed that the size of the network has almost no influence on the convergence speed.

7.5.6 Random Graphs

In this experiment, each link can fail in each iteration, with a certain probability p. This models packet loss in the communication between nodes. Basically, this means that $\mathcal{N}_k$ in (7.54) changes with the iteration index i. We simulated 200 MC runs for each value of $p \in \{0, 0.1, 0.2, \ldots, 0.9\}$. The results are shown in Fig. 7.7 for a fixed stepsize $\mu = 1$. It is observed that the algorithm still performs pretty well under significant packet loss. However, high packet loss significantly decreases convergence speed, especially when close to the optimal solution.

7.5.7 Self-Healing Property

In this experiment, we demonstrate the self-healing property of the D-TLS algorithm, i.e. it's capability to adapt to permanent changes in the network topology. Notice that this is different from the previous experiment with random graphs, where each link is active in an infinite number of iterations. Here, we demonstrate that the algorithm can recover the optimal TLS solution when nodes are permanently removed from the network. It is noted that the simpli-

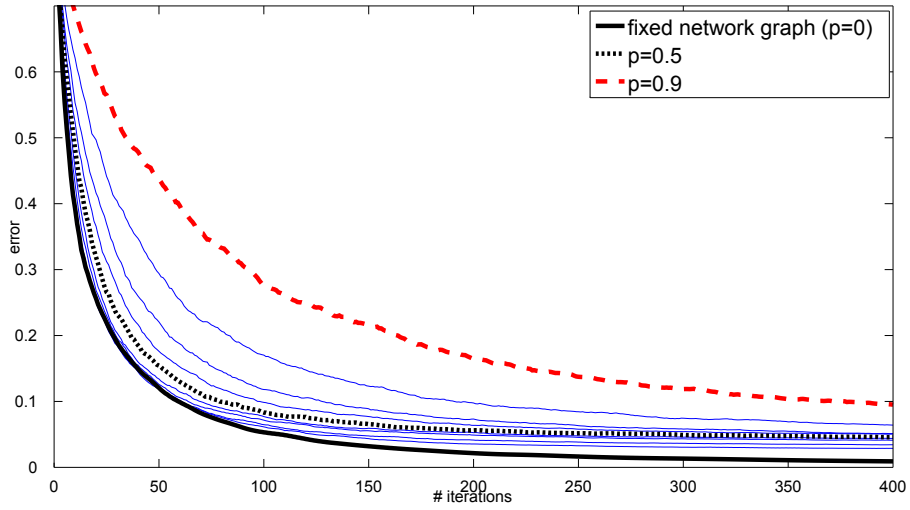


Figure 7.7: Convergence properties of D-TLS for random network graphs with different link failure probabilities, averaged over 200 Monte-Carlo runs.

fied algorithm, as described in Remark I at the end of section 7.4.2, does not have this self-healing property.

In the experiment, 2 random nodes are permanently removed[14] after 800 iterations, and another 2 after 1600 iterations. When these nodes are removed, the topology of the network changes significantly since many links disappear, and also the centralized TLS solution changes since two terms in the cost function (7.34) are removed. The $\mathbf{\Lambda}_l$'s corresponding to the disappearing links l are removed from the expressions (7.51) and (7.52). The result is shown in Fig. 7.8 for a fixed stepsize $\mu = 1$, and a single run. After every 800 iterations, the error increases significantly, but the algorithm swiftly recovers from the changes in the network.

7.6 Conclusions

In this paper, we have considered the total least squares (TLS) problem in an ad hoc wireless sensor network, where each node collects observations that yield a node-specific subset of linear equations. We have derived a distributed TLS (D-TLS) algorithm that computes the centralized TLS solution of the full set of equations in a distributed fashion, without gathering the data in a fusion center. To facilitate the use of the dual based subgradient algorithm (DBSA),

[14]Nodes can only be removed if the network graph remains connected after the removal.

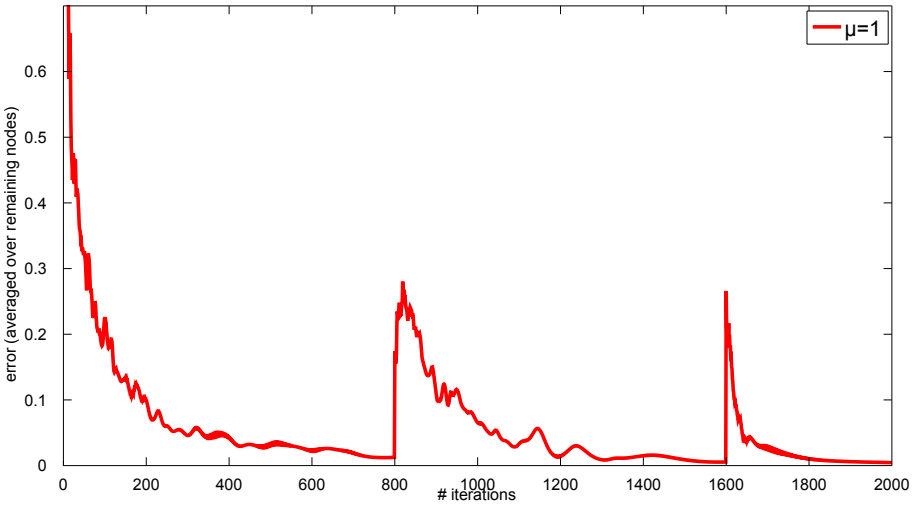


Figure 7.8: Self-healing property of the D-TLS algorithms after removal of nodes.

we have transformed the TLS problem to an equivalent convex semidefinite program (SDP) that satisfies the convergence conditions of DBSA, and yields the same solution as in the original problem. Even though we have made a detour through SDR and SDP theory, the resulting D-TLS algorithm relies on solving local TLS-like problems at each node, rather than on computationally expensive SDP optimization techniques. The algorithm is flexible and fully distributed, i.e. it does not make any assumptions on the network topology and nodes only share data with their direct neighbors through local broadcasts. Due to the flexibility and the uniformity of the network, there is no single point of failure, which makes the algorithm robust to sensor failures. We have provided Monte-Carlo simulation that demonstrate the effectiveness of the algorithm.

Acknowledgements

The authors would like to thank Prof. M. Diehl, Dr. P. Tsiaflakis and the OPTEC and DSP co-workers at the E.E. Department of K.U. Leuven for the interesting discussions with respect to the distributed TLS problem, during the GOA-MaNet seminar. The authors also want to thank the anonymous reviewers, whose comments significantly improved this manuscript.

Bibliography

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. 2033–2036 vol.4, 2001.

[2] 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, pp. 5277–5291, Oct. 2010.

[3] ——, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part II: simultaneous & asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5292–5306, Oct. 2010.

[4] ——, "Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology," *IEEE Transactions on Signal Processing*, vol. 59, pp. 2196–2210, May 2011.

[5] 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, Aug. 2007.

[6] ——, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, July 2008.

[7] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[8] G. Mateos, I. Schizas, and G. Giannakis, "Closed-form MSE performance of the distributed LMS algorithm," in *Proc. 13th IEEE Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop (DSP/SPE)*, 4-7 2009, pp. 66 –71.

[9] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 981030, 19 pages, 2009. doi:10.1155/2009/981030.

[10] ——, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, 2009.

[11] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435.

[12] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[13] C. E. Davila, "An efficient recursive total least squares algorithm for FIR adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 42, pp. 267–280, 1994.

[14] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods," *Signal Processing*, vol. 87, no. 10, pp. 2283 – 2302, 2007.

[15] D. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003, ch. 8.2.

[16] Z. quan Luo, W. kin Ma, A.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20 –34, may 2010.

[17] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.

[18] B. Johansson, C. Carretti, and M. Johansson, "On distributed optimization using peer-to-peer communications in wireless sensor networks," in *Proc. IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 16-20 2008, pp. 497 –505.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[20] H. Wolkowicz, R. Saigal, and L. Vandenberghe, Eds., *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, 2000.

[21] G. Pataki, "On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues," *Math. Oper. Res.*, vol. 23, pp. 339–358, 1998.

[22] M. Trnovska, "Strong duality conditions in semidefinite programming," *Journal of Electrical Engineering*, vol. 56, pp. 87–89, 2005.

[23] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, pp. 31–48.

[24] K.-B. Yu, "Recursive updating the eigenvalue decomposition of a covariance matrix," *IEEE Transactions on Signal Processing*, vol. 39, pp. 1136–1145, May 1991.

[25] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. international symposium on Information processing in sensor networks (IPSN)*. Piscataway, NJ, USA: IEEE Press, 2005, p. 9.

[26] D. Shah, "Gossip algorithms," *Foundations and Trends in Networking*, vol. 3, pp. 1–125, 2009.

Chapter 8

Diffusion Bias-Compensated RLS

Diffusion Bias-Compensated RLS Estimation over Adaptive Networks

Alexander Bertrand, Marc Moonen and Ali H. Sayed

Submitted for publication, 2011.

Contributions of first author

- literature study
- co-derivation and co-analysis of the BC-RLS and diffBC-RLS algorithm
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

We study the problem of distributed least-squares estimation over ad hoc adaptive networks, where the nodes have a common objective to estimate and track a parameter vector. We consider the case where there is stationary additive colored noise on both the regressors and the output response, which results in a bias on the local least-squares estimates. Assuming that the noise covariance can be estimated (or is known a priori), we first propose a bias-compensated recursive least-squares algorithm. However, this bias compensation increases the variance of the local estimates, and errors in the noise covariance estimates may still result in a residual bias. We demonstrate that the variance and the residual bias can then be significantly reduced by applying diffusion adaptation, i.e. by letting nodes combine their local estimates with those of their neighbors. We derive a necessary and sufficient condition for mean-square stability of the algorithm, under some mild assumptions. Furthermore, we derive closed-form expressions for its steady-state mean and mean-square performance. Simulation results are provided, which agree well with the theoretical results. We also consider some special cases where the mean-square stability improvement of diffusion BC-RLS over BC-RLS can be mathematically verified.

8.1 Introduction

We study the problem of distributed least-squares estimation over ad hoc adaptive networks, where the nodes collaborate to pursue a common objective, namely, to estimate and track a common deterministic parameter vector. We consider the case where there is stationary additive colored noise on both the regressors and the output response, which results in a bias on the local least-squares estimates. This is for example a common problem in the analysis of auto-regressive (AR) processes such as speech. If this bias is significant and undesired, traditional adaptive methods, such as least mean squares (LMS) or recursive least squares (RLS), are not effective.

For the white noise case, many methods have been developed that yield unbiased estimates, some of which require prior knowledge of the noise variance. A popular method is total least squares (TLS) estimation. Several adaptive TLS algorithms have been proposed, e.g., recursive TLS [1], total least-mean squares (TLMS) [2], and a distributed TLS method for ad hoc networks [3]. Under the additional assumption that the noise-free regressors are also white, the modified least-mean squares (MLMS) [4] and modified recursive least squares (MRLS) [5] algorithm have been proposed. Another important class of algorithms are based on the bias compensation principle [6], where the idea is to subtract an estimate of the asymptotic bias from the least squares estimates [7–9].

In this paper, we consider the case where the regressor noise may be colored and correlated with the noise on the output response. We also rely on the bias compensation principle, and we assume that we have a good estimate of the noise covariance[1]. A bias-compensated recursive least-squares (BC-RLS) algorithm is then proposed, based on an exponentially weighted least-squares estimation problem. The latter allows to track the parameter vector when it changes over time, by putting less weight on older samples in the estimation.

It is a common observation in estimation theory that any attempt to reduce a bias usually results in an increased variance of the estimate. This is also the case with the proposed BC-RLS algorithm. However, recent developments in adaptive filtering have demonstrated that it is possible to significantly reduce the variance by letting multiple nodes cooperate [10–16]. In this paper, we rely on the idea of diffusion adaptation [10–13, 16], where nodes combine their local estimates with the estimates of the nodes in their neighborhoods. It is known that diffusion adaptation usually results in a smaller mean square deviation[2] (MSD) at each node, without increasing the bias. Diffusion adaptation has been successfully applied to the LMS algorithm [10–12], and to the RLS algorithm [13]. In this paper, we apply diffusion to the BC-RLS algorithm, which we refer to as diffusion BC-RLS (diffBC-RLS). Simulations demonstrate that diffusion indeed reduces the MSD of the algorithm, and furthermore, that it reduces the residual bias resulting from possible errors in the noise covariance estimates.

The main contribution of this paper is the derivation of the diffusion BC-RLS algorithm, as well as the study of the steady-state performance, both for the diffusion BC-RLS and for the undiffused BC-RLS algorithms. Under some assumptions that are common in the adaptive filtering literature, we will derive a necessary and sufficient condition for the mean-square stability of (diff)BC-RLS. For some special cases, it can be mathematically verified that diffusion improves the mean-square stability of algorithm. This has also been observed in [11] for the case of diffusion LMS, i.e., cooperation has a stabilizing effect. We also derive a closed-form expression for the residual bias and the MSD in (diff)BC-RLS. The final results of this theoretical analysis have been listed in [17], but without derivations. In this paper, we provide full derivations and we add more details and discussion.

The outline of the paper is as follows. In Section 8.2, we formally define the estimation problem, and we introduce the BC-RLS algorithm. We then define the diffusion BC-RLS algorithm in Section 8.3. We analyze the diffBC-RLS algorithm (the undiffused BC-RLS algorithm is a special case) in terms of its mean and mean-square performance in Section 8.4. In Section 8.5, we con-

[1]For example, in speech analysis, this can be estimated during silent periods in between words and sentences.

[2]In the sequel, we will focus on the mean square deviation of the estimate instead of its variance, since the former is usually used to assess the mean-square performance of an adaptive filtering algorithm.

sider some special cases where some extra theoretical results can be obtained. Simulation results are presented in Section 8.6, and conclusions are drawn in Section 8.7.

Notation

In this paper, we use boldface letters for random quantities and normal font for non-random (deterministic) quantities or samples of random quantities. We use capital letters for matrices and small letters for vectors. The superscript H denotes complex-conjugate transposition. The index i is used to denote time instants, and the index k is used to denote different nodes in a network with N nodes, defining the set of nodes $\mathcal{J}$. We use $E\{\mathbf{x}\}$ to denote the expected value of a random quantity $\mathbf{x}$.

8.2 Least Squares Estimation with Bias Compensation

8.2.1 Problem Statement

Consider an ad hoc sensor network with N nodes (the set of nodes is denoted by $\mathcal{J}$). The objective for each node is to estimate a common deterministic $M \times 1$ parameter vector w^o. At every time instant i, node k collects a measurement $d_k(i)$ (referred to as the 'output response') that is assumed to be related to the unknown vector w^o by

$$d_k(i) = \overline{u}_{k,i} w^o + v_k(i) \tag{8.1}$$

where the regressor $\overline{u}_{k,i}$ is a sample of an $1 \times M$ stochastic row vector[3] $\overline{\mathbf{u}}_{k,i}$, and $v_k(i)$ is a sample of a zero-mean stationary noise process $\mathbf{v}_k$ with variance $\sigma^2_{v_k}$. In [11–15], it was assumed that node k also has access to the regressors $\{\overline{u}_{k,i}\}$. Here, we assume that node k observes noisy regressors $\{u_{k,i}\}$, given by

$$u_{k,i} = \overline{u}_{k,i} + n_{k,i} \tag{8.2}$$

with the $1 \times M$ vector $n_{k,i}$ denoting a sample of a zero-mean stationary noise process $\mathbf{n}_k$ with covariance matrix $R_{n_k} = E\{\mathbf{n}_k^H \mathbf{n}_k\}$. We assume that $\mathbf{n}_k$ is uncorrelated with the regressors $\overline{\mathbf{u}}_{k,i}$, and that $\mathbf{n}_k$ and $\mathbf{v}_k$ are correlated[4] , yielding a non-zero covariance vector $r_{n_k v_k} = E\{\mathbf{n}_k^H \mathbf{v}_k\}$.

[3]We adopt the notation of [18], i.e., the regressors are defined as row vectors, rather than column vectors.

[4]For example, this may be the case for the estimation of the prediction coefficients of an auto-regressive (AR) process where the data is corrupted by additive colored noise, e.g., in the analysis of speech signals.

The local least squares (LS) estimate of w^o at node k at time instant i, based on the noisy regressors, is the solution of the optimization problem

$$\hat{w}_{k,i} = \arg\min_{w} \sum_{j=1}^{i} \left(d_k(j) - u_{k,j} w\right)^2 + \delta \|w\|_2^2 \tag{8.3}$$

where δ is a small positive number that serves as a regularization parameter. The solution of (8.3) is given by

$$\hat{w}_{k,i} = \hat{R}_{u_k,i}^{-1} \hat{r}_{u_k d_k,i} \tag{8.4}$$

where

$$\hat{R}_{u_k,i} = \frac{1}{i+1} \left(\sum_{j=1}^{i} u_{k,j}^H u_{k,j} + \delta I_M \right) \tag{8.5}$$

$$\hat{r}_{u_k d_k,i} = \frac{1}{i+1} \sum_{j=1}^{i} u_{k,j}^H d_k(j) \tag{8.6}$$

and where I_M denotes the $M \times M$ identity matrix. The normalization with $1/(i+1)$ does not have an influence on $\hat{w}_{k,i}$, but its purpose will become clear in Section 8.2.2. Since we use noisy regressors, the LS estimate has a bias. In the case of stationary and ergodic data, it can be verified that

$$w_k^{\mathrm{LS}} = w^o + R_{u_k}^{-1} r_{n_k v_k} - R_{u_k}^{-1} R_{n_k} w^o \; . \tag{8.7}$$

where $w_k^{\mathrm{LS}} = \lim_{i\to\infty} \hat{w}_{k,i} = R_{u_k}^{-1} r_{u_k d_k}$ with $R_{u_k} = E\{\mathbf{u}_{k,i}^H \mathbf{u}_{k,i}\}$ and $r_{u_k d_k} = E\{\mathbf{u}_{k,i}^H \mathbf{d}_k(i)\}$, for all $i \in \mathbb{N}$, where $\mathbf{u}_{k,i}$ and $\mathbf{d}_k(i)$ are defined as the stochastic processes that generate the samples $u_{k,i}$ and $d_k(i)$ defined in (8.2) and (8.1), respectively. It is noted that w_k^{LS} is in fact a minimum mean-square error estimate (MMSE), but we keep the superscript LS to emphasize that it is a limit case of the LS estimate (8.4). Let $w_k^b = w_k^{\mathrm{LS}} - w^o$, then the bias w_k^b of the MMSE estimate w_k^{LS} is equal to

$$w_k^b = R_{u_k}^{-1} \left(r_{n_k v_k} - R_{n_k} w^o\right) \tag{8.8}$$

8.2.2 Bias-Compensated Least Squares (BC-LS)

Several BC-LS algorithms have been proposed for the white noise case ($R_{n_k} = \sigma_{n_k}^2 I_M$), which are asymptotically unbiased when the number of observations goes to infinity [7–9]. All these BC-LS algorithms are based on the bias compensation principle [6], i.e., if the asymptotic bias w_k^b can be estimated, it can be subtracted from the LS estimate $\hat{w}_{k,i}$ to obtain the unbiased estimate (generalized here to incorporate colored noise and mutually correlated noise):

$$\psi_{k,i} \triangleq \hat{w}_{k,i} + \hat{R}_{u_k,i}^{-1} \left(\hat{R}_{n_k} w^o - \hat{r}_{n_k v_k}\right) \tag{8.9}$$

where $\hat{r}_{n_k v_k}$ and $\hat{R}_{n_k}$ are estimates of $r_{n_k v_k}$ and R_{n_k}, respectively. It is assumed that good estimates $\hat{r}_{n_k v_k}$ and $\hat{R}_{n_k}$ are available. In the case of white noise, these estimates can be computed blindly during operation of the algorithm [7–9].

Since w^o is unknown in (8.9), it has to be replaced with an estimate. The common approach is then to use $\psi_{k,i-1}$ as an estimate for w^o in (8.9). We then obtain the recursive algorithm

$$\psi_{k,i} = \hat{w}_{k,i} + \hat{R}_{u_k,i}^{-1} \left(\hat{R}_{n_k} \psi_{k,i-1} - \hat{r}_{n_k v_k} \right) . \tag{8.10}$$

8.2.3 Bias-Compensated Recursive Least Squares (BC-RLS)

The BCLS algorithm can be modified so that it fits into an adaptive filtering context, where also exponential weighting can be incorporated (for tracking purposes). The exponentially-weighted LS estimate (at node k) solves the optimization problem

$$\hat{w}_{k,i} = \arg\min_{w} \sum_{j=1}^{i} \lambda^{i-j} \left(d_k(j) - u_{k,j} w \right)^2 + \lambda^i \delta \|w\|_2^2 \tag{8.11}$$

where $0 \ll \lambda \leq 1$ is a forgetting factor, putting more weight on more recent observations. The solution of this problem is again given by (8.4), but the estimates $\hat{R}_{u_k,i}$ and $\hat{r}_{u_k d_k,i}$ are now redefined as

$$\hat{R}_{u_k,i} = \sum_{j=1}^{i} \lambda^{i-j} u_{k,j}^H u_{k,j} + \delta I_M \tag{8.12}$$

$$\hat{r}_{u_k d_k,i} = \sum_{j=1}^{i} \lambda^{i-j} u_{k,j}^H d_k(j) . \tag{8.13}$$

It is noted that the effective window length is equal to $\frac{1}{1-\lambda} = \sum_{j=0}^{\infty} \lambda^j$, and since there is no normalization for the window length, $\hat{R}_{u_k,i}$ and $\hat{r}_{u_k d_k,i}$ can be considered to be estimates of $\frac{1}{1-\lambda} R_{u_k}$ and $\frac{1}{1-\lambda} r_{u_k d_k}$, respectively [18]. From now on, $\hat{w}_{k,i}$ refers to the solution of the exponentially weighted LS problem (8.11) and not to the solution of the unweighted LS problem (8.3), and the same holds for $\hat{R}_{u_k,i}$ and $\hat{r}_{u_k d_k,i}$,now defined by (8.12)-(8.13). In Section 8.4, we will show that, under certain assumptions, (8.11) is an unbiased estimate of the local MMSE solution at node k, i.e., w_k^{LS}.

The solution of (8.11) is recursively computed by means of the recursive least squares (RLS) algorithm [18]:

$$P_{k,i} = \lambda^{-1} \left(P_{k,i-1} - \frac{\lambda^{-1} P_{k,i-1} u_{k,i}^H u_{k,i} P_{k,i-1}}{1 + \lambda^{-1} u_{k,i} P_{k,i-1} u_{k,i}^H} \right) \tag{8.14}$$

$$\hat{w}_{k,i} = \hat{w}_{k,i-1} + P_{k,i} u_{k,i}^H \left(d_k(i) - u_{k,i}\hat{w}_{k,i-1}\right) \tag{8.15}$$

with $\hat{w}_{k,0} = 0$ and $P_{k,0} = \delta^{-1} I_M$. At every time instant i, the matrix $P_{k,i}$ is equal to $\hat{R}_{u_k,i}^{-1}$ as defined in (8.12).

Using this construction, (8.10) is transformed into the recursion

$$\psi_{k,i} = \hat{w}_{k,i} + \frac{1}{1-\lambda} P_{k,i} \left(\hat{R}_{n_k} \psi_{k,i-1} - \hat{r}_{n_k v_k}\right) . \tag{8.16}$$

The factor $\frac{1}{1-\lambda}$ scales $\hat{R}_{n_k}$ and $\hat{r}_{n_k v_k}$ to match with the effective window length in (8.12)-(8.13). We will refer to the above algorithm as bias-compensated RLS (BC-RLS). It is noted that (8.16) reduces to the BC-LS recursion (8.10) if $\lambda = 1$ and if the scaling factor $\frac{1}{1-\lambda}$ in (8.16) is omitted. We do not provide a convergence analysis of BC-RLS here, since it is a special case of the diffusion BC-RLS algorithm described in the sequel, in particular when cooperation is turned off.

8.3 Diffusion BC-RLS

In a sensor network, each node k has its own node-specific BC-RLS estimate of w^o, denoted by $\psi_{k,i}$. It is often observed in estimation theory that bias removal introduces a larger variance and vice versa (see, e.g., [19]). This also often holds in the case of BC-RLS, since the bias compensation usually increases the variance or MSD of the estimates in each node due to the addition of the extra term . It is to be expected that the spatial average of all the $\psi_{k,i}$'s provides a better estimate for w^o, with a smaller MSD. This average could in principle be computed in a distributed fashion by iterative consensus averaging algorithms [20]. The main idea of these algorithms is to collect the estimates $\{\psi_{l,i}\}$ from the neighbors of node k at time i and to iterate over them repeatedly by computing a weighted average, i.e.,

1. Initialize $j \leftarrow 0$ and $\psi_{l,i}^0 = \psi_{l,i}$.
2. Compute a weighted average

$$\psi_{k,i}^{j+1} = \sum_{l \in \mathcal{N}_k} a_{kl} \psi_{l,i}^j \tag{8.17}$$

3. $j \leftarrow j + 1$
4. Return to step 2.

Here, $\mathcal{N}_k$ denotes the set of neighboring nodes of node k (node k included), and a_{kl} is the entry in row k and column l of an $N \times N$ combiner matrix[5] A, where A satisfies

$$A\mathbb{1} = \mathbb{1} \tag{8.18}$$

with $\mathbb{1} = [1 \dots 1]^H$ and where $a_{kl} = 0$ if $l \notin \mathcal{N}_k$. The matrix A can be any right-stochastic matrix, but with some constraints due to the network topology. After convergence, the result of (8.17) becomes the actual estimate $\psi_{k,i}$ for node k at time i. Thus, observe that at every time instant i, multiple consensus iterations need to be applied to the data $\{\psi_{l,i}\}$ to approximate their mean and obtain an improved $\psi_{k,i}$.

Applying consensus averaging in the case of BC-RLS would therefore require a 2-step approach involving two time-scales: one over i and another over j, in between successive i's. First, the nodes estimate a local $\psi_{k,i}$ based on (8.16), after which an average consensus algorithm is started to iteratively compute

$$\psi_{k,i} = \frac{1}{N} \sum_{l=1}^{N} \psi_{l,i} \tag{8.19}$$

at each node $k \in \mathcal{J}$. This two-step approach is impractical in real-time systems with large sampling rates since the consensus averaging requires multiple iterations over j for every single iteration i, resulting in a large amount of communication bandwidth and processing power. By applying diffusion strategies instead (see, e.g., [11, 12]), the iterations of the consensus averaging are merged with those of the BC-RLS algorithm, i.e., the consensus averaging is cut off after a single iteration over j. As a result, only one iteration index remains, and the computational complexity and communication bandwidth are significantly reduced while the network is endowed with improved learning and tracking abilities. The following table summarizes the diffusion BC-RLS (diffBC-RLS) algorithm that would result from a diffusion strategy. Observe how the left-hand side of (8.23) is a new variable $w_{k,i}$, which then enters into the update (8.22). In contrast, in a consensus implementation (apart from the second time-scale), the variables that appear on both sides of (8.17) are the same ψ variables. In (8.22)-(8.23), a filtering operation is embedded into (8.22) to map $w_{k,i-1}$ to $\psi_{k,i}$ at each node and all $\psi_{l,i}$ are then combined into $w_{k,i}$ in (8.23).

[5]This combiner matrix has to satisfy some constraints to let the consensus averaging algorithm converge [20]. However, since the diffusion BC-RLS algorithm, as derived in the sequel, does not require these constraints, we omit them here.

Diffusion BC-RLS algorithm

Start with $w_{k,0} = 0$, $\hat{w}_{k,0} = 0$ and $P_{k,0} = \delta^{-1} I_M$ for each node $k \in \mathcal{J}$.
For every time instant $i > 0$, repeat

1. **RLS update**: for every node $k \in \mathcal{J}$, repeat

$$P_{k,i} = \lambda^{-1} \left(P_{k,i-1} - \frac{\lambda^{-1} P_{k,i-1} u_{k,i}^H u_{k,i} P_{k,i-1}}{1 + \lambda^{-1} u_{k,i} P_{k,i-1} u_{k,i}^H} \right) \tag{8.20}$$

$$\hat{w}_{k,i} = \hat{w}_{k,i-1} + P_{k,i} u_{k,i}^H \left(d_k(i) - u_{k,i} \hat{w}_{k,i-1} \right) . \tag{8.21}$$

2. **Bias correction update**: for every node $k \in \mathcal{J}$, repeat

$$\psi_{k,i} = \hat{w}_{k,i} + \frac{1}{1-\lambda} P_{k,i} \left(\hat{R}_{n_k} w_{k,i-1} - \hat{r}_{n_k v_k} \right) . \tag{8.22}$$

3. **Spatial update**: for every node $k \in \mathcal{J}$, repeat

$$w_{k,i} = \sum_{l \in \mathcal{N}_k} a_{kl} \psi_{l,i} \tag{8.23}$$

Remark: It is noted that the RLS update (8.20)-(8.21) in node k is spatially isolated, i.e., it does not involve cooperation between the nodes. One may be tempted to also apply diffusion to the RLS estimates, based on the diffusion RLS algorithm in [13]. However, applying diffusion on (8.20)-(8.21) will change the local bias in each node, i.e., the local bias at node k will not satisfy (8.8) anymore. Since the bias compensation (8.22) is based on (8.8), and only relies on local statistics, it will not match with the actual bias. Therefore, diffusion of the RLS estimates in combination with the bias compensation (8.22) is only possible when an invariant spatial profile can be assumed, such that the bias (8.8) is the same in each node. Furthermore, the MSD of the RLS estimates $\hat{w}_{k,i}$ is often negligible compared to the MSD of the BC-RLS estimates in (8.16). Therefore, even when an invariant spatial profile can be assumed, there is usually only a marginal performance gain which does not outweigh the cost of doubling the communication load.

8.4 Analysis

In this section, we analyze the steady-state performance of the diffBC-RLS algorithm described in Section 8.3. First, we provide a closed-form expression for the bias if there are estimation errors in $\hat{R}_{n_k}$ and $\hat{r}_{n_k v_k}$ (under some standard

ergodicity assumptions). Secondly, if $\hat{R}_{n_k} = R_{n_k}$ and $\hat{r}_{n_k v_k} = r_{n_k v_k}$, we show that the diffBC-RLS algorithm is asymptotically unbiased and we provide a closed-form expression for the mean-square deviation (MSD), i.e.

$$\mathrm{MSD}_k = E\{\|\tilde{w}_{k,i}\|^2\} \tag{8.24}$$

where

$$\tilde{w}_{k,i} = w^o - w_{k,i} \ . \tag{8.25}$$

It is noted that all results of the analysis of diffBC-RLS also apply to the undiffused BC-RLS algorithm (8.16), by choosing the combiner matrix A equal to the identity matrix.

8.4.1 Data Model

The performance analysis of adaptive filters is rather challenging [18, 21, 22], and it is common to adopt some simplifying assumptions to gain insight in the properties of these algorithms. For the analysis of the diffBC-RLS algorithm, we will introduce some assumptions that are similar to what is traditionally used in the adaptive filtering literature. Simulations show that the theoretical results that are obtained under these assumptions match well with the true performance of the algorithm, for forgetting factors λ that are close to unity and for stationary data.

Assumption 1: The regressors $\overline{u}_{k,i}$ and the additive noise components $n_{k,i}$ are both zero-mean and temporally independent. Furthermore, the covariance matrix $R_{\overline{u}_{k,i}} = E\{\overline{\mathbf{u}}_{k,i}^H \overline{\mathbf{u}}_{k,i}\}$ is time-invariant, i.e., $R_{\overline{u}_{k,i}} = R_{\overline{u}_k}$, $\forall\, i \in \mathbb{N}$. We will therefore often omit the index i in the sequel, when referring to random processes.

It is noted that this assumption also implies that the same conditions hold for the noisy regressors $u_{k,i}$, i.e. $R_{u_{k,i}} = R_{u_k}$, $\forall\, i \in \mathbb{N}$. Furthermore, since the stochastic processes $\overline{\mathbf{u}}_k$ and $\mathbf{n}_k$ are assumed to be uncorrelated, we find that

$$R_{u_k} = R_{\overline{u}_k} + R_{n_k} \ . \tag{8.26}$$

Assumption 2: All data is spatially uncorrelated, i.e., for $k \neq l$: $E\{\mathbf{u}_k^H \mathbf{u}_l\} = 0$, $E\{\overline{\mathbf{u}}_k^H \overline{\mathbf{u}}_l\} = 0$, $E\{\mathbf{n}_k^* \mathbf{n}_l\} = 0$, $E\{\mathbf{v}_k^* \mathbf{v}_l\} = 0$, $E\{\mathbf{v}_k^* \mathbf{n}_l\} = 0$ and $E\{\mathbf{u}_k^H \mathbf{d}_l\} = 0$.

Since we only perform a steady-state analysis of the algorithm, we will consider the steady-state behavior of the matrix $P_{k,i}$. As $i \to \infty$, we find from (8.12), and the fact that $P_{k,i}^{-1} = \hat{R}_{u_k,i}$, that

$$\lim_{i\to\infty} E\{P_{k,i}^{-1}\} = \frac{1}{1-\lambda} R_{u_k} \triangleq P_k^{-1} \ . \tag{8.27}$$

The following two assumptions are made to make the analysis of diffBC-RLS tractable, and both of them are common in the analysis of RLS-type algorithms (see for example [18]).

Assumption 3: $\exists\, i_0$ such that for all $i > i_0$, $P_{k,i}$ and $P_{k,i}^{-1}$ can be replaced with their expected values, i.e. $\exists\, i_0$, such that for all $i > i_0$:

$$P_{k,i} \approx E\{P_{k,i}\} \tag{8.28}$$

$$P_{k,i}^{-1} \approx E\{P_{k,i}^{-1}\}\ . \tag{8.29}$$

Assumption 4: $\exists\, i_0$ such that for all $i > i_0$:

$$E\{P_{k,i}\} \approx E\{P_{k,i}^{-1}\}^{-1} = P_k = (1-\lambda)R_{u_k}^{-1}\ . \tag{8.30}$$

The last assumption is a coarse approximation, since the expected values $E\{P_{k,i}^{-1}\}$ and $E\{P_{k,i}\}$ do not necessarily share the same inverse relation as their arguments. However, for λ close to unity and a not too large condition number for R_{u_k}, this is a good approximation [13, 18]. However, even in cases where this approximation is not very good, the formulas that are derived in the analysis are still useful to analyze the influence of different parameters, i.e., they usually reflect the correct trends when parameters are varied.

Remark I: Assumption 3 removes some temporal variations in the algorithm, which usually results in an underestimate of the MSD. Assumption 4 increases this effect even more. This can be intuitively explained as follows. Assume that we can approximate the stochastic matrix $P_{k,i}$ with the model $P_{k,i} = Q_{k,i}\Lambda_{k,i}Q_{k,i}^H$, where $\Lambda_{k,i}$ is a stochastic diagonal matrix, and $Q_{k,i}$ a deterministic unitary matrix. In this case $E\{P_{k,i}\} = Q_{k,i}E\{\Lambda_{k,i}\}Q_{k,i}^H$. By using Jensen's inequality, we know that for any positive diagonal matrix Σ

$$E\{\Sigma^{-1}\} \geq E\{\Sigma\}^{-1} \tag{8.31}$$

(this is an elementwise inequality). By substituting $\Sigma = \Lambda_{k,i}^{-1}$, we find that

$$E\{\Lambda_{k,i}\} \geq E\{\Lambda_{k,i}^{-1}\}^{-1}\ . \tag{8.32}$$

As a consequence, the norm of $E\{P_{k,i}\}$ will be larger than the norm of $E\{P_{k,i}^{-1}\}^{-1}$. Hence, when using approximation (8.30), we replace $E\{P_{k,i}\}$ with a matrix that has a smaller norm. This usually results in an underestimate of the MSD.

Remark II: For notational convenience, we will replace the approximate equality signs '$\approx$' in (8.28)-(8.30) with strict equality signs '=' in the sequel.

8.4.2 Mean Performance

In this subsection, we analyze the steady-state mean performance of the diffBC-RLS algorithm, i.e., we derive a closed-form expression for $E\{\tilde{w}_{k,i}\}$ when i goes to infinity. In this analysis, we incorporate possible estimation errors on the noise covariances, i.e.,

$$\hat{R}_{n_k} = R_{n_k} + \Delta R_{n_k} \tag{8.33}$$

$$\hat{r}_{n_k v_k} = r_{n_k v_k} + \Delta r_{n_k v_k} \ . \tag{8.34}$$

We will first derive an expression for the asymptotic bias of the RLS estimate $\hat{w}_{k,i}$. Similar to (8.25), we define $\check{w}_{k,i} = w^o - \hat{w}_{k,i}$. With (8.21), we readily find that

$$\check{w}_{k,i} = \check{w}_{k,i-1} - P_{k,i} u_{k,i}^H \left(d_k(i) - u_{k,i}\hat{w}_{k,i-1}\right) \ . \tag{8.35}$$

Substituting (8.1) and (8.2) into (8.35), we obtain

$$\check{w}_{k,i} = \check{w}_{k,i-1} - P_{k,i}\overline{u}_{k,i}^H \overline{u}_{k,i} w^o - P_{k,i} n_{k,i}^H \overline{u}_{k,i} w^o - P_{k,i} u_{k,i}^H v_k(i) + P_{k,i} u_{k,i}^H u_{k,i} \hat{w}_{k,i-1} \ . \tag{8.36}$$

Taking the expectation of both sides, and using (8.26), (8.28)-(8.30), we find that for sufficiently large i

$$E\{\check{w}_{k,i}\} = E\{\check{w}_{k,i-1}\} - P_k(R_{u_k} - R_{n_k})w^o - P_k r_{n_k v_k} + (1-\lambda)E\{\hat{w}_{k,i-1}\} \ . \tag{8.37}$$

Again using (8.30), we obtain

$$E\{\check{w}_{k,i}\} = \lambda E\{\check{w}_{k,i-1}\} + P_k(R_{n_k} w^o - r_{n_k v_k}) \ . \tag{8.38}$$

Expanding the recursion in (8.38), we find that

$$E\{\check{w}_{k,i}\} = \lambda^{i-i_0} E\{\check{w}_{k,i_0}\} + \sum_{j=i_0}^{i-1} \lambda^{j-i_0} P_k(R_{n_k} w^o - r_{n_k v_k}) \tag{8.39}$$

where i_0 is chosen such that Assumptions 3 and 4 remain valid. Letting i go to infinity, we obtain

$$\lim_{i\to\infty} E\{\check{w}_{k,i}\} = \frac{1}{1-\lambda} P_k(R_{n_k} w^o - r_{n_k v_k}) \ . \tag{8.40}$$

Not surprisingly, we find that the asymptotic bias of the exponentially weighted RLS algorithm is equal to the asymptotic bias (8.8) of the unweighted least-squares estimate.

Let us now introduce some notation that is required to describe the diffusion process of diffBC-RLS, based on stacked variables from all nodes. Let

$$\begin{array}{lcll}
\boldsymbol{w}_i & = & \mathrm{col}\{w_{1,i}, \ldots, w_{N,i}\} & (MN \times 1) \\
\hat{\boldsymbol{w}}_i & = & \mathrm{col}\{\hat{w}_{1,i}, \ldots, \hat{w}_{N,i}\} & (MN \times 1) \\
\boldsymbol{r}_{nv} & = & \mathrm{col}\{r_{n_1 v_1}, \ldots, r_{n_N v_N}\} & (MN \times 1) \\
\boldsymbol{w}^o & = & \mathbb{1} \otimes w^o & (MN \times 1) \\
\mathcal{A} & = & A \otimes I_M & (MN \times MN) \\
\mathcal{P}_i & = & \mathrm{blockdiag}\{P_{1,i}, \ldots, P_{N,i}\} & (MN \times MN) \\
\mathcal{R}_n & = & \mathrm{blockdiag}\{R_{n_1}, \ldots, R_{n_N}\} & (MN \times MN) \\
\mathcal{R}_u & = & \mathrm{blockdiag}\{R_{u_1}, \ldots, R_{u_N}\} & (MN \times MN)
\end{array}$$

where col{.} denotes a stacked column vector, $\otimes$ denotes a Kronecker product and blockdiag{.} denotes a block-diagonal matrix. All the derived quantities (such as $\tilde{w}_i$, $\hat{R}_n$, etc.) have a similar notation for the stacked case, but are omitted for conciseness. Using this notation, and by combining (8.22) and (8.23), the recursion of the diffusion RLS algorithm can now be written as

$$w_i = \mathcal{A}\left(\hat{w}_i + \frac{1}{1-\lambda}\mathcal{P}_i(\hat{\mathcal{R}}_n w_{i-1} - \hat{r}_{nv})\right) . \tag{8.41}$$

Subtracting (8.41) from w^o, and using the fact that $w^o = \mathcal{A}w^o$, yields

$$\tilde{w}_i = \mathcal{A}\left(\check{w}_i - \frac{1}{1-\lambda}\mathcal{P}_i(\hat{\mathcal{R}}_n w_{i-1} - \hat{r}_{nv})\right) . \tag{8.42}$$

Taking the expectation of both sides, and using (8.33), (8.34) and (8.40), we obtain (for $i \to \infty$):

$$\begin{aligned} E\{\tilde{w}_i\} &= \frac{1}{1-\lambda}\mathcal{A}\mathcal{P}\mathcal{R}_n E\{\tilde{w}_{i-1}\} - \frac{1}{1-\lambda}\mathcal{A}\mathcal{P}\left(\Delta\mathcal{R}_n E\{w_{i-1}\} - \Delta r_{nv}\right) \\ &= \frac{1}{1-\lambda}\mathcal{A}\mathcal{P}\mathcal{R}_n E\{\tilde{w}_{i-1}\} \\ &\quad - \frac{1}{1-\lambda}\mathcal{A}\mathcal{P}\left(\Delta\mathcal{R}_n E\{w_{i-1}\} - \Delta r_{nv} + \Delta\mathcal{R}_n w^o - \Delta\mathcal{R}_n w^o\right) \\ &= \frac{1}{1-\lambda}\mathcal{A}\mathcal{P}\hat{\mathcal{R}}_n E\{\tilde{w}_{i-1}\} - \frac{1}{1-\lambda}\mathcal{A}\mathcal{P}\left(\Delta\mathcal{R}_n w^o - \Delta r_{nv}\right) . \end{aligned} \tag{8.43}$$

Notice that, in the last step, we incorporate the term with $\Delta\mathcal{R}_n$ into the first term, such that $\mathcal{R}_n$ is transformed into $\hat{\mathcal{R}}_n$. Expanding the recursion (8.43), and using $\mathcal{P} = (1-\lambda)\mathcal{R}_u^{-1}$ (Assumption 4), we find that

$$\begin{aligned} E\{\tilde{w}_i\} &= \left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{i-i_0} E\{\tilde{w}_{i_0}\} \\ &\quad - \left(\sum_{j=i_0}^{i-1}\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{j-i_0}\right)\mathcal{A}\mathcal{R}_u^{-1}\left(\Delta\mathcal{R}_n w^o - \Delta r_{nv}\right) \end{aligned} \tag{8.44}$$

where i_0 is chosen such that Assumptions 3 and 4 remain valid. From this equation, it is observed that stability in the mean[6] of the diffBC-RLS algorithm is obtained if and only if

$$\boxed{\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) < 1} \tag{8.45}$$

where $\rho(X)$ denotes the spectral radius of the matrix X, i.e., the magnitude of the eigenvalue of X with largest absolute value. Indeed, if this spectral radius is strictly smaller than 1, the first term vanishes when $i \to \infty$ and the summation

[6] In Subsection 8.4.3, we will show that condition (8.45) for stability in the mean also implies mean-square stability of the diffBC-RLS algorithm.

in the second term converges. The latter follows from the Taylor expansion of a matrix $(I_M - X)^{-1}$ for any $M \times M$ matrix X satisfying $\rho(X) < 1$, which is given by

$$(I_M - X)^{-1} = \sum_{j=0}^{\infty} X^j \ . \tag{8.46}$$

Therefore, if (8.45) holds, it follows that the asymptotic bias of the diffBC-RLS estimates is equal to

$$\boxed{E\{\tilde{w}_i\} = \left(I_{MN} - \mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{-1} \mathcal{A}\mathcal{R}_u^{-1}\left(\Delta r_{nv} - \Delta\mathcal{R}_n w^o\right).} \tag{8.47}$$

A first important observation is that the estimate is asymptotically unbiased if $\Delta\mathcal{R}_n = 0$ and $\Delta r_{nv} = 0$, i.e., if there is perfect knowledge of the noise covariance. The smaller the error in $\hat{\mathcal{R}}_n$ and $\hat{r}_{nv}$, the smaller the resulting bias.

Note that setting $\mathcal{A} = I_{MN}$ yields the bias of the undiffused BC-RLS estimates (8.16). It is not possible to make general statements whether diffusion ($\mathcal{A} \neq I_{MN}$) will decrease the bias of the estimates, since this depends on the space-time data statistics (represented by $\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n$) and the network topology (represented by $\mathcal{A}$). This also holds for the stability condition (8.45). However, since $\mathcal{A}$ has a unity spectral radius, it often has a 'non-expanding' effect, and therefore improves the stability (i.e. the spectral radius in (8.45) decreases). For some particular cases, it can be mathematically verified that the stability indeed increases, and we refer to section 8.5 for some examples. If the stability increases, this often yields a smaller bias. To see this, observe that $\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \leq \rho\left(\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)$ implies that

$$\rho\left(\left(I_{MN} - \mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{-1}\right) \leq \rho\left(\left(I_{MN} - \mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{-1}\right) \ . \tag{8.48}$$

This implies that a mapping based on the lefthand side of (8.48) is 'more contractive' or 'less expanding' than the mapping on the righthand side (corresponding to the undiffused case). Therefore, the bias given in (8.47) with $\mathcal{A} \neq I_{MN}$ is often (but not necessarily) smaller than with $\mathcal{A} = I_{MN}$. Note that, if diffusion is applied, there is an additional effect, namely an averaging operator $\mathcal{A}$ applied to the error vector $\mathcal{R}_u^{-1}\left(\Delta r_{nv} - \Delta\mathcal{R}_n w^o\right)$. If the combiner matrix A is symmetric, this is a non-expanding mapping, i.e. $\|\mathcal{A}x\| \leq \|x\|$ for all x.

Remark: It is noted that one has to be careful when using the stability condition (8.45), as it is derived based on Assumption 4. For small values of λ, this assumption is not satisfied, and the algorithm may become unstable, even if (8.45) holds. Decreasing λ is observed to make the algorithm less stable, since the true matrix $P_{k,i}$ has a larger norm than $R_{u_k,i}^{-1}$ due to Jensens inequality (see Remark I in subsection 8.4.1).

8.4.3 Mean-Square Performance

In this subsection, we analyze the steady-state mean-square performance[7] of the diffBC-RLS algorithm, i.e., we derive a closed-form expression for $\text{MSD}_k = E\{\|\tilde{w}_{k,i}\|^2\}$ when i goes to infinity. To make the analysis tractable, we assume that $\Delta\mathcal{R}_n = 0$ and $\Delta r_{nv} = 0$.

Let

$$w^{\text{LS}} \quad = \quad \text{col}\{w_1^{\text{LS}}, \ldots, w_N^{\text{LS}}\} \quad (MN \times 1) \tag{8.49}$$

where w_k^{LS} is the MMSE estimate in node k, defined in (8.7). From (8.7), we find that

$$w^o = \mathcal{A}w^o = \mathcal{A}\left(w^{\text{LS}} - \mathcal{R}_u^{-1}r_{nv} + \mathcal{R}_u^{-1}\mathcal{R}_n w^o\right) \ . \tag{8.50}$$

Subtracting w_i from both sides in (8.50), and substituting the diffBC-RLS recursion (8.22)-(8.23), we obtain (with $\mathcal{P}_i = \mathcal{P} = (1-\lambda)\mathcal{R}_u^{-1}$ (Assumption 4)):

$$\tilde{w}_i = \mathcal{A}m_i + \mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n\tilde{w}_{i-1} \tag{8.51}$$

where

$$m_i \triangleq w^{\text{LS}} - \hat{w}_i \ . \tag{8.52}$$

By expanding the recursion (8.51), we find that

$$\tilde{w}_i = \sum_{j=i_0}^{i}\left(\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n\right)^{i-j}\mathcal{A}m_j + \left(\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n\right)^{i-i_0}\tilde{w}_{i_0} \tag{8.53}$$

where i_0 is chosen such that Assumptions 3 and 4 remain valid. If the stability condition (8.45) is satisfied, the second term in (8.53) vanishes when $i \to \infty$, so we will omit it in the sequel. For the sake of an easy exposition, we will set $i_0 = 0$, which does not affect the righthand side of (8.53) for $i \to \infty$, and if (8.45) holds. Using the notation $\|x\|_\Sigma^2 = x^H\Sigma x$, we find the following expression for the MSD of node k (in steady-state):

$$\text{MSD}_k = E\{\|\tilde{w}_i\|_{\mathcal{E}_k}^2\} = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}E\{m_{i-m}^H\mathcal{B}_k^{mn}m_{i-n}\} \tag{8.54}$$

where

$$\mathcal{B}_k^{mn} = \mathcal{A}^H\left(\mathcal{R}_n\mathcal{R}_u^{-1}\mathcal{A}^H\right)^m\mathcal{E}_k\left(\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n\right)^n\mathcal{A} \tag{8.55}$$

and where $\mathcal{E}_k = E_k \otimes I_M$ with E_k denoting an $N \times N$ matrix with zero-valued entries, except for a one on the k-th diagonal entry. The matrix $\mathcal{E}_k$ serves as a selector matrix to select the part of $\tilde{w}_i$ corresponding to the k-th node.

[7] In the mean-square analysis of adaptive filters, one is usually also interested in the so-called excess mean-square error (EMSE) defined by $E\{\|u_{k,i}\tilde{w}_{k,i-1}\|^2\}$. However, since the goal of BC-RLS is to obtain an unbiased estimate for w^o, and not to minimize the EMSE, we do not consider the latter.

Expression (8.54) can be rewritten with a trace operator Tr(.):

$$E\{\|\tilde{w}_i\|^2_{\mathcal{E}_k}\} = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \mathrm{Tr}\left(\mathcal{B}_k^{mn} E\{m_{i-m} m_{i-n}^H\}\right) \ . \tag{8.56}$$

In Appendix 8.A, the following expression is derived:

$$E\{m_{i-m} m_{i-n}^H\} = \lambda^{|m-n|} E\{m_i m_i^H\} \ . \tag{8.57}$$

With this result, we can rewrite (8.56) as

$$E\{\|\tilde{w}_i\|^2_{\mathcal{E}_k}\} = \mathrm{Tr}\left(\mathcal{M}_k E\{m_i m_i^H\}\right) \ . \tag{8.58}$$

where

$$\mathcal{M}_k = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \lambda^{|m-n|} \mathcal{B}_k^{mn} \ . \tag{8.59}$$

In Appendix 8.B the following approximation for $E\{m_i m_i^H\}$ is derived, based on a result from [23]:

$$E\{m_i m_i^H\} \approx \frac{1-\lambda}{2} \mathcal{V} \mathcal{R}_u^{-1} \tag{8.60}$$

where

$$\mathcal{V} = \mathrm{diag}\{\sigma_1^2, \ldots, \sigma_N^2\} \otimes I_M \tag{8.61}$$

and with

$$\sigma_k^2 = w^{o\,H} b_k - r_{n_k v_k}^H w^o + \sigma_{v_k}^2 - b_k^H R_{u_k}^{-1} b_k \tag{8.62}$$

where

$$b_k = R_{n_k} w^o - r_{n_k v_k} \ . \tag{8.63}$$

It is possible to derive a closed form expression for $\mathcal{M}_k$ defined in (8.59), based on the eigenvalue decomposition $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n = Q\Sigma Q^{-1}$ where Σ is a diagonal matrix with the eigenvalues as its diagonal elements, and where Q contains the corresponding normalized eigenvectors in its columns. We also define the MN-dimensional vector η containing the diagonal elements of Σ^H (the conjugated eigenvalues) in the same order as they appear on the diagonal. In Appendix 8.C, the following closed form expression is derived:

$$\mathcal{M}_k = \mathcal{A}^H \left(\mathcal{M}_{k,2} + \mathcal{M}_{k,2}^H - \mathcal{M}_{k,1}\right) \mathcal{A} \tag{8.64}$$

with

$$\mathcal{M}_{k,1} = Q^{-H} \left(\frac{Q^H \mathcal{E}_k Q}{\mathbb{1}\mathbb{1}^H - \eta\eta^H}\right) Q^{-1} \tag{8.65}$$

$$\mathcal{M}_{k,2} = Q^{-H} \left(I_{MN} - \lambda \Sigma^H\right)^{-1} \left(\frac{Q^H \mathcal{E}_k Q}{\mathbb{1}\mathbb{1}^H - \eta\eta^H}\right) Q^{-1} \tag{8.66}$$

where the double-lined fraction denotes an elementwise division of the matrices in the numerator and denominator (i.e. a Hadamard quotient).

We thus find a closed-form expression for the MSD at node k:

$$\boxed{\text{MSD}_k = \tfrac{1-\lambda}{2}\text{Tr}\left(\mathcal{M}_k \mathcal{V} \mathcal{R}_u^{-1}\right) \ .} \tag{8.67}$$

It is noted that only the matrix $\mathcal{M}_k$ depends on the combiner matrix A, since it is incorporated in the eigenvalue decomposition of $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n$. Note that $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n$ is the same matrix that appears in the stability condition (8.45). Note also that, if the stability condition (8.45) holds, the denominators in (8.65) and (8.66) cannot become zero and $\left(I_{MN} - \lambda\Sigma^H\right)$ cannot become singular, i.e., the algorithm is stable in the mean-square sense.

Again, it is impossible to make general statements about the impact of diffusion on the MSD at a certain node. However, from the Hadamard quotient in (8.65)-(8.66), one can expect that the norm of $\mathcal{M}_k$ will be smaller if the norm of η is small. In many cases, setting the matrix $\mathcal{A} \neq I_{MN}$ will decrease the norm of η (although this is not true in general), and then diffusion indeed has a beneficial influence on the MSD. This means that, if diffusion increases the stability (i.e., the spectral radius of $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n$ decreases), it often also improves the mean-square performance. In section 8.5, we will consider some special cases where it can indeed be mathematically verified that diffusion decreases the infinity norm of η.

8.5 Special Cases

In this section, we consider some special cases where the diffBC-RLS algorithm is guaranteed to be stable, or where it can be mathematically verified that diffusion improves stability of the BC-RLS algorithm, i.e., (compare with (8.45))

$$\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \leq \rho\left(\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \ . \tag{8.68}$$

As mentioned earlier, if (8.68) holds, diffusion often (but not necessarily) also decreases the bias and the MSD of the estimates. It is noted that diffusion in general provides better results (with respect to stability, bias and MSD) due to the non-expanding effect of the combiner matrix A. The beneficial influence of diffusion is therefore not limited to the special cases given below. These merely serve as "motivating" examples where the beneficial influence of diffusion can be theoretically verified.

Remark: Unless stated otherwise, we assume that the combiner matrix A is symmetric, which is required in most cases to make some conclusions. The Metropolis rule (see, e.g., [20]) offers a procedure to select the weights, based on the network topology, that yields a symmetric combiner matrix A.

8.5.1 Invariant Spatial Profile

If the regressor and noise covariance matrices are the same in each node, i.e., $R_{n_k} = R_n$, $R_{u_k} = R_u$, and if each node uses the same estimate $\hat{R}_{n_k} = \hat{R}_n$, for $k \in \{1, \ldots, N\}$, we find that

$$\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n = A \otimes R_u^{-1}\hat{R}_n \; . \tag{8.69}$$

Since the set of eigenvalues of $X \otimes Y$ is equal to the set of all pairwise products between the eigenvalues of X and the eigenvalues of Y, we find that

$$\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) = \rho\left(R_u^{-1}\hat{R}_n\right) \tag{8.70}$$

i.e., the algorithm is stable if and only if the undiffused BC-RLS at a single node is stable. In this case, diffusion has no effect on stability[8]. However, since the eigenvalues of $\mathcal{A}$ are inside the unit circle, many eigenvalues of $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n = A \otimes R_u^{-1}R_n$ will be strictly smaller than the corresponding eigenvalues of $R_u^{-1}R_n$ (and none of the eigenvalues can increase). This means that the norm of η in (8.65) will be smaller than in the undiffused case ($A = I_N$), which mostly results in a smaller MSD. The same holds for the asymptotic bias given in (8.47), since smaller eigenvalues of $\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n$ yield a more contractive or a less expanding mapping $\left(I_{MN} - \mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{-1}$. It is noted that the combiner matrix A does not need to be symmetric to obtain the above results.

8.5.2 2-norm Constraint ($\|\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 < 1$)

If $\|R_{u_k}^{-1}\hat{R}_{n_k}\|_2 < 1$, for $k \in \{1, \ldots, N\}$, where $\|.\|_2$ denotes the matrix 2-norm, the block-diagonal structure of $\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n$ implies that

$$\|\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 < 1 \; . \tag{8.71}$$

Although the condition (8.71) does not imply (8.68), it is an interesting case since stability of the diffBC-RLS algorithm is guaranteed when (8.71) holds. Indeed, we have that

$$\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \leq \|\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 \leq \|\mathcal{A}\|_2\|\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 = \|\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 < 1 \; . \tag{8.72}$$

The first inequality follows from the fact that the spectral radius is the infimum of all induced norms of a matrix (including the two-norm), and the second inequality follows from the fact that the two-norm is sub-multiplicative. Since the two-norm and the spectral radius are the same for symmetric matrices (we assume a symmetric $\mathcal{A}$), we have that $\|\mathcal{A}\|_2 = \rho(\mathcal{A}) = 1$.

[8] This is not surprising, since the invariant spatial profile assumption implies that either all nodes are stable, or none of them are. In the latter case, diffusion adaptation cannot help.

It is noted that $\|\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 < 1$ is satisfied if either the noise $n_{k,i}$ or the regressors $\overline{u}_{k,i}$ are white at each node, and if $\hat{\mathcal{R}}_n = \mathcal{R}_n$ (see subsections 8.5.3 and 8.5.4).

8.5.3 White Noise on Regressors

Assume that we have prior knowledge that $R_{n_k} = \sigma_{n_k}^2 I_M$ and $\hat{R}_{n_k} = \hat{\sigma}_{n_k}^2 I_M$, for $k \in \{1, \ldots, N\}$. Let $\overline{U}_k \overline{\Lambda}_k \overline{U}_k^H$ denote the eigenvalue decomposition of the clean regressor covariance matrix $R_{\overline{u}_k}$, then we obtain

$$R_{u_k}^{-1}\hat{R}_{n_k} = \overline{U}_k \mathrm{diag}\left\{ \frac{\hat{\sigma}_{n_k}^2}{\overline{\lambda}_{k,1} + \sigma_{n_k}^2}, \ldots, \frac{\hat{\sigma}_{n_k}^2}{\overline{\lambda}_{k,M} + \sigma_{n_k}^2} \right\} \overline{U}_k^H \tag{8.73}$$

where $\overline{\lambda}_{k,1} \geq \overline{\lambda}_{k,2} \geq \ldots \geq \overline{\lambda}_{k,M}$ are the diagonal elements of $\overline{\Lambda}_k$.

Note that $\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n$ is symmetric in this case, and therefore

$$\begin{aligned} \rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) &\leq \|\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 \\ &\leq \|\mathcal{A}\|_2 \|\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\|_2 \\ &= \rho\left(\mathcal{A}\right)\rho\left(\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \\ &= \rho\left(\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \end{aligned} \tag{8.74}$$

i.e., (8.68) holds. Furthermore, $\|R_{u_k}^{-1}\hat{R}_{n_k}\|_2 = \frac{\hat{\sigma}_{n_k}^2}{\overline{\lambda}_{k,1}+\sigma_{n_k}^2}$. If $\hat{\sigma}_{n_k}^2 < \overline{\lambda}_{k,1} + \sigma_{n_k}^2$ for each k, i.e., if the noise variances are not significantly overestimated, we know from Subsection 8.5.2 that the diffBC-RLS algorithm is stable.

8.5.4 White Regressors

Assume that we have prior knowledge that $R_{\overline{u}_k} = \sigma_{\overline{u}_k}^2 I_M$, for $k \in \{1, \ldots, N\}$. Furthermore, assume that $\hat{\mathcal{R}}_n = \mathcal{R}_n$, i.e., a good estimate of the noise covariance is available. We thus have that

$$R_{u_k}^{-1}\hat{R}_{n_k} = \left(\sigma_{\overline{u}_k}^2 I_M + R_{n_k}\right)^{-1} R_{n_k} \ . \tag{8.75}$$

Since $(\alpha I + X)^{-1} X = X (\alpha I + X)^{-1}$ for every X and α, it follows that $R_{u_k}^{-1}\hat{R}_{n_k}$ is a symmetric matrix. Therefore, with a similar reasoning as in subsection 8.5.3, we again obtain (8.68). From (8.75), it is also obvious that $\|R_{u_k}^{-1}\hat{R}_{n_k}\|_2 < 1$, and therefore we know from Subsection 8.5.2 that the diffBC-RLS algorithm is stable.

8.6 Simulation Results

In this section, we provide simulation results to compare the performance of the BC-RLS and diffBC-RLS algorithm, and we compare the simulation results with the theoretical results of Section 8.4.

The measurements $d_k(i)$ were generated according to (8.1), and the clean regressors $\overline{u}_{k,i}$ were chosen Gaussian i.i.d. with a covariance matrix $R_{\overline{u}_k} = Q_1 \mathrm{diag}\{5,4,3,2,1\} Q_1^H$, where Q_1 is a random unitary matrix. The stacked vectors of the regressor noises and the measurement noises $\overline{n}_{k,i} = [n_{k,i}\ v_k(i)]$ were also chosen Gaussian i.i.d. with a random covariance matrix $E\{\overline{\mathbf{n}}_{k,i}^H \overline{\mathbf{n}}_{k,i}\} = s_k Q_2 \mathrm{diag}\{2, 1.8, 1.6, 1.4, 1.2, 1\} Q_2^H$, where Q_2 is again a random unitary matrix, and s_k is a random scalar drawn from a uniform distribution in the interval $[0.1, 1]$. Note that, due to the scaling with s_k, this is not an invariant spatial profile, since there is a different SNR in each node. The network had $N = 20$ nodes, and the topology was chosen randomly with a connectivity of 5 links per node on average. The size of the unknown vector w^o was $M = 5$, and the combiner matrix A was constructed using Metropolis weights. All results are averaged over 200 experiments.

8.6.1 Bias

In this subsection, we add some errors to the noise estimates $\hat{R}_{n_k} = R_{n_k} + \Delta R_{n_k}$ and $\hat{r}_{n_k v_k} = r_{n_k v_k} + \Delta r_{n_k v_k}$ to investigate the effect on the bias of the BC-RLS and diffBC-RLS estimates. The errors were modelled as

$$\Delta R_{n_k} = \sqrt{p|R_{n_k}|} \odot R_k \tag{8.76}$$

$$\Delta r_{n_k v_k} = \sqrt{p|r_{n_k v_k}|} \odot r_k \tag{8.77}$$

where $\odot$ denotes a Hadamard product (elementwise multiplication), the operator $|.|$ denotes an elementwise absolute value operator, and p is a positive scalar variable that is used to increase the error. The entries of the $M \times M$ matrix R_k and the M-dimensional vector were independently drawn from a normal distribution (i.e. with zero mean and unity variance).

Fig. 8.1 shows $\|E\{\tilde{w}_i\}\|$, i.e. the norm of the stacked bias, as a function of p, both for BC-RLS (without cooperation) and diffBC-RLS in steady state (with $\lambda = 0.999$). We see that the theoretical results (8.47) match very well with the simulated results. Furthermore, we observe that diffusion indeed significantly decreases the asymptotic bias of the BC-RLS estimates. Fig. 8.2 shows the entries of the stacked bias vector $E\{\tilde{w}_i\}$ for $p = 0.2$, again demonstrating that the theoretical results are very accurate.

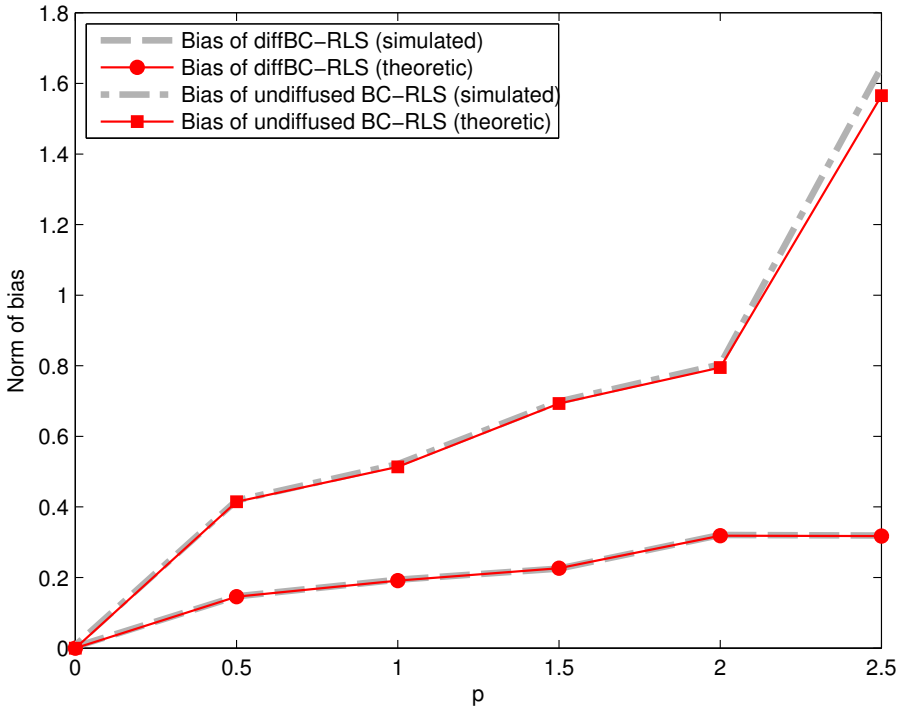


Figure 8.1: The norm of the stacked asymptotic bias as a function of p, using $\lambda = 0.999$.

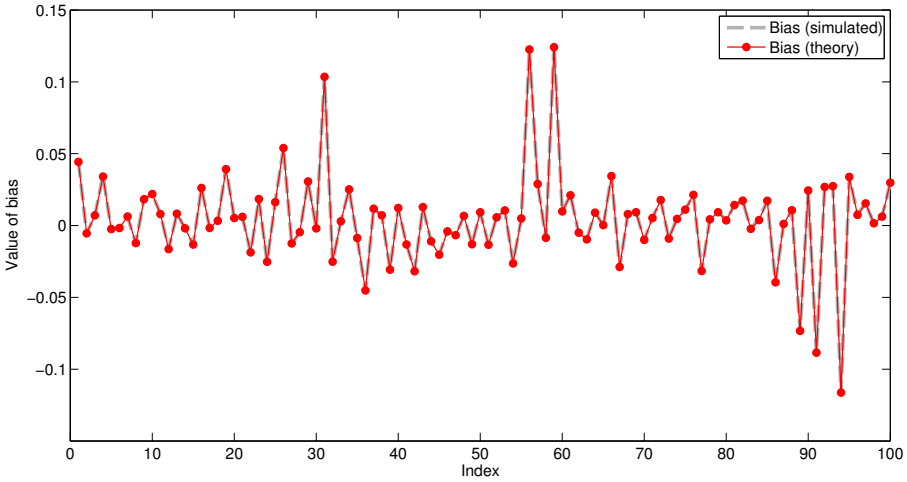


Figure 8.2: The entries of $E\{\tilde{w}_i\}$ in steady state for $p = 0.2$, using $\lambda = 0.999$.

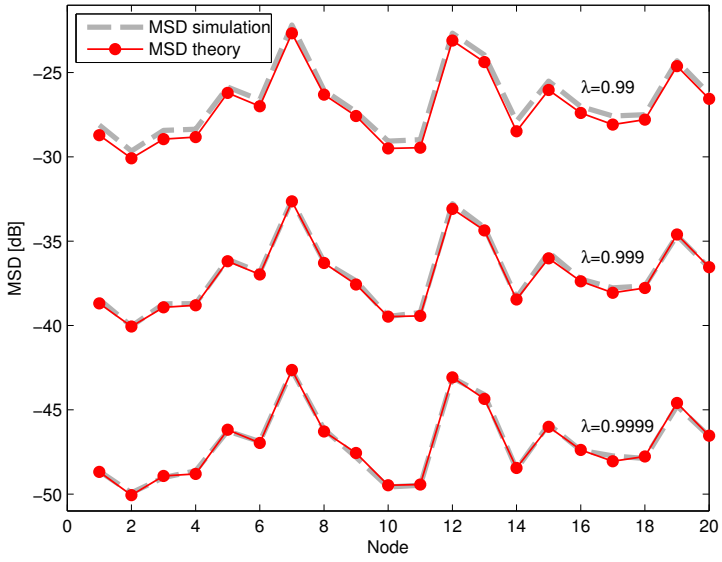


Figure 8.3: The steady-state MSD values in each node, for different values of λ.

8.6.2 MSD

In Fig. 8.3, we show the MSD in the different nodes, for several values of λ. We see that the theoretical results (8.67) match very well with the simulated results, especially when λ is close to unity. However, when λ is too small (e.g. $\lambda = 0.9$), the algorithm becomes unstable in some iterations. The reason for this is the fact that the approximation (8.30) in Assumption 4 becomes invalid. As mentioned at the end of subsection 8.4.2, the algorithm may become unstable at some iterations due to Jensen's inequality, even though the stability condition (8.45) is satisfied. This is demonstrated[9] in Fig. 8.4. Since the theoretical analysis does not incorporate this effect, there is no match between the theoretical results and the simulation results for this case.

In Fig. 8.5, the MSD is plotted as a function of the number of observations[10] both for BC-RLS (without cooperation) and diffBC-RLS. It is observed that the MSD is significantly smaller when the nodes diffuse their estimations.

[9]There is no steady-state in this case. The plotted MSD is the time average of the last 3000 iterations of the algorithm.

[10]Since the norm of $P_{k,i}$ can be very large in the beginning due to a small regularization parameter δ, the recursion usually starts to diverge until i becomes large enough. Therefore, to obtain an intelligible figure, we initialized the matrices $P_{k,0}$ with $(1-\lambda)R_{u_k}^{-1}$, i.e., the convergence of the RLS part is removed.

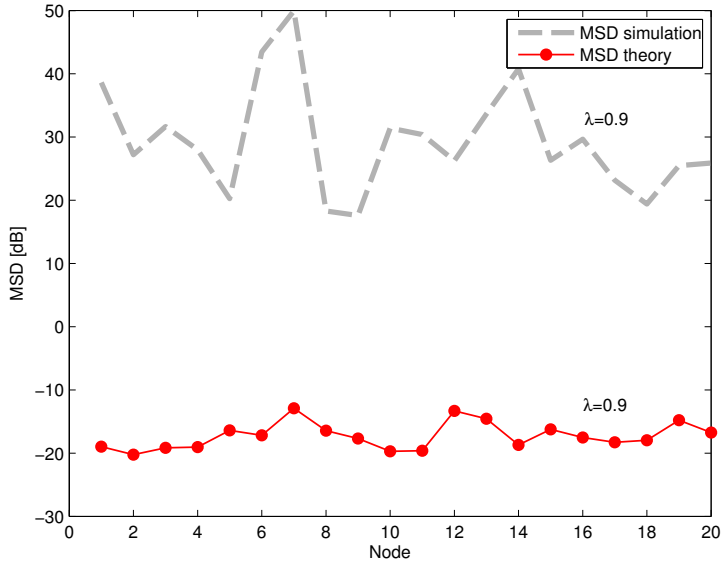


Figure 8.4: The average MSD values in each node for $\lambda = 0.9$.

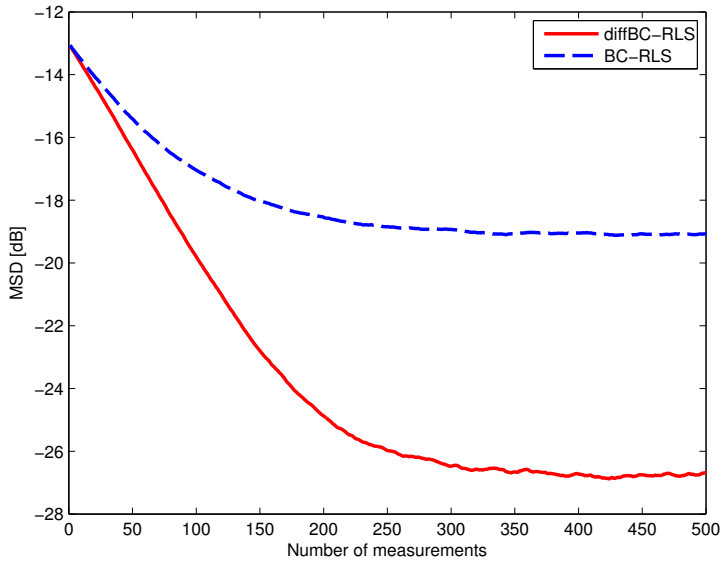


Figure 8.5: MSD curves of the BC-RLS algorithm and the diffBC-RLS algorithm with $\lambda = 0.99$.

8.7 Conclusions

We have addressed the problem of distributed least-squares estimation over adaptive networks when there is stationary additive colored noise on both the regressors and the output response, which results in a bias on the least-squares estimates. Assuming that the noise covariance can be estimated (or is known a-priori), we have proposed a bias-compensated recursive least-squares (BC-RLS) algorithm. This bias compensation significantly increases the variance of the local estimates, and errors in the noise covariance estimates may still result in a significant residual bias. By applying diffusion, i.e., letting neighboring nodes combine their local estimates, the variance (or MSD) and the residual bias can be significantly reduced. The latter is referred to as diffusion BC-RLS (diffBC-RLS). We have derived a necessary and sufficient condition for mean-square stability of the algorithm, under some mild assumptions. Furthermore, we have derived closed-form expressions for the residual bias and the MSD, which match well with the simulation results if the forgetting factor is close to unity. We have also considered some special cases where the stability improvement of diffBC-RLS over BC-RLS can be mathematically verified.

A possible application of the diffBC-RLS algorithm is the AR analysis of a speech signal in a wireless sensor networks, with microphone nodes that are spatially distributed over an environment. RLS has been demonstrated to be able to track speech AR parameters [24] in environments with limited noise. For noisy recordings, bias compensation is crucial for AR analysis of speech signals. However, this bias compensation usually severely increases the variance of the estimated speech AR coefficients (often resulting in unstable behavior), due to the ill-conditioned nature of the speech covariance matrix in certain speech phonemes. Since all the microphones observe the same speech signal (possibly at a different SNR), the stability of the algorithm and the variance of the estimates can be greatly improved by applying diffusion adaptation.

Acknowledgements

The first author would like to thank all the co-workers in the Adaptive Systems Laboratory at UCLA for the fruitful discussions regarding several aspects of this manuscript.

Appendix

8.A Derivation of Expression (8.57)

We first consider the case where $m \leq n$. Because of the steady-state assumption, only the difference $t = n - m$ is important, i.e. $E\{\boldsymbol{m}_{i-m}\boldsymbol{m}_{i-n}^H\} = E\{\boldsymbol{m}_i\boldsymbol{m}_{i-t}^H\}$. Because of the spatial independence assumption and the fact that $E\{\boldsymbol{m}_i\} = 0$ (this follows from (8.40)), $E\{\boldsymbol{m}_i\boldsymbol{m}_{i-t}^H\}$ will be a block-diagonal matrix (note that there is no diffusion on the local RLS estimates). Therefore, we can focus on a single block corresponding to node k, i.e. the submatrix $E\{m_{k,i}m_{k,i-t}^H\}$ where $m_{k,i} \triangleq w_k^{\mathrm{LS}} - \hat{w}_{k,i}$.

From the RLS update (8.21), we find that

$$\begin{aligned}
E\{m_{k,i}m_{k,i-t}^H\} &= E\{m_{k,i-1}m_{k,i-t}^H\} + P_{k,i}R_{u_k}E\{\hat{w}_{k,i-1}m_{k,i-t}^H\} \\
&= E\{m_{k,i-1}m_{k,i-t}^H\} + (1-\lambda)E\{\hat{w}_{k,i-1}m_{k,i-t}^H\} \\
&= E\{m_{k,i-1}m_{k,i-t}^H\} + (\lambda-1)E\{-\hat{w}_{k,i-1}m_{k,i-t}^H\} \\
&\quad + (\lambda-1)E\{w_k^{\mathrm{LS}}m_{k,i-t}^H\} \\
&= E\{m_{k,i-1}m_{k,i-t}^H\} + (\lambda-1)E\{m_{k,i-1}m_{k,i-t}^H\} \\
&= \lambda E\{m_{k,i-1}m_{k,i-t}^H\} \qquad (8.78)
\end{aligned}$$

where we used Assumption 4 in the second step, and the fact that $E\{m_{k,i}\} = 0$ (for $i \to \infty$) in the first and third step. By expanding the recursion (8.78), and because of the steady-state assumption, we find that

$$E\{m_{k,i}m_{k,i-t}^H\} = \lambda^t E\{m_{k,i}m_{k,i}^H\} \qquad (8.79)$$

The case where $m \geq n$ or $t \leq 0$ immediately follows from (8.79) by using a substitution $j = i - t$:

$$\begin{aligned}
E\{m_{k,i}m_{k,i-t}^H\} &= E\{m_{k,j+t}m_{k,j}^H\} \\
&= E\{m_{k,j}m_{k,j+t}^H\}^H \\
&= \lambda^{-t}E\{m_{k,j}m_{k,j}^H\} \\
&= \lambda^{-t}E\{m_{k,i}m_{k,i}^H\}\,. \qquad (8.80)
\end{aligned}$$

Both cases can be handled simultaneously by using an absolute value in the exponent of λ, which straightforwardly results in (8.57).

8.B Derivation of Expression (8.60)-(8.63)

Because of the spatial independence assumption, and the fact that $E\{\boldsymbol{m}_i\} = 0$ (this follows from (8.40)), $E\{\boldsymbol{m}_i \boldsymbol{m}_i^H\}$ will be a block-diagonal matrix (note that there is no diffusion on the local RLS estimates). Therefore, we can focus on a single block corresponding to node k, i.e. the submatrix $E\{m_{k,i} m_{k,i}^H\}$ where $m_{k,i} \triangleq w_k^{\mathrm{LS}} - \hat{w}_{k,i}$.

For an RLS algorithm that observes regressors u_i and corresponding $d(i) = u_i \phi^o + e(i)$ where $e(i)$ is zero-mean noise with variance σ_e^2, the following approximation holds for $i \to \infty$ [23]:

$$E\{(\phi^o - w_i)\,(\phi^o - w_i)^H\} \approx \frac{1-\lambda}{2}\sigma_e^2 E\{u_i^H u_i\}\;. \tag{8.81}$$

It is noted that the variable $\hat{w}_{k,i}$ in the diffBC-RLS algorithm is an unbiased estimate of w_k^{LS}. Therefore, we can view $\hat{w}_{k,i}$ as the outcome of an RLS algorithm that observes regressor $u_{k,i} = \overline{u}_{k,i} + n_{k,i}$ and a corresponding

$$d_k(i) = u_{k,i} w_k^{\mathrm{LS}} + e_k(i)\;. \tag{8.82}$$

Since the observations $d_k(i)$ are actually equal to

$$d_k(i) = \overline{u}_{k,i} w^o + v_k(i) \tag{8.83}$$

we have to rewrite it to the form (8.82). With (8.7), and setting (8.82) equal to (8.83), some straightforward algebra yields

$$e_k(i) = v_k(i) - n_{k,i} w^o + u_{k,i} R_{u_k}^{-1}\left(R_{n_k} w^o - r_{n_k,v_k}\right)\;. \tag{8.84}$$

Using the approximation (8.81), we find that

$$E\{m_{k,i} m_{k,i}^H\} \approx \frac{1-\lambda}{2}\sigma_{e_k}^2 R_{u_k} \tag{8.85}$$

where $\sigma_{e_k}^2 = E\{|e_k(i)|^2\}$. We will now derive an expression for $\sigma_{e_k}^2$. For the sake of an easy exposition, we omit the subscript k and the time index i in the sequel. We introduce the notation $b \triangleq R_n w^o - r_{nv}$. Observe that

$$E\{(v - n w^o)\, u^H\} = r_{nv} - R_n w^o = -b\;. \tag{8.86}$$

With this, we find that

$$E\{|e|^2\} = E\{|v - n w^o|^2\} + E\{u R_u^{-1} b b^H R_u^{-1} u^H\} - 2 b^H R_u^{-1} b\;. \tag{8.87}$$

The first term is equal to

$$E\{|v - n w^o|^2\} = w^{o\,*} R_{n_k} w^o - r_{n_k v_k}^H w^o - w^{o\,*} r_{n_k v_k} + \sigma_v^2\;. \tag{8.88}$$

Using the trace operator, we find that the second term is equal to

$$E\{u R_u^{-1} b b^H R_u^{-1} u^H\} = \mathrm{Tr}\left(R_u^{-1} b b^H R_u^{-1} E\{u^H u\}\right) = b^H R_u^{-1} b\;. \tag{8.89}$$

Inserting (8.87)-(8.89) in (8.85), and repeating this for each node, we find expression (8.60)-(8.63).

8.C Derivation of Expression (8.65)-(8.66)

We introduce the notation

$$\mathcal{D} \triangleq \mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n \tag{8.90}$$

where the eigenvalue decomposition of $\mathcal{D}$ is given by $\mathcal{D} = Q\Sigma Q^{-1}$. The matrix $\mathcal{M}_k$ defined in (8.59) can then be rewritten as

$$\mathcal{M}_k = \mathcal{A}^H \left(\sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \lambda^{|m-n|} \mathcal{D}^{H\,m} \mathcal{E}_k \mathcal{D}^n \right) \mathcal{A} \,. \tag{8.91}$$

First, observe that

$$\begin{aligned} \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \lambda^{|m-n|} \mathcal{D}^{H\,m} \mathcal{E}_k \mathcal{D}^n &= \sum_{m=0}^{\infty} \mathcal{D}^{H\,m} \mathcal{E}_k \mathcal{D}^m \\ &\quad + \sum_{n=1}^{\infty} \lambda^n \sum_{m=n}^{\infty} \mathcal{D}^{H\,m} \mathcal{E}_k \mathcal{D}^{m-n} \\ &\quad + \sum_{n=1}^{\infty} \lambda^n \sum_{m=n}^{\infty} \mathcal{D}^{H\,m-n} \mathcal{E}_k \mathcal{D}^m \,. \end{aligned} \tag{8.92}$$

We will first focus on the second term of (8.92), which we denote as T_2. Note that the last term is the conjugate transpose of T_2. Using the eigenvalue decomposition of $\mathcal{D}$, we find that

$$\begin{aligned} T_2 &= Q^{-H} \left(\sum_{n=1}^{\infty} \lambda^n \sum_{m=n}^{\infty} \Sigma^{H\,m} Q^H \mathcal{E}_k Q \Sigma^{m-n} \right) Q^{-1} \\ &= Q^{-H} \left(\sum_{n=1}^{\infty} \lambda^n \Sigma^{H\,n} \sum_{m=n}^{\infty} \Sigma^{H\,m-n} Q^H \mathcal{E}_k Q \Sigma^{m-n} \right) Q^{-1} \\ &= Q^{-H} \left(\sum_{n=1}^{\infty} \lambda^n \Sigma^{H\,n} \left(\left(Q^H \mathcal{E}_k Q \right) \odot E \right) \right) Q^{-1} \end{aligned} \tag{8.93}$$

where $\odot$ denotes a Hadamard product (elementwise multiplication), and with E a matrix where the entry on the i-th row and the j-th column is given by

$$E_{ij} = \sum_{m=n}^{\infty} \eta_i^{m-n} \eta_j^{H\,m-n} \tag{8.94}$$

where η_i is the i-th diagonal element of Σ^H (the conjugate of the i-th eigenvalue). If the stability condition (8.45) holds, then $|\eta_i \eta_j^H| < 1, \forall i, j \in \{1, \ldots, M\}$. Based on a Taylor expansion similar to (8.46), but for the complex scalar case, we find that

$$E_{ij} = \frac{1}{1 - \eta_i \eta_j^H} \,. \tag{8.95}$$

Using this, (8.93) can be rewritten as

$$T_2 = Q^{-H} \left(\sum_{n=1}^{\infty} \lambda^n \Sigma^{H\,n} \right) \frac{Q^H \mathcal{E}_k Q}{\mathbb{1}\mathbb{1}^H - \eta\eta^H} Q^{-1} \tag{8.96}$$

where the MN-dimensional vector $\eta = \text{col}\{\eta_1, \ldots, \eta_{MN}\}$ contains the diagonal elements of Σ^H (the conjugated eigenvalues) in the same order as they appear on the diagonal.

Based on the expansion (8.46), we can rewrite (8.96) as

$$T_2 = Q^{-H} \left(\left(I_{MN} - \lambda \Sigma^H \right)^{-1} - I_{MN} \right) \frac{Q^H \mathcal{E}_k Q}{\mathbb{1}\mathbb{1}^H - \eta\eta^H} Q^{-1} \,. \tag{8.97}$$

A closed-form expression for the first term of (8.92) can be derived in a similar way, which results in

$$T_1 = Q^{-H} \frac{Q^H \mathcal{E}_k Q}{\mathbb{1}\mathbb{1}^H - \eta\eta^H} Q^{-1} \,. \tag{8.98}$$

Since $T_1 + T_2 + T_2^H = \mathcal{M}_{k,2} + \mathcal{M}_{k,2}^H - \mathcal{M}_{k,1}$, we eventually find (8.65)-(8.66).

Bibliography

[1] C. E. Davila, "An efficient recursive total least squares algorithm for FIR adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 42, pp. 267–280, 1994.

[2] D.-Z. Feng, Z. Bao, and L.-C. Jiao, "Total least mean squares algorithm," *IEEE Transactions on Signal Processing*, vol. 46, no. 8, pp. 2122 –2130, Aug. 1998.

[3] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, pp. 2320–2330, May 2011.

[4] H. So, "Modified LMS algorithm for unbiased impulse response estimation in non-stationary noise," *Electronics Letters*, vol. 35, pp. 791–792, May 1999.

[5] D. Feng, Z. Bao, and X. Zhang, "Modified RLS algorithm for unbiased estimation of fir system with input and output noise," *Electronics Letters*, vol. 36, no. 3, pp. 273 –274, Feb. 2000.

[6] S. Sagara and K. Wada, "On-line modified least-squares parameter estimation of linear discrete dynamic systems," *International Journal of Control*, vol. 25, pp. 329–343, 1977.

[7] W. X. Zheng, "A least-squares based algorithm for fir filtering with noisy data," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2003, pp. IV–444 – IV–447 vol.4.

[8] L. Jia, C. Jin, and K. Wada, "On bias compensated recursive least-squares algorithm for FIR adaptive filtering," in *Proc. of IFAC Workshop on Adaptation and Learning in Control and Signal Processing*, 2001, pp. 347–352.

[9] L. Jia, R. Tao, Y. Wang, and K. Wada, "Forward/backward prediction solution for adaptive noisy FIR filtering," *Science in China: Information Sciences*, vol. 52, pp. 1007–1014, 2009.

[10] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, 2007, pp. III–917 –III–920.

[11] ——, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, July 2008.

[12] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1035–1048, March 2010.

[13] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[14] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 981030, 19 pages, 2009. doi:10.1155/2009/981030.

[15] ——, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, 2009.

[16] F. Cattivelli and A. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," *Automatic Control, IEEE Transactions on*, vol. 55, no. 9, pp. 2069 –2084, 2010.

[17] A. Bertrand, M. Moonen, and A. H. Sayed, "Diffusion-based bias-compensated RLS for distributed estimation over adaptive sensor networks," *Submitted for publication*, 2011.

[18] A. H. Sayed, *Adaptive Filters*. NJ: John Wiley & Sons, 2008.

[19] D. W. Marquardt, "Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation," *Technometrics*, vol. 12, no. 3, pp. pp. 591–612, 1970.

[20] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*. Piscataway, NJ, USA: IEEE Press, 2005, pp. 63–70.

[21] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[22] B. Widrow and S. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[23] L. Guo, L. Ljung, and P. Priouret, "Performance analysis of the forgetting factor RLS algorithm," *International Journal of Adaptive Control and Signal Processing*, vol. 7, pp. 525–537, 1993.

[24] C. Papaodysseus, G. Roussopoulos, and A. Panagopoulos, "Using a fast RLS adaptive algorithm for efficient speech processing," *Mathematics and Computers in Simulation*, vol. 68, no. 2, pp. 105 – 113, 2005.

Part IV

Supporting Techniques for Signal Estimation

Chapter 9

Blind Separation of Non-Negative Source Signals

Blind separation of non-negative source signals using multiplicative updates and subspace projection

Alexander Bertrand and Marc Moonen

Published in *Signal Processing*, vol. 90, no.10, pp. 2877 - 2890, Oct. 2010.

Contributions of first author

- literature study
- co-development of the M-NICA algorithm
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

In this paper, we consider a noise-free blind source separation problem with independent non-negative source signals, also known as non-negative independent component analysis (NICA). We assume that the source signals are well-grounded, which means that they have a non-vanishing pdf in a positive neighborhood of zero. We propose a novel algorithm, referred to as multiplicative NICA (M-NICA), which uses multiplicative updates together with a subspace projection based correction step to reconstruct the original source signals from the observed linear mixtures, and which is based only on second order statistics. A multiplicative update has the facilitating property that it preserves non-negativity, and does not depend on a user-defined learning rate, as opposed to gradient based updates such as in the non-negative PCA (NPCA) algorithm. We provide batch mode simulations of M-NICA and compare its performance to NPCA, for different types of signals. It is observed that M-NICA generally yields a better unmixing accuracy, but converges slower than NPCA. Especially when the amount of data samples is small, M-NICA signficantly outperforms NPCA. Furthermore, a sliding window implementation of both algorithms is described and simulated, where M-NICA is observed to provide the best results.

9.1 Introduction

Assume that we observe a set of instantaneous linear mixtures of mutually independent source signals. The goal of independent component analysis (ICA) is then to reconstruct the original source signals from the observed mixtures. This problem is widely studied in literature (see [1, 2] for a survey), usually under the general assumption that the source signals are nongaussian and that the mixing matrix is full rank. However, if some prior knowledge on the source signals is available, this knowledge may be exploited to design more efficient unmixing algorithms. In this paper, we consider an ICA problem with *non-negative* sources, i.e. we collect observations of a J-channel signal $\mathbf{y}$ that satisfies

$$\mathbf{y} = \mathbf{A}\mathbf{s} \tag{9.1}$$

where $\mathbf{s} = [s_1 \dots s_N]^T$ is a vector of N mutually independent source signals with $s_n \geq 0$, $\forall\, n \in \{1 \dots N\}$, and where $\mathbf{A}$ is an unknown $J \times N$ mixing matrix with $J \geq N$. In [3], this problem is referred to as the *non-negative independent component analysis* (NICA) problem. Non-negativity arises in many practical problems, e.g. source activity detection [4], unmixing spectral data [5], hyperspectral imaging [6, 7], chemistry [8], environmetrics [9], music transcription [10], etc.

A closely related problem is 'non-negative matrix factorization' (NMF) [11, 12], in which a non-negative matrix is factorized in two smaller non-negative matrices. This corresponds to the case where the mixing matrix $\mathbf{A}$ is also assumed to be non-negative. However, NMF does not take independence of the sources into account, and therefore NMF algorithms often yield suboptimal results when applied to the NICA problem.

By making additional assumptions on the source signals, several algorithms are proposed to solve the NICA problem [3, 13, 14]. In this paper, we add the assumption that the sources are well-grounded, as also done in [3]. This means that all sources have a non-zero pdf in any positive neighborhood of zero, i.e. $\forall\, \delta > 0$: $Pr(s_n < \delta) > 0$, for all source signals s_n, $n = 1 \ldots N$. Well-groundedness of the sources is a weaker assumption than the locally-dominant assumption[1]in [13, 14], and it is often satisfied in practice, e.g. when the sources have an on-off behavior or when the source signals are sparse. The locally-dominant assumption is more easily violated, especially for short time windows. We will consider two different algorithms to solve the NICA problem with well-grounded sources: the non-negative PCA algorithm (NPCA), which is introduced in [3], and the multiplicative NICA algorithm (M-NICA), which is a novel approach to tackle the NICA problem.

The NPCA algorithm [3] first decorrelates the data by applying a whitening procedure without taking the non-negativity into account. In a second step, the algorithm computes a rotation matrix that restores the non-negativity of the data. This is done by means of a gradient-descent algorithm with additional correction steps to preserve orthogonality. The learning rate of the NPCA algorithm is a critical parameter to obtain satisfying results. If the learning rate is chosen too small, the algorithm can have extremely slow convergence. On the other hand, if the learning rate is too high, it is possible that the NPCA algorithm does not converge at all.

The M-NICA algorithm, on the other hand, decorrelates the data while at the same time maintaining non-negativity, by means of a multiplicative update step. Multiplicative updating is a popular technique to solve non-negative optimization problems, e.g. [12, 15]. Since a multiplicative update results in data that is not in the original signal subspace, it requires a correction step based on a subspace projection to restore the original signal subspace. The M-NICA algorithm has the facilitating property that it does not depend on a user-defined learning rate, as opposed to the NPCA algorithm. This is particularly relevant in applications for which a step-size search is impossible or undesirable.

NPCA and M-NICA have similar computational complexity. We will compare the performance of both algorithms by means of simulations with different types

[1]The locally-dominant assumption states that for each source s_j in a set of N source signals $\{s_1, \ldots, s_N\}$, there is a sample time t_j in the data set such that $s_j[t_j] \neq 0$ and $s_i[t_j] = 0$ for all $i \neq j$.

of signals. As will be demonstrated, the convergence speed and the unmixing accuracy of both algorithms heavily depends on the type of signals involved. By averaging over multiple experiments, it is observed that M-NICA generally provides a better unmixing accuracy, but at a slower convergence rate than NPCA. The difference between the unmixing accuracy of M-NICA and NPCA becomes more significant in cases where the amount of available data samples is small, where the former is observed to provide the best results. Also in an adaptive sliding window implementation, M-NICA clearly outperforms NPCA in terms of unmixing accuracy, at a slightly slower adaptation speed.

The outline of the paper is as follows. In Section 9.2, the NPCA algorithm is briefly reviewed. The batch mode M-NICA algorithm is described in Section 9.3. In Section 9.4, a sliding window implementation of the M-NICA algorithm is described. Batch mode simulations of M-NICA and NPCA are provided in Section 9.5. The performance of the sliding window implementations of M-NICA and NPCA are analyzed in Section 9.6. Conclusions are given in Section 9.7.

9.2 Non-Negative PCA (NPCA)

In [16], the following theorem is proven:

Theorem 9.1 *Let* $\mathbf{s}$ *be an* N*-dimensional vector of non-negative and well-grounded mutually independent source signals with unit variance, and let* $\mathbf{z} = \mathbf{U}\mathbf{s}$ *be an orthonormal rotation of* $\mathbf{s}$ *where* $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}_N$*, with* $\mathbf{I}_N$ *denoting the* $N \times N$ *identity matrix. Then* $\mathbf{z}$ *is a permutation of* $\mathbf{s}$ *if and only if the signals in* $\mathbf{z}$ *are non-negative with probability 1.*

This theorem states that any orthogonal mixture of well-grounded mutually independent non-negative sources, that preserves the non-negativity, must be a permutation of the sources. It is noted that, although the well-groundedness of the source signals is not explicitly exploited in the algorithms described in the sequel, it is a crucial assumption. The intuition behind this is that, if the source signals are well-grounded, there is only one possible rotation to completely fit the rectangular (decorrelated) data cloud in the positive orthant [16].

In [3], Theorem 9.1 is used to derive the non-negative PCA algorithm (NPCA), which is able to solve NICA problems with well-grounded sources. The algorithm uses only second order statistics, which makes it very efficient compared to ICA algorithms that use higher order statistics. The outline of the NPCA algorithm is as follows (here described in batch mode):

1. Let $\mathbf{C}_y = E\{(\mathbf{y} - \overline{\mathbf{y}})(\mathbf{y} - \overline{\mathbf{y}})^T\}$, where $E\{.\}$ denotes the expectation

operator and where $\overline{\mathbf{y}} = E\{\mathbf{y}\}$. Compute the eigenvalue decomposition of the covariance matrix $\mathbf{C}_y$, i.e. $\mathbf{C}_y = \mathbf{E}\mathbf{D}\mathbf{E}^T$ with $\mathbf{D}$ a diagonal matrix containing the eigenvalues of $\mathbf{C}_y$ on its diagonal, and with $\mathbf{E}$ containing the corresponding eigenvectors in its columns. Since $\mathbf{C}_y$ is a rank N matrix, we can write $\mathbf{C}_y = \overline{\mathbf{E}}\,\overline{\mathbf{D}}\,\overline{\mathbf{E}}^T$, with $\overline{\mathbf{D}}$ an $N \times N$ diagonal matrix, containing the N non-zero eigenvalues of $\mathbf{C}_y$ on its diagonal, and with $\overline{\mathbf{E}}$ the $J \times N$ matrix containing the corresponding eigenvectors.

2. Whiten the signal $\mathbf{y}$ with a whitening matrix[2]

$$\mathbf{V} = \overline{\mathbf{D}}^{-\frac{1}{2}}\overline{\mathbf{E}}^T \tag{9.2}$$

yielding the whitened compressed signal $\mathbf{v} = \mathbf{V}\mathbf{y}$.

3. Assume w.l.o.g. that the sources s_n, $n = 1 \dots N$, have unit variance[3], such that $\mathbf{C}_s = E\{(\mathbf{s} - \overline{\mathbf{s}})(\mathbf{s} - \overline{\mathbf{s}})^T\} = \mathbf{I}_N$. Then the matrix $\mathbf{Z} = \mathbf{V}\mathbf{A}$ is orthogonal, since $\mathbf{Z}\mathbf{Z}^T = \mathbf{V}\mathbf{A}\mathbf{A}^T\mathbf{V}^T = \mathbf{V}\mathbf{C}_s\mathbf{A}^T\mathbf{V}^T = \mathbf{V}\mathbf{C}_y\mathbf{V}^T = \mathbf{I}_N$. According to Theorem 9.1, it is then sufficient to find an orthogonal matrix $\mathbf{W}$ such that $\mathbf{z} = \mathbf{W}\mathbf{v} = \mathbf{W}\mathbf{Z}\mathbf{s}$ is non-negative with probability 1. This matrix $\mathbf{W}$ can be computed by means of the following learning rule:

$$\mathbf{W}^{\text{temp}} = \mathbf{W}^i - \eta\mathbf{M}^i\mathbf{W}^i \tag{9.3}$$

$$\mathbf{W}^{i+1} = \left(\mathbf{W}^{\text{temp}}\mathbf{W}^{\text{temp}\,T}\right)^{-\frac{1}{2}}\mathbf{W}^{\text{temp}} \tag{9.4}$$

with

$$\mathbf{M}^i = E\{f(\mathbf{z}^i)\mathbf{z}^{i\,T} - \mathbf{z}^i f(\mathbf{z}^{i\,T})\} \tag{9.5}$$

where $\mathbf{z}^i = \mathbf{W}^i\mathbf{v}$, $f(z_n) = \min(0, z_n)$ and with η denoting a positive learning rate.

Since (9.3) does not enforce orthogonality of $\mathbf{W}$, the correction step (9.4) is added to guarantee orthogonality of $\mathbf{W}$. Let $\mathbf{y}[k]$ denote the observation of $\mathbf{y}$ at time k, and let M denote the number of observations of $\mathbf{y}$. Then the expected value in (9.5) can be computed by simple averaging over the M transformed observations $\mathbf{z}^i[k]$, $k = 1 \dots M$. Assuming that $M \gg N$, then (9.5) is the computationally most expensive step of the NPCA algorithm, yielding an overall complexity of $O(N^2M)$.

It is observed that the learning rate η is a crucial parameter for the algorithm to converge, i.e. its value should be small enough to guarantee convergence. However, a too small η results in a very slow convergence. In [17], an adaptive

[2] In [3], a symmetric whitening matrix was chosen, i.e. $\mathbf{V} = \mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T$. This is however only possible when $\mathbf{y}$ is an N-dimensional vector, i.e. when the mixing matrix $\mathbf{A}$ is square. If $J > N$, the whitening matrix (9.2) performs a dimension reduction, in addition to a decorrelation.

[3] If this is not the case, the source signals can be scaled accordingly, yielding a reciprocal scaling of the columns of the mixing matrix $\mathbf{A}$

strategy is proposed to update η. Although convergence can be enforced in this way, the strategy is observed to yield rather conservative learning rates. It remains unclear how an optimal value for η can be chosen automatically to provide a fast convergence.

9.3 Multiplicative NICA (M-NICA)

In this section, we present a new algorithm to solve the NICA problem with well-grounded sources. It is based on the following corollary, which follows straightforwardly from Theorem 9.1:

Corollary 9.2 *Let* $\mathbf{s}$ *be an* N*-dimensional vector of non-negative and well-grounded mutually independent source signals, and let* $\mathbf{y} = \mathbf{A}\mathbf{s}$ *with* $\mathbf{A}$ *a full column rank* $J \times N$ *mixing matrix. Let* $\mathbf{z} = \mathbf{K}\mathbf{y}$ *where* $\mathbf{K}$ *is a* $N \times J$ *unmixing matrix. Then* $\mathbf{z}$ *is a permutation of* $\mathbf{s}$ *if and only if the signals in* $\mathbf{z}$ *are mutually uncorrelated and non-negative with probability 1.*

Proof : We only prove the '$\Leftarrow$' direction, since the '$\Rightarrow$' direction is trivially proved. We thus assume that $\mathbf{z}$ is non-negative and that its signals are mutually uncorrelated, i.e.

$$E\{(\mathbf{z} - \overline{\mathbf{z}})(\mathbf{z} - \overline{\mathbf{z}})^T\} = \mathbf{I}_N \ . \tag{9.6}$$

Since $\mathbf{z} = \mathbf{K}\mathbf{y}$ and $\mathbf{y} = \mathbf{A}\mathbf{s}$, expression (9.6) can be rewritten as

$$\mathbf{K}\mathbf{A}E\{(\mathbf{s} - \overline{\mathbf{s}})(\mathbf{s} - \overline{\mathbf{s}})^T\}\mathbf{A}^T\mathbf{K}^T = \mathbf{I}_N \ . \tag{9.7}$$

Assume w.l.o.g. that the source signals in $\mathbf{s}$ have unit variance. Since these source signals are mutually independent, they are uncorrelated, and therefore (9.7) becomes

$$\mathbf{U}\mathbf{U}^T = \mathbf{I}_N \tag{9.8}$$

where $\mathbf{U} = \mathbf{K}\mathbf{A}$. Expression (9.8) shows that $\mathbf{U}$ is a $N \times N$ orthogonal matrix. Since $\mathbf{z}$ is non-negative and $\mathbf{z} = \mathbf{U}\mathbf{s}$, Theorem 9.1 shows that $\mathbf{z}$ is a permutation of $\mathbf{s}$.

□

To solve the NICA problem (9.1), it is thus sufficient to find an $N \times J$ unmixing matrix $\mathbf{K}$ that results in N non-negative uncorrelated signals. Notice that the first step of the NPCA algorithm decorrelates the data by applying a straightforward whitening procedure without taking the non-negativity into account. In the second step, the algorithm computes a rotation matrix that restores the non-negativity of the data, while preserving the decorrelation. In the M-NICA

algorithm described infra, we will decorrelate the data while preserving the non-negativity. This has several advantages. Since we use a multiplicative update, the algorithm does not require any user-defined learning rate. Furthermore, since we explicitly minimize the correlation under non-negativity constraints, the algorithm is more robust than NPCA when using small sample sets (as will be demonstrated in Section 9.5.4). For the sake of an easy exposition, we will first describe the M-NICA algorithm in batch mode. A sliding window algorithm will be described in Section 9.4.

9.3.1 Multiplicative Decorrelation with Subspace Projection

Assume we collect a $J \times M$ data matrix $\mathbf{Y}$ that contains M observations $\mathbf{y}[k]$, $k = 1 \dots M$, in its columns. We will try to find an unmixing matrix $\mathbf{K}$ such that the rows of the $N \times M$ matrix $\mathbf{S} = \mathbf{KY}$ are uncorrelated and only contain non-negative values. Notice that $\mathbf{S}$ does not necessarily contain the samples $\mathbf{s}[k]$, $k = 1 \dots M$, in its columns, since it depends on the choice of $\mathbf{K}$ (even when $\mathbf{K}$ yields perfect unmixing, there remains a scaling and permutation ambiguity compared to the signals in $\mathbf{s}$).

Define $\mathbf{C}_S = (\mathbf{S} - \overline{\mathbf{S}})(\mathbf{S} - \overline{\mathbf{S}})^T$, where $\overline{\mathbf{S}}$ denotes the $N \times M$ matrix for which each column contains the sample mean of the rows of $\mathbf{S}$, i.e. $\overline{\mathbf{S}} = \frac{1}{M}\mathbf{S}\,\mathbf{1}_M\mathbf{1}_M^T$, where $\mathbf{1}_M$ denotes an M-dimensional column vector in which each entry is 1. For notational convenience, we introduce the matrix $\mathbf{P} = \mathbf{I}_M - \frac{1}{M}\mathbf{1}_M\mathbf{1}_M^T$, to write $\mathbf{C}_S = (\mathbf{S} - \overline{\mathbf{S}})(\mathbf{S} - \overline{\mathbf{S}})^T = \mathbf{SPP}^T\mathbf{S}^T = \mathbf{SPS}^T$. Let

$$F(\mathbf{S}) = \sum_{n,m} \frac{\left[\mathbf{SPS}^T\right]_{nm}^2}{\left[\mathbf{SPS}^T\right]_{nn}\left[\mathbf{SPS}^T\right]_{mm}} \tag{9.9}$$

i.e. the function $F(\mathbf{S})$ evaluates the sum of the squared (cross-)correlation coefficients of the rows of $\mathbf{S}$.

According to Corollary 9.2, to obtain the original source signals in the rows of $\mathbf{S}$, it is sufficient to construct $\mathbf{S} = \mathbf{KY}$ such that $\mathbf{S} \geq 0$ and $\mathbf{C}_S$ is a diagonal matrix. This is translated into the following optimization problem:

$$\min_{\mathbf{S}} F(\mathbf{S}) \tag{9.10}$$

$$\text{s.t.} \begin{cases} \mathbf{S} \geq 0 \\ \exists\, \mathbf{K} \in \mathbb{R}^{N \times J} : \mathbf{S} = \mathbf{KY}\ . \end{cases} \tag{9.11}$$

The first constraint in (9.11) enforces non-negativity and the second constraint links the matrix $\mathbf{S}$ to the observations in $\mathbf{Y}$, such that both have the same

row space. The minimization of the cost function[4] $F(\mathbf{S})$ yields decorrelation of the rows of $\mathbf{S}$. The function $F(\mathbf{S})$ has multiple global minima, which are all equal to N, corresponding to the case where all cross-correlation coefficients are zero. Since the cost function $F(\mathbf{S})$ is non-convex, it has multiple stationary points. However, as shown by the following theorem, every local minimizer $\mathbf{S}^*$ corresponds to perfectly uncorrelated source signals in the rows of $\mathbf{S}^*$.

Theorem 9.3 *Let $\mathbf{S}^*$ denote a local minimizer of $F(\mathbf{S})$ (without taking the constraints (9.11) into account). Then the rows of $\mathbf{S}^*$ are uncorrelated, i.e. $\mathbf{C}_S^* = (\mathbf{S}^* - \overline{\mathbf{S}}^*)(\mathbf{S}^* - \overline{\mathbf{S}}^*)^T$ is a diagonal matrix.*

Proof : In the sequel, we ignore the points $\mathbf{S}$ for which $F(\mathbf{S})$ does not exist, i.e. the case where $\mathbf{S}$ has one or more zero-variance rows. The gradient of the cost function (9.9) is

$$\nabla F(\mathbf{S}) = 4\left(\mathbf{\Lambda}_1^{-1}\mathbf{S}\mathbf{P}\mathbf{S}^T\mathbf{\Lambda}_1^{-1} - \mathbf{\Lambda}_1^{-1}\mathbf{\Lambda}_2\right)\mathbf{S}\mathbf{P} \tag{9.12}$$

with

$$\mathbf{\Lambda}_1 = \mathcal{D}\left\{\mathbf{S}\mathbf{P}\mathbf{S}^T\right\} \tag{9.13}$$

$$\mathbf{\Lambda}_2 = \mathcal{D}\left\{\left(\mathbf{\Lambda}_1^{-1}\mathbf{S}\mathbf{P}\mathbf{S}^T\right)^2\right\} \tag{9.14}$$

and with $\mathcal{D}\{\mathbf{X}\}$ denoting the operator that sets all off-diagonal elements of $\mathbf{X}$ to zero. Let $\mathbf{S}^*$ denote a local minimizer of F, and therefore it satisfies $\nabla F(\mathbf{S}^*) = 0$, which is equivalent to

$$\mathbf{\Lambda}_1^{*-1}\mathbf{S}^*\mathbf{P}\mathbf{S}^{*\,T}\mathbf{\Lambda}_1^{*-1}\mathbf{S}^*\mathbf{P} = \mathbf{\Lambda}_1^{*-1}\mathbf{\Lambda}_2^*\mathbf{S}^*\mathbf{P} \tag{9.15}$$

where $\mathbf{\Lambda}_1^*$ and $\mathbf{\Lambda}_2^*$ are defined by (9.13)-(9.14) with $\mathbf{S}$ replaced by $\mathbf{S}^*$.

Note that $\mathbf{S}^*\mathbf{P} = (\mathbf{S}^* - \overline{\mathbf{S}}^*)$ has full row rank. This can be shown by contradiction as follows. Assume that $\mathbf{S}^*\mathbf{P}$ does not have full row rank. Then either $\mathbf{S}^*$ has a zero variance row, which can be excluded since then $F(\mathbf{S}^*)$ does not exist, or $\mathbf{S}^*$ has at least one row which is a linear combination of the other rows. Let the i-th row $[\mathbf{S}]_{i,:}$ denote such a row which is a linear combination of the other rows. Let $\mathbf{e}^T$ be an $M \times 1$ row vector with random numbers, which are uncorrelated with any row in $\mathbf{S}$. Then adding the vector $\alpha\mathbf{e}^T$ to the row $[\mathbf{S}]_{i,:}$, with α denoting any positive number, will result in a decrease of the cost function F. This shows that there exists a descent direction in $\mathbf{S}^*$. However, since $\mathbf{S}^*$ is a local minimizer of F, no such direction can exist in the point $\mathbf{S}^*$.

[4]Notice that we do not use the cost function $\sum_{n,m}\left[\mathbf{S}\mathbf{P}\mathbf{S}^T - \mathbf{I}_N\right]_{nm}^2$. Although this cost function would yield simpler updating formulas, cost function (9.9) is observed to yield a better performance due to its implicit normalization. This normalization makes the resulting updating formulas independent of the variance of the elements in $\mathbf{S}$.

Since $\mathbf{S}^*\mathbf{P} = (\mathbf{S}^* - \overline{\mathbf{S}}^*)$ has full row rank, $\mathbf{S}^*\mathbf{P}\mathbf{P}^T\mathbf{S}^{*\,T} = \mathbf{S}^*\mathbf{P}\mathbf{S}^{*\,T}$ has full rank and non-zero elements on its diagonal. Using this, and since both $\mathbf{\Lambda}_1^*$ and $\mathbf{\Lambda}_2^*$ are diagonal matrices, (9.15) is equivalent to

$$\mathbf{S}^*\mathbf{P}\mathbf{S}^{*\,T} = \mathbf{\Lambda}_1^*\mathbf{\Lambda}_2^* \,. \tag{9.16}$$

Since $\mathbf{S}^*\mathbf{P}\mathbf{S}^{*\,T} = (\mathbf{S}^* - \overline{\mathbf{S}}^*)(\mathbf{S}^* - \overline{\mathbf{S}}^*)^T = \mathbf{C}_S^*$, and since the righthand side of (9.16) is a diagonal matrix, the theorem is proven. □

Theorem 9.3 implies that any local minimizer $\mathbf{S}^*$ of F satisfies $F(\mathbf{S}^*) = N$ and hence is a global minimizer. It is thus sufficient to find a local minimum of (9.9) that satisfies the constraints (9.11), to solve the NICA problem.

The first constraint of (9.11) is a non-negativity constraint on the matrix $\mathbf{S}$. A popular way to minimize a cost function $F(\mathbf{S})$ under non-negativity constraints, is to use multiplicative update rules (cfr. e.g. [12, 15]). Multiplicative optimization algorithms are usually easy to implement compared to general contrained optimization (CO) techniques, and they do not require any step size search. The multiplicative update rules can be derived if the gradient of the cost function $F(\mathbf{S})$ can be split into a positive part and a negative part, i.e.

$$\nabla F(\mathbf{S}) = \nabla^+ F(\mathbf{S}) - \nabla^- F(\mathbf{S}) \tag{9.17}$$

with $[\nabla^- F(\mathbf{S})]_{nm} \geq 0$ and $[\nabla^+ F(\mathbf{S})]_{nm} \geq 0$, $n = 1 \ldots N$, $m = 1 \ldots M$, then the following multiplicative update rule can be used [15]:

$$[\mathbf{S}]_{nm} \leftarrow [\mathbf{S}]_{nm} \frac{[\nabla^- F(\mathbf{S})]_{nm}}{[\nabla^+ F(\mathbf{S})]_{nm}} \,. \tag{9.18}$$

Notice that, if $\mathbf{S}$ is initialized with non-negative numbers, all of its elements remain non-negative under the update (9.18), and the non-negativity constraint of (9.11) is automatically satisfied. There exist two kinds of fixed points for (9.18). The first satisfies $\nabla^+ F(\mathbf{S}) = \nabla^- F(\mathbf{S})$, yielding $\nabla F(\mathbf{S}) = 0$, i.e. a stationary point of the cost function $F(\mathbf{S})$. The other is $[\mathbf{S}]_{nm} = 0$, $n = 1 \ldots N$, $m = 1 \ldots M$. Notice that the updating procedure (9.18) cannot converge to a stationary point of F if certain elements of $\mathbf{S}$ that are non-zero in any stationary point, are set to zero. Indeed, any element that has a value of zero remains zero in all future iterations. We will refer to this as 'false zeros'.

It is generally difficult to prove convergence of multiplicative update formulas of the form (9.18). However, for many cost functions F, update (9.18) is found to converge to a local minimizer of F. This can be explained intuitively as follows. The variable $[\mathbf{S}]_{nm}$ decreases when $[\nabla^+ F(\mathbf{S})]_{nm} > [\nabla^- F(\mathbf{S})]_{nm}$, i.e. when $[\nabla F(\mathbf{S})]_{nm} > 0$. This means that the value changes in the opposite direction of the gradient. Therefore (9.18) is similar to a gradient descent update, where the step-size is different for each variable and in each step. More specifically,

(9.18) is equivalent to a *natural* gradient descent update, as pointed out in [15]. A natural gradient learning algorithm has the convenient property that it has isotropic convergence around any local optimum, independent of the model parametrization or the signals being processed [18].

By applying this technique to the gradient of $F(\mathbf{S})$, as given in (9.12)-(9.14), the following updating formula for the matrix $\mathbf{S}$ is found[5]:

$$[\mathbf{S}]_{nm} \leftarrow [\mathbf{S}]_{nm} \frac{\left[\overline{\mathbf{S}}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\mathbf{S} + \mathbf{S}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\overline{\mathbf{S}} + \mathbf{\Lambda}_2\mathbf{S}\right]_{nm}}{\left[\overline{\mathbf{S}}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\overline{\mathbf{S}} + \mathbf{S}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\mathbf{S} + \mathbf{\Lambda}_2\overline{\mathbf{S}}\right]_{nm}} \ . \tag{9.19}$$

Notice that this update does not take the second constraint of (9.11) into account. Therefore, an additional correction step is required after each update (9.19). To enforce the second constraint of (9.11), the rows of $\mathbf{S}$ are projected onto the signal subspace $\mathcal{S}$, which is equal to the row space of $\mathbf{Y}$:

$$\mathbf{S} \leftarrow \mathscr{P}_{\mathcal{S}}\{\mathbf{S}\} \tag{9.20}$$

where $\mathscr{P}_{\mathcal{S}}\{\mathbf{X}\}$ denotes the projection operator that projects the rows of the matrix $\mathbf{X}$ onto the signal subspace $\mathcal{S}$.

Notice that the projection $\mathscr{P}_{\mathcal{S}}\{\mathbf{S}\}$ can result in negative values in $\mathbf{S}$. To preserve non-negativity, $\mathbf{S}$ should actually be projected onto $\mathcal{S}^+ = \mathcal{S} \cap \mathbb{P}$ where $\mathbb{P}$ denotes the positive orthant, i.e.

$$\mathbf{S} \leftarrow \mathscr{P}_{\mathcal{S}^+}\{\mathbf{S}\} \ . \tag{9.21}$$

This projection can be iteratively computed with Dykstra's algorithm [19]. However, to reduce the complexity of the M-NICA algorithm, we use a heuristic procedure instead, as described in the next section.

Remark: It is noted that general CO techniques can also be used to solve the problem $\min_{\mathbf{K}} F(\mathbf{KY})$ s.t. $\mathbf{KY} \geq 0$, which is equivalent to (9.10)-(9.11). Experiments[6] indicate that only the interior point (IP) method [20] gives good results that are comparable to the unmixing performance of M-NICA and NPCA. However, for the experiments in Section 9.5, the computation time of the IP method is roughly the double[7] of the computation time of M-NICA and NPCA. Furthermore, the M-NICA algorithm (see Section 9.3.2) is much simpler to implement compared to an IP method, where in each iteration the Hessian matrix must be evaluated (i.e. second order numerical differentiation) or approximated, and a corresponding IP-KKT system must be solved with a

[5] We replaced $\mathbf{SPS}^T$ with $(\mathbf{S} - \overline{\mathbf{S}})\mathbf{S}^T$, instead of the equivalent substitution $\mathbf{SPS}^T = \mathbf{SPP}^T\mathbf{S}^T = (\mathbf{S} - \overline{\mathbf{S}})(\mathbf{S} - \overline{\mathbf{S}})^T$.

[6] We also tried an active set method and a Levenberg-Marquardt based algorithm [20]. Both methods give good results in some cases, but their separation performance varies significantly over multiple Monte-Carlo experiments. Especially in scenarios with many sources ($N > 2$) and/or overdetermined observations ($J > N$), both methods generally yield very poor results.

[7] Based on Matlab's *fmincon* command.

subsequent step-size search. Each IP-KKT system is of large dimension due to the large amount of inequality constraints that enforce non-negativity of each unmixed sample.

9.3.2 The Multiplicative NICA Algorithm (M-NICA)

The following fixed-point algorithm is used to solve (9.9)-(9.11), and is referred to as multiplicative non-negative ICA (M-NICA):

1. *Initialization*:
   (a) $\forall\, n = 1 \ldots N, \forall\, m = 1 \ldots M:\ [\mathbf{S}]_{nm} \leftarrow |[\mathbf{Y}]_{nm}|$
   (b) Replace $\mathbf{Y}$ by its best rank N approximation by means of the singular value decomposition (SVD), i.e.

$$\{\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}\} \leftarrow \mathrm{SVD}\,(\mathbf{Y}) \tag{9.22}$$

$$\mathbf{Y} \leftarrow \overline{\mathbf{U}}\ \overline{\mathbf{\Sigma}}\ \overline{\mathbf{V}}^T \tag{9.23}$$

   where $\overline{\mathbf{\Sigma}}$ is the $N \times N$ diagonal matrix containing the N largest singular values[8] of $\mathbf{Y}$ on its diagonal, and where the corresponding left and right singular vectors are stored in the columns of $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$ respectively.

2. *Decorrelation step*:

$$\forall\, n = 1 \ldots N, \forall\, m = 1 \ldots M:$$
$$\left[\mathbf{S}^{\mathrm{temp}}\right]_{nm} \leftarrow [\mathbf{S}]_{nm} \frac{\left[\overline{\mathbf{S}}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\mathbf{S} + \mathbf{S}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\overline{\mathbf{S}} + \mathbf{\Lambda}_2\mathbf{S}\right]_{nm}}{\left[\overline{\mathbf{S}}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\overline{\mathbf{S}} + \mathbf{S}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\mathbf{S} + \mathbf{\Lambda}_2\overline{\mathbf{S}}\right]_{nm}} \tag{9.24}$$

   with

$$\overline{\mathbf{S}} = \frac{1}{M}\mathbf{S}\,\mathbf{1}_M\mathbf{1}_M^T \tag{9.25}$$
$$\mathbf{C}_S = (\mathbf{S} - \overline{\mathbf{S}})(\mathbf{S} - \overline{\mathbf{S}})^T \tag{9.26}$$
$$\mathbf{\Lambda}_1 = \mathcal{D}\left\{\mathbf{C}_S\right\} \tag{9.27}$$
$$\mathbf{\Lambda}_2 = \mathcal{D}\left\{\left(\mathbf{\Lambda}_1^{-1}\mathbf{C}_S\right)^2\right\}\ . \tag{9.28}$$

3. *Signal subspace projection step*:

$$\forall\, n = 1 \ldots N, \forall\, m = 1 \ldots M:$$
$$[\mathbf{S}]_{nm} \leftarrow \max\left(\left[\mathbf{S}^{\mathrm{temp}}\ \overline{\mathbf{V}}\ \overline{\mathbf{V}}^T\right]_{nm}, 0\right)\ . \tag{9.29}$$

4. Return to step 2.

[8] Notice that, if noise were present, this step will remove some noise from the observations. In the noise-free case, $\mathbf{Y}$ has exactly N non-zero singular values.

In step 3, the algorithm computes a projection onto $\mathcal{S}$, followed by a projection onto $\mathbb{P}$, instead of computing the exact projection $\mathscr{P}_{\mathcal{S}^+}\{\mathbf{S}\}$ as given in (9.21). This significantly reduces the computational load, and it is observed to work fine in our simulations, since the negative values that appear after the projection onto $\mathcal{S}$ are observed to be very sparse. After several iterations of the algorithm, the number of negative values after the projection onto $\mathcal{S}$ becomes negligible. Notice that $\mathbf{S}$ is initialized with absolute values of the elements of $\mathbf{Y}$. The absolute value guarantees that the initial $\mathbf{S} \in \mathbb{P}$, which is required when using multiplicative updates. Furthermore, by initialising $\mathbf{S}$ with (positive) elements of $\mathbf{Y}$, the initial matrix $\mathbf{S}$ will be 'close' to the subspace $\mathcal{S}$. Notice that, if the mixing matrix $\mathbf{A}$ is non-negative, then $\mathbf{Y}$ is non-negative, and hence the initial matrix $\mathbf{S}$ starts in the solution space $\mathcal{S}^+$, defined by the constraints (9.11).

The M-NICA algorithm is a fixed-point type algorithm, which has the facilitating property that it does not depend on any user-defined stepsize parameter, as opposed to the NPCA algorithm described in Section 9.2. The algorithm searches for a good approximation of the solution of (9.9)-(9.11), i.e. a common fixed point of (9.24) and (9.29). Notice that many of the false zeros of (9.24) are eliminated since they are reset to a non-zero value due to (9.29) and therefore, they can again be updated by the multiplicative decorrelation process. In extensive simulations with different types of signals, the M-NICA algorithm was always observed to converge. This is stated here as an observation, since a formal proof is not available. Once the algorithm has converged, the mixing matrix[9] $\hat{\mathbf{A}}$ and the unmixing matrix $\mathbf{K}$ can be computed as

$$\hat{\mathbf{A}} = \mathbf{Y}\mathbf{S}^T \left(\mathbf{S}\mathbf{S}^T\right)^{-1} \tag{9.30}$$

$$\mathbf{K} = \mathbf{S}\overline{\mathbf{V}}\,\overline{\mathbf{\Sigma}}^{-1}\overline{\mathbf{U}}^T \,. \tag{9.31}$$

Notice that there always remains a permutation and scaling ambiguity between the columns of $\hat{\mathbf{A}}$ and the rows of $\mathbf{S}$.

Assuming that $M \gg N$, then the overall complexity of the M-NICA algorithm is $O(N^2M)$, which is the same as the NPCA algorithm.

9.4 Sliding-Window M-NICA

In this section, we describe an adaptive version of the M-NICA algorithm, which corresponds to a sliding window implementation of the batch mode version of M-NICA. The window slides over the observed signal $\mathbf{y}$, sample by sample. After each window shift, a new sample is added to the window, and an old sample is removed. A sample that enters the window is first unmixed with an unmixing matrix computed from the previous samples. After each window

[9] We add a hat to denote that $\hat{\mathbf{A}}$ is an estimate of the actual mixing matrix $\mathbf{A}$.

shift, a single iteration[10] of the batch mode M-NICA algorithm is performed on the samples that are currently in the window.

Let K denote the number of samples in the sliding window. We use Matlab notation to denote rows and columns, i.e. $[\mathbf{M}]_{i,:}$ and $[\mathbf{M}]_{:,j}$ respectively denote the i-th row and the j-th column of the matrix $\mathbf{M}$. The adaptive implementation of M-NICA is then given as follows. For notational convenience, we omit all universal quantifiers. Index n is always assumed to attain all values from 1 to N and index k is assumed to attain all values from 1 to K.

1. *Initialization*:
   (a) $[\mathbf{W}_Y]_{:,k} \leftarrow \mathbf{y}[k]$
   (b) $[\mathbf{W}_S]_{nk} \leftarrow |[\mathbf{W}_Y]_{nk}|$
   (c) $\mathbf{K} \leftarrow \mathbf{W}_S \mathbf{W}_Y^{\dagger}$, where $\mathbf{W}_Y^{\dagger}$ denotes the pseudo-inverse of $\mathbf{W}_Y$.
   (d) $i \leftarrow K - 1$
2. *Window updates*:
   (a) $c \leftarrow (i \mod K) + 1$
   (b) $i \leftarrow i + 1$
   (c) $[\mathbf{W}_Y]_{:,c} \leftarrow \mathbf{y}[i]$
   (d) Replace $\mathbf{W}_Y$ by its best rank N approximation by means of the singular value decomposition (SVD), i.e.

$$\{\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}\} \leftarrow \mathrm{SVD}\left(\mathbf{W}_Y\right) \tag{9.32}$$

$$\mathbf{W}_Y \leftarrow \overline{\mathbf{U}}\ \overline{\mathbf{\Sigma}}\ \overline{\mathbf{V}}^T \tag{9.33}$$

   where $\overline{\mathbf{\Sigma}}$ is the $N \times N$ diagonal matrix containing the N largest singular values of $\mathbf{W}_Y$ on its diagonal, and where the corresponding left and right singular vectors are stored in the columns of $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$ respectively.
   (e) $[\mathbf{W}_S]_{nk} \leftarrow \max\left([\mathbf{K}\mathbf{W}_Y]_{nk}, 0\right)$
3. *Decorrelation step*: compute (9.24) where $\mathbf{S}$ and $\mathbf{S}^{\mathrm{temp}}$ are replaced by $\mathbf{W}_S$ and $\mathbf{W}_S^{\mathrm{temp}}$, respectively.
4. *Computation of unmixing matrix*:

$$\mathbf{K} \leftarrow \mathbf{W}_S^{\mathrm{temp}} \overline{\mathbf{V}}\ \overline{\mathbf{\Sigma}}^{-1} \overline{\mathbf{U}}^T$$

$$[\mathbf{K}]_{n,:} \leftarrow \frac{[\mathbf{K}]_{n,:}}{\| [\mathbf{K}]_{n,:} \|}$$

5. *Estimation of sample* $\mathbf{s}[i]$:

$$\hat{\mathbf{s}}[i] = \mathbf{K}\mathbf{y}[i]$$

[10] To achieve faster convergence, multiple iterations can be performed after each window shift.

6. Return to step 2.

Notice that step 2(e) corresponds to the signal subspace projection step in the batch mode algorithm. Step (9.32) can be implemented efficiently by means of sliding window subspace tracking methods, e.g. [21, 22]. Also notice that the rows of the unmixing matrix $\mathbf{K}$ are normalized in each iteration to remove the scaling ambiguity in the NICA problem. Notice that $\mathbf{K}$ is normalized rather than $\mathbf{W}_S$, since a normalization of $\mathbf{W}_S$ would result in an unmixing matrix that varies over time when the signals in $\mathbf{s}$ are non-stationary, which is undesirable in view of the sample by sample unmixing in step 5 of the algorithm.

The window length K introduces a trade-off: it should be large enough such that the window contains enough samples to compute a reliable estimate of the correlation coefficients, and to make sure that the independence assumption is not violated. On the other hand, it should be small enough to achieve sufficient tracking performance.

9.5 Batch Mode Simulations

In this section, we provide batch mode simulation results for M-NICA and NPCA with different types of signals. We use two different measures to assess the performance of these algorithms: the signal-to-error ratio (SER) and the mean squared error (MSE), i.e.

$$\text{SER} = \frac{1}{N}\sum_{n=1}^{N} 10\log_{10}\frac{E\{s_n^2\}}{E\{(s_n-\hat{s}_n)^2\}} \tag{9.34}$$

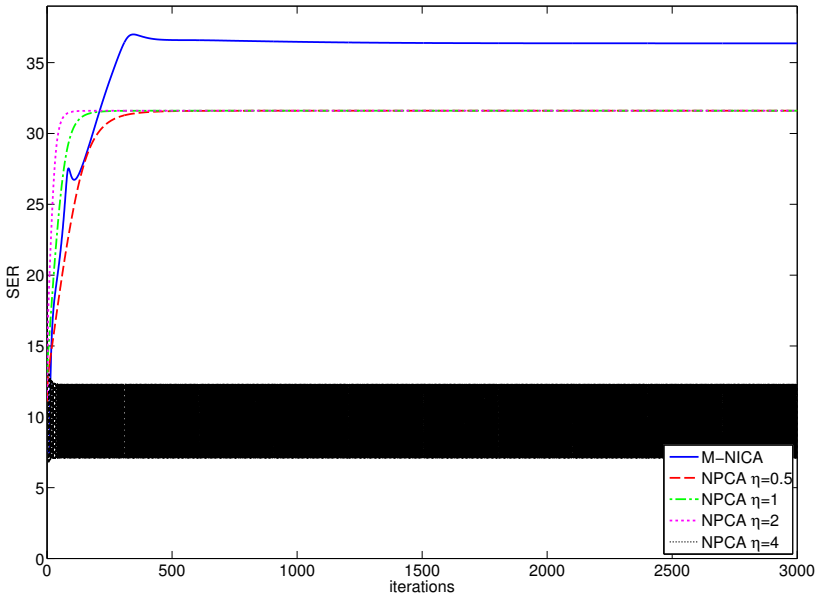
and

$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N} E\{(s_n-\hat{s}_n)^2\} \tag{9.35}$$

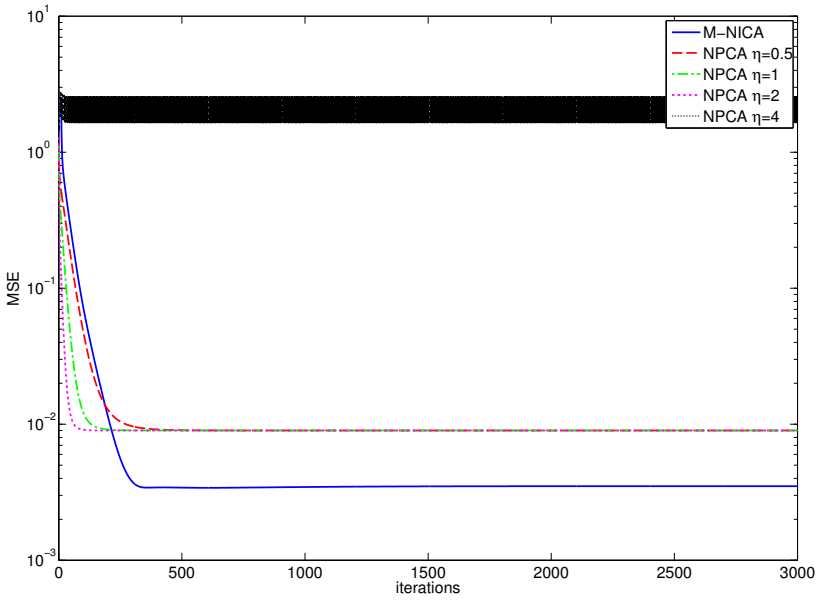
where $\hat{s}_n$ denotes the reconstruction of the n-th source signal, after an optimal (least squares) rescaling to resolve the scaling ambiguity between s_n and $\hat{s}_n$. Notice that NPCA does not explicitly enforce the unmixed signals to be non-negative, whereas M-NICA enforces this in (9.29). To obtain a fair comparison between both algorithms, we half-wave rectify the signals obtained by NPCA, i.e. negative values are set to zero.

9.5.1 Uniformly Distributed Random Signals on the Unit Interval

In this experiment, we used a uniformly distributed random process on the unit interval to generate $M = 1000$ samples of the $N = 3$ source signals. The

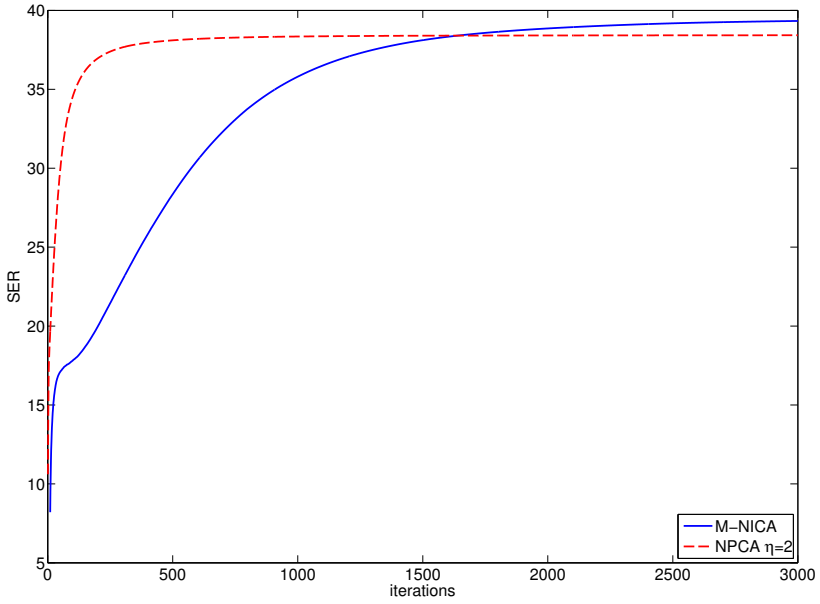


(a) SER

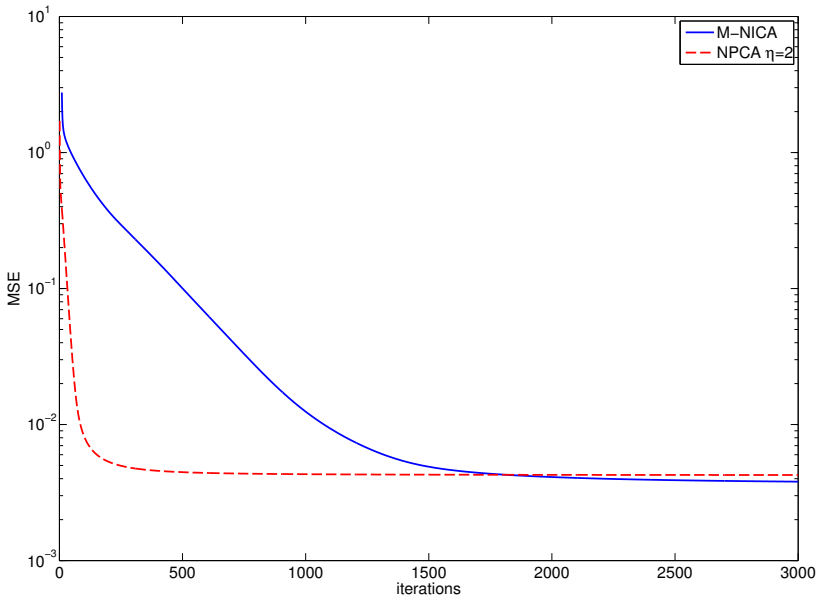


(b) MSE

Figure 9.1: SER and MSE for random signals that are uniformly distributed on the unit interval.



(a) SER



(b) MSE

Figure 9.2: SER and MSE for random signals that are uniformly distributed on the unit interval, averaged over 1000 experiments.

mixing matrix $\mathbf{A}$ is a 10×3 (J=10) matrix with random numbers drawn from a zero-mean normal distribution.

In Fig. 9.1(a) and 9.1(b) the SER and the MSE of both algorithms are plotted versus the number of iterations. It is observed that the convergence rate of NPCA depends on the choice of η. If η is set to a proper value, NPCA converges faster than M-NICA. However, if the value for η is too large, i.e. $\eta = 4$ in this case, NPCA does not converge and results in a suboptimal unmixing (in Fig. 9.1(a) and 9.1(b), this results in a black band due to the oscillation of the SER and MSE over the different iterations). Despite the slower convergence, M-NICA has a higher unmixing accuracy.

It should be noted that the convergence speed and the accuracy of the algorithms varies over different experiments. To draw more general conclusions, we performed 1000 Monte-Carlo simulations and averaged out the results. The learning rate for NPCA is set to $\eta = 2$, which is observed to provide the best results (both in terms of convergence and accuracy). The average SER and MSE versus the number of iterations are shown in Fig. 9.2(a) and 9.2(b). It is observed that NPCA generally converges much faster than M-NICA, but M-NICA slightly outperforms NPCA in terms of unmixing accuracy.

9.5.2 Sparse Signals on the Unit Interval

In this experiment, we model sparse random processes, i.e. $\exists\, \alpha > 0,\ \forall\, \delta > 0 :\ Pr(0 \leq s_n < \delta) > \alpha$. This model can be used when the sources have an on-off behaviour, or when analyzing signal spectra that are known to be sparse, e.g. [4, 8]. Notice that the well-grounded assumption is very well satisfied for this type of signals.

For the simulations, we use a signal that is similar to what we used in the previous section, but we modify it to model on-off behavior of the sources, i.e. the signal contains clusters of zero valued samples corresponding to the source being 'off' during a certain time segment[11]. To model this, the following random process is repeated for each of the $N = 3$ sources signals, until $M = 1000$ samples are generated

1. Let p define a binary random variable that can attain the values 0 or 1 with equal probability. Let q define an integer random variable that can attain values from 1 to 10 with equal probability.
2. Draw a sample P from p. If $P = 0$, go to step 3, and if $P = 1$, go to step 4.
3. Draw a sample Q from q. The next Q samples of the signal s are zero. Then go back to step 2.

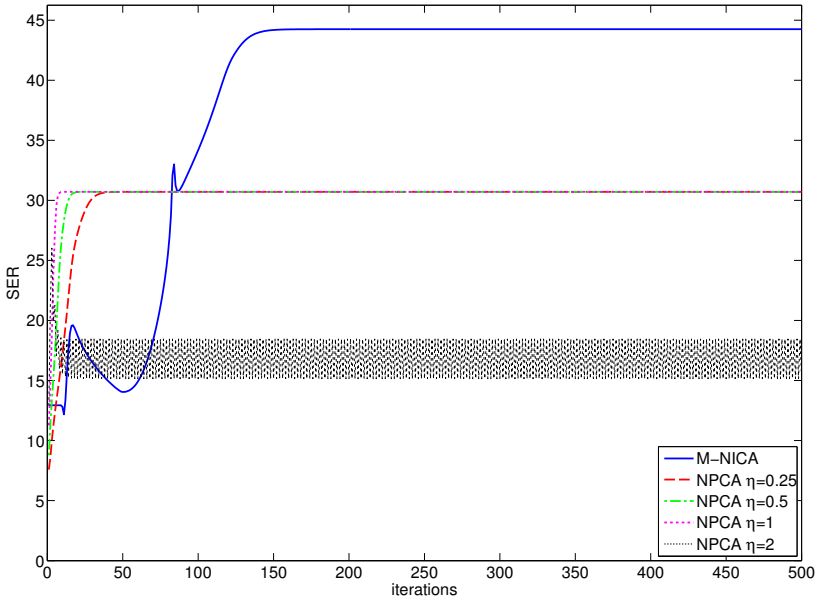
[11] For example, this is similar to the power of a speech signal analyzed in time, where pauses in between words and sentences create bursts of zeros [4].

4. Draw a sample Q from q. The next Q samples of the signal s are drawn from a uniformly distributed random process on the unit interval. Then go back to step 2.

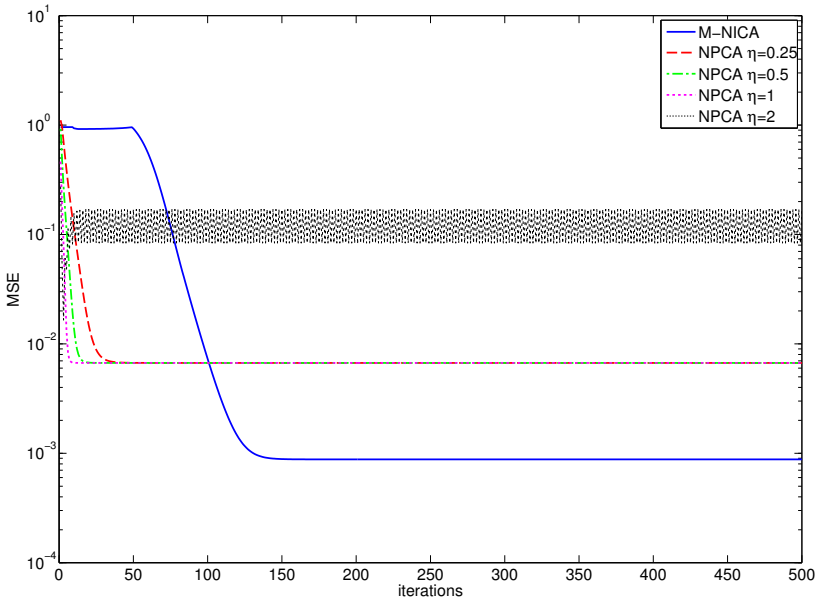
Notice that the total time during which the source is switched off is approximately equal to the time during which the source is active. The mixing matrix is constructed as in the experiment described in Section 9.5.1.

Fig. 9.3(a) and 9.3(b) plot the SER and MSE versus the number of iterations for both algorithms. It is observed that NPCA converges faster than M-NICA. However, M-NICA again yields a better unmixing accuracy. As opposed to the previous experiment, the learning rate of NPCA should now be set to a smaller value to obtain convergence.

To draw more general conclusions, we again performed 1000 Monte-Carlo simulations and averaged out the results. The learning rate of NPCA is set to $\eta = 0.5$. Larger values are observed to often cause NPCA not to converge. The average SER and MSE are shown in Fig. 9.4(a) and 9.4(b). Both algorithms converge much faster compared to the previous experiment (compare with Fig. 9.2(a) and 9.2(b)), which is due to the sparsity of the signal. It is again observed that NPCA converges fastest, but that M-NICA outperforms NPCA in terms of unmixing accuracy. The difference in unmixing accuracy between both algorithms appears to be more significant for sparse signals, i.e. more than 5 dB in SER (compare to Fig. 9.2).

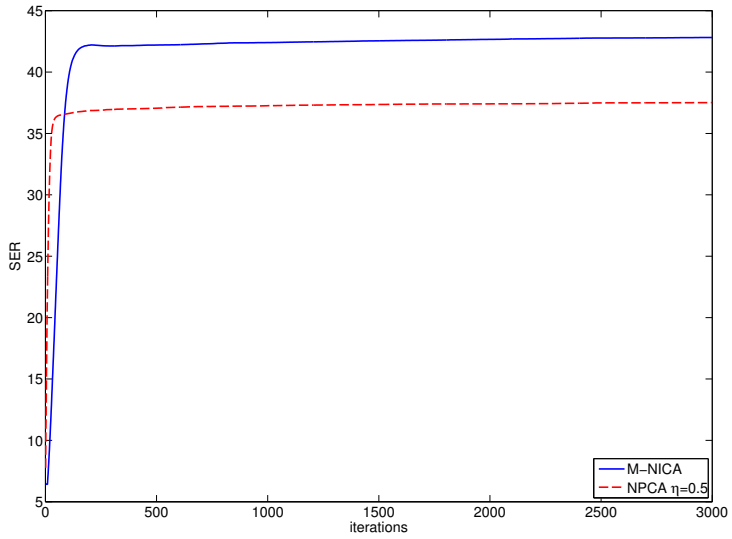


(a) SER

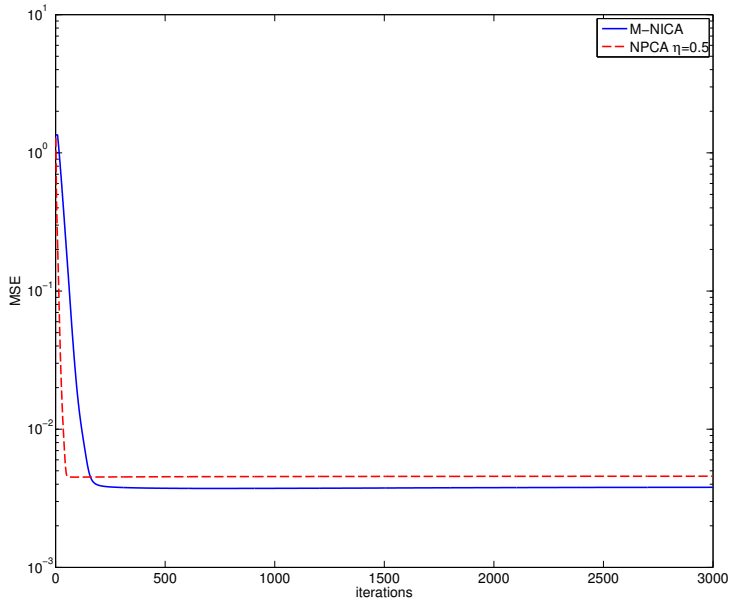


(b) MSE

Figure 9.3: SER and MSE for random sparse signals on the unit interval.



(a) SER



(b) MSE

Figure 9.4: SER and MSE for random sparse signals on the unit interval, averaged over 1000 experiments.

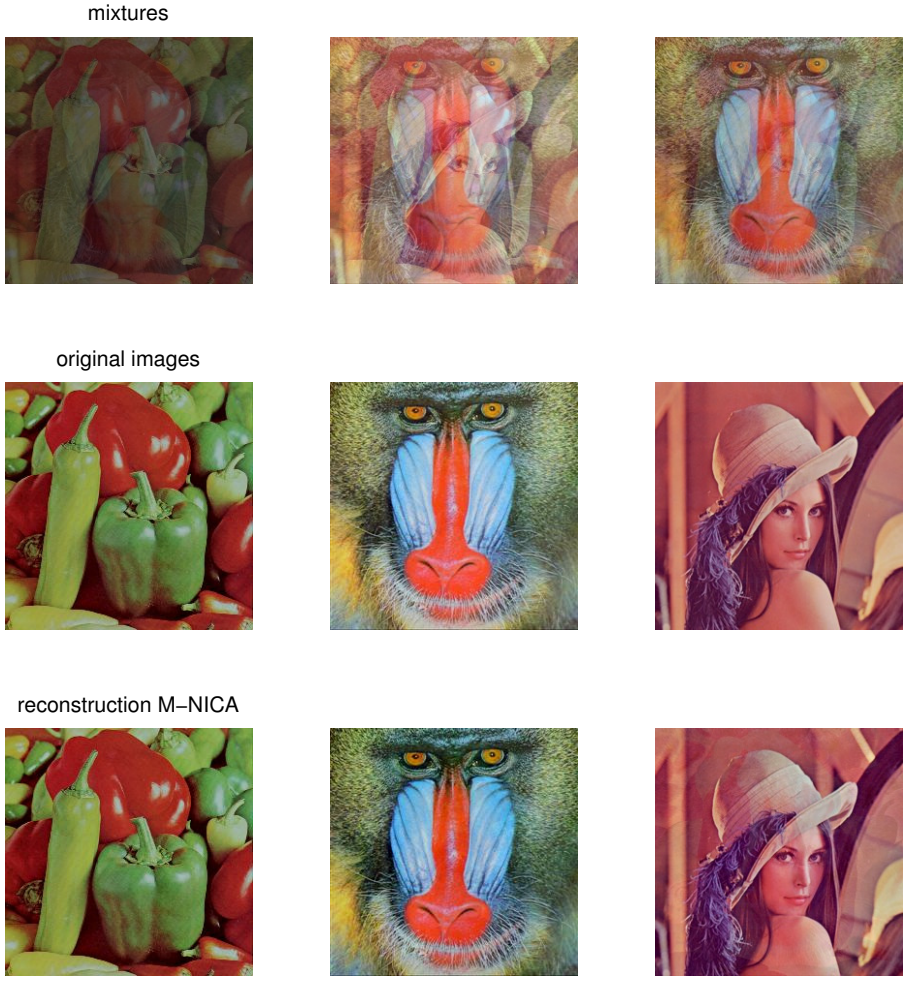


Figure 9.5: Three mixtures (first row) of the three original images (second row), and the corresponding unmixed images with the M-NICA algorithm (third row).

9.5.3 Images

In this experiment, we generate 3 non-negative mixtures of 3 color images. Notice that the pixel values of images are non-negative, and therefore this defines a NICA problem. The original images and the unmixed images by M-NICA are shown in Fig. 9.5. Fig. 9.6 shows the SER versus the iteration index for M-NICA and NPCA. It is observed that M-NICA yields a significantly more accurate unmixing.

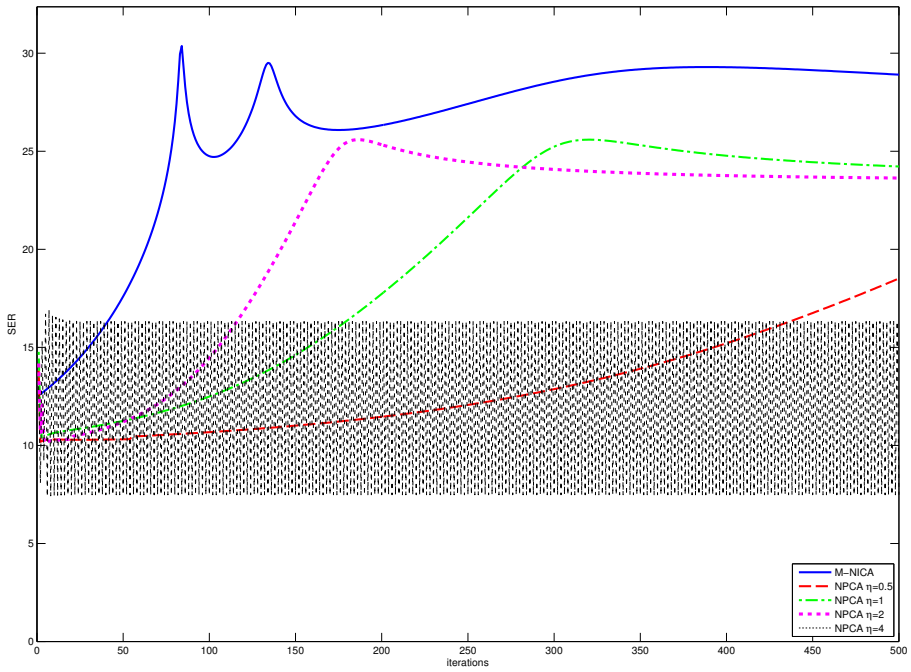


Figure 9.6: SER versus iteration index for images.

9.5.4 Effect of Sample Size

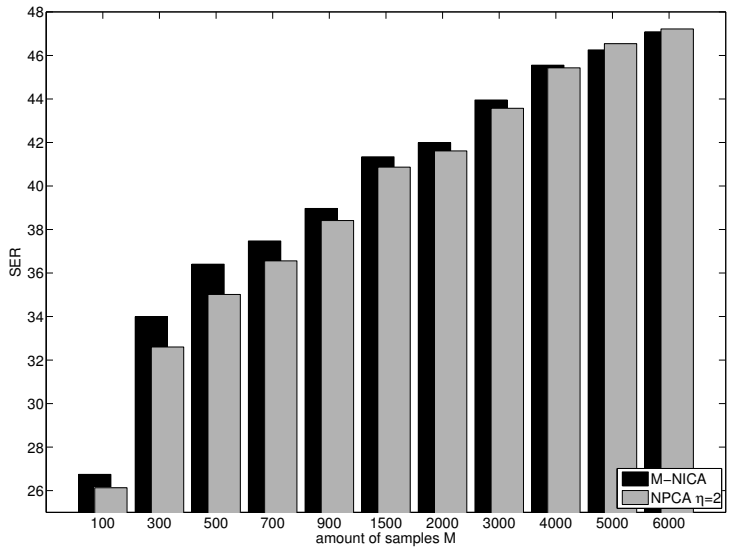
In the following Monte-Carlo experiment, we want to analyze the performance of M-NICA and NPCA for different amounts of available data samples M. Fig. 9.7(a) and Fig. 9.7(b) show the resulting SER for data generated as in Section 9.5.1 (uniformly distributed signals) and Section 9.5.2 (sparse signals) respectively. The results are averaged over 200 experiments. We performed 3000 iterations of M-NICA and NPCA with the uniformly distributed data, and 600 iterations with the sparse data since the latter yields faster convergence.

In Fig. 9.7(a), i.e. the case of uniformly distributed signals, it is observed that M-NICA outperforms NPCA if the amount of available samples is small. A possible reason for this is the fact that the samples of the original source signals are slightly correlated due to using finite sample sets. Since the decorrelation process of M-NICA is based on an explicit minization process that satisfies a non-negativity constraint, this correlation between the original samples will partly remain in the unmixed data. On the other hand, NPCA starts by perfectly decorrelating the data samples with a whitening matrix while ignoring this non-negativity constraint. This removes all correlation that was present in the original samples of the unmixed source signals, yielding an unavoidable distortion. If the amount of data samples is sufficiently large[12], NPCA has a similar (or better) unmixing accuracy compared to M-NICA. In Fig. 9.7(b), it is again observed that M-NICA outperforms NPCA, and that this effect is more significant when using small sample sizes. For $M = 100$, the relative difference in SER is approximately 20%, whereas this is approximately 8% when $M = 30000$.

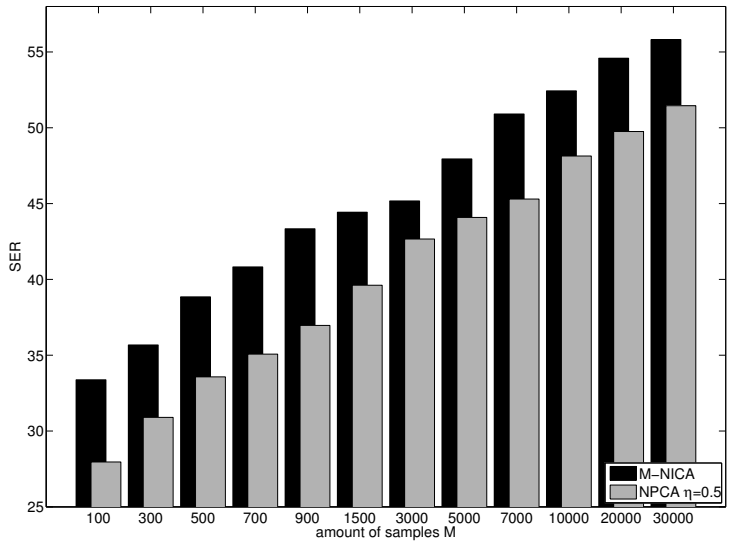
9.5.5 Conclusions

The above experiments demonstrate that the behavior of NPCA heavily depends on the choice of the learning rate η. The proper choice of η depends on the signals that are involved, and should be tuned by the user to ensure convergence and to obtain a good separation performance. The major advantage of the M-NICA algorithm is that it does not depend on any user-defined parameter. Furthermore, although the M-NICA algorithm is usually slower than the NPCA algorithm, it generally yields a better separation performance than NPCA, especially when the amount of available data samples is small. In the case of sparse signals, M-NICA has good convergence properties and a significantly better unmixing accuracy than NPCA.

[12]For very large data sets (i.e. $M > 10000$), the results are not shown here since M-NICA needs more than 3000 iteration to converge in this case. This is not the case for sparse signals, as observed in Fig. 9.7(b), since M-NICA converges much faster on this type of data.



(a) Uniformly distributed



(b) Sparse

Figure 9.7: SER, averaged over 200 experiments, as a function of sample size.

9.6 Sliding Window Simulations

In this section, we provide simulation results of a sliding window implementation of M-NICA and NPCA with different types of signals. We use the same measures as in Section 9.5 to assess the performance of the algorithms, i.e. the SER and the MSE. However, since we consider a sliding window implementation, both measures are computed over a window of length K and vary over time.

The sliding window implementation of M-NICA is described in Section 9.4. We add $K-1$ zeros at the beginning of each signal, to be able to estimate each sample of $\mathbf{s}[i]$ starting from $i=0$. This means that the windows $\mathbf{W}_Y$ and $\mathbf{W}_S$ are initialized with $K-1$ all-zero columns.

The sliding window implementation of NPCA corresponds to its batch mode version described in Section 9.2, where now one iteration is performed for each position of the sliding window. This means that each time a new sample is added, the whitening matrix is updated according to (9.2), and the rotation matrix $\mathbf{W}$ is updated according to (9.3)-(9.5), where the expectation operator is replaced by an averaging over the samples in the window.

9.6.1 Uniformly Distributed Random Signals on the Unit Interval

The signal and mixing matrix generation for this experiment is the same as in Section 9.5.1. However, to show the adaptation capabilities of the algorithms, we change the mixing matrix $\mathbf{A}$ after 1000 samples to another mixing matrix. A window length of $K=200$ seems to provide a good balance between adaptation speed and unmixing accuracy.

Fig. 9.8, shows the variation in SER, MSE and the cross-correlation between the estimated source signals, over time. The cross correlation is computed as the sum of the absolute values of the cross-correlation coefficients between the estimated sources signals. This is only shown for M-NICA since the cross-correlation is always zero in the case of NPCA, due to the whitening procedure. The drop in the SER, and the increase in the MSE and the cross-correlation at sample time 1000 is due to the sudden change of the mixing matrix $\mathbf{A}$. Again, it is observed that NPCA breaks down if the learning rate η is set too large. M-NICA provides the best unmixing accuracy.

To draw more general conclusions, we performed 1000 Monte-Carlo simulations of this experiment and averaged out the results. We set the learning rate of NPCA to $\eta=2$, which is observed to provide best results. The average SER and MSE over time is shown in Fig. 9.9. It is observed that, in general, M-NICA performs significantly better than NPCA in terms of unmixing accuracy, which

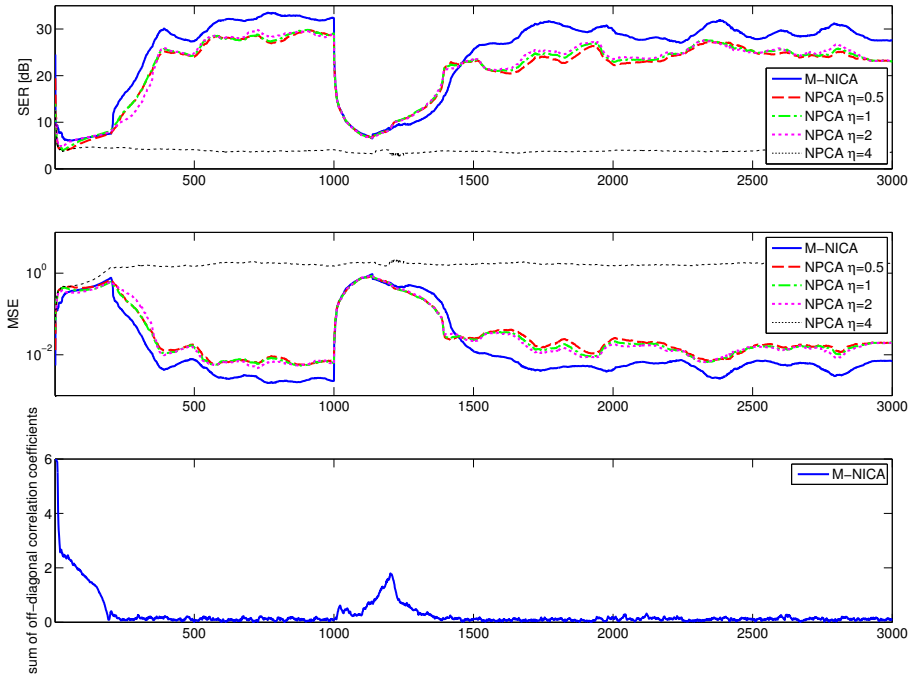


Figure 9.8: The SER (above), MSE (middle) and the absolute value of the sum of the cross-correlation coefficients between the sources (below). All three measures are computed over the samples in the sliding window buffer.

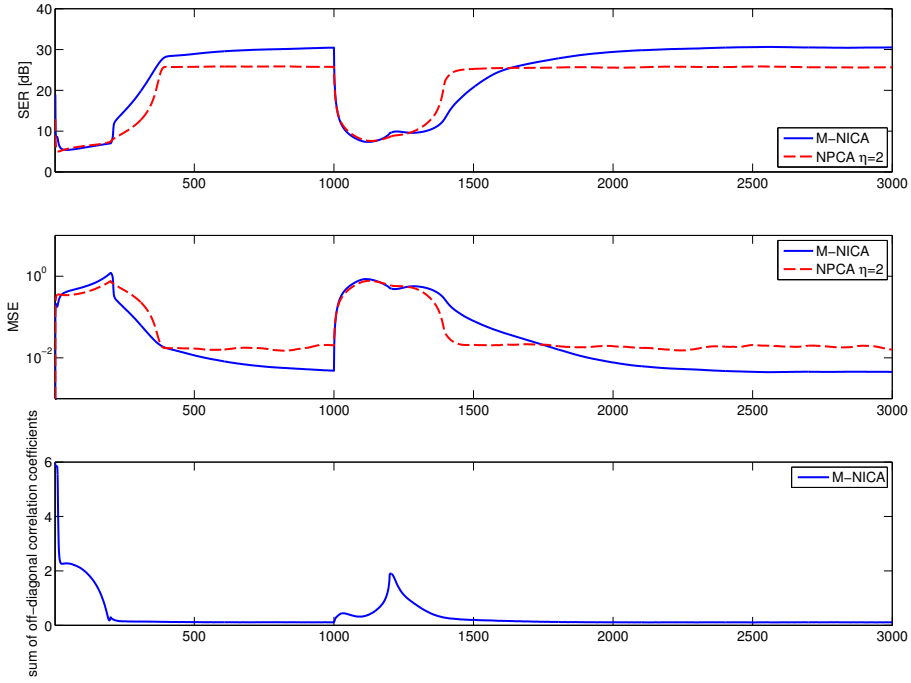


Figure 9.9: The averaged SER (above), MSE (middle) and the absolute value of the sum of the cross-correlation coefficients between the sources (below). All three measures are computed over the samples in the sliding window buffer, and averaged over 1000 experiments.

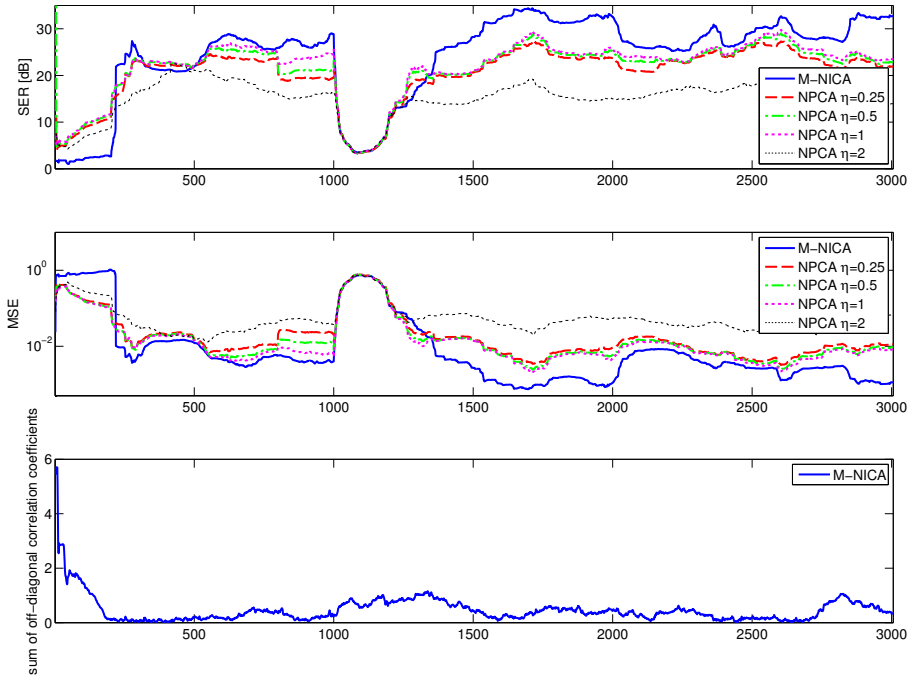


Figure 9.10: The SER (above), MSE (middle) and the absolute value of the sum of the cross-correlation coefficients between the sources (below). All three measures are computed over the samples in the sliding window buffer.

may be explained by the fact that the window contains only a small amount of data samples. The difference in convergence speed between both algorithms is less distinct compared to the batch mode experiments (compare with Fig. 9.2).

9.6.2 Sparse Signals on the Unit Interval

In this experiment, we analyze the performance of sliding window M-NICA and NPCA for sparse signals, generated in the same way as in Section 9.5.2. Again, we change the mixing matrix $\mathbf{A}$ after 1000 samples, and the window length is again set to $K = 200$.

Fig. 9.8 shows the variation in SER, MSE and the cross-correlation between the estimated source signals, over time. Again it is observed that M-NICA yields a better reconstruction of the source signal, compared to NPCA.

The averaged results of 1000 Monte-Carlo simulations is shown in Fig. 9.11. The learning rate for NPCA is set to $\eta = 0.5$. Again it is observed that, in gen-

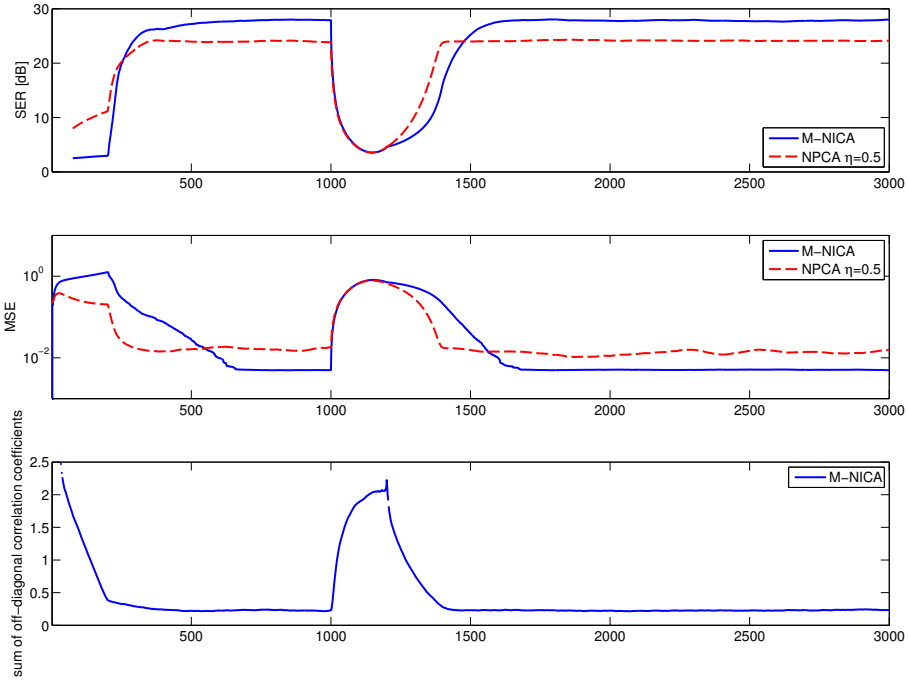


Figure 9.11: The averaged SER (above), MSE (middle) and the absolute value of the sum of the cross-correlation coefficients between the sources (below). All three measures are computed over the samples in the sliding window buffer, and averaged over 1000 experiments.

eral, M-NICA performs significantly better than NPCA in terms of unmixing accuracy.

In [4], both sliding window algorithms are applied to track the power of multiple simultaneous speech signals. The results are consistent with the experiments in this paper, i.e. M-NICA significantly outperforms NPCA at the cost of a slightly slower adaptation speed.

9.7 Conclusions

In this paper, we have proposed a new algorithm, referred to as M-NICA, to solve non-negative ICA problems with well-grounded sources. The M-NICA algorithm is based on multiplicative update rules which preserve non-negativity, together with a subspace projection based correction step. It has the facilitating property that it does not depend on a user-defined learning rate, as opposed to gradient based techniques such as the NPCA algorithm, where a proper choice for the learning rate is crucial to provide satisfying results.

The performance of M-NICA has been demonstrated by means of simulation results with different types of signals. Batch mode simulations indicated that M-NICA has a better unmixing accuracy than NPCA, but with slower convergence. In the case of sparse signals, M-NICA has good convergence properties, and significantly outperforms NPCA in terms of unmixing accuracy. It is also observed that M-NICA is best suited when the amount of available data samples is small. A sliding window implementation of both algorithms has also been described and validated, again showing that M-NICA significantly outperforms NPCA.

Bibliography

[1] P. Comon and C. Jutten, *Handbook of Blind Source Separation.* New York: Academic Press, 2010.

[2] A. Cichocki and S. Amari, *Adaptive Blind Signal and Image Processing.* New York: J. Wiley, 2002.

[3] E. Oja and M. Plumbley, "Blind separation of positive sources using non-negative PCA," in *Proc. of the 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, Nara, Japan, April 2003.

[4] A. Bertrand and M. Moonen, "Energy-based multi-speaker voice activity detection with an ad hoc microphone array," in *Proc. IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), March 2010, pp. 85–88.

[5] P. Pauca, R. Plemmons, M. Giffin, and K. Hamada, "Unmixing spectral data for space objects using independent component analysis and nonnegative matrix factorization," in *Proceedings Amos Technical Conference*, 2004.

[6] J. Nascimento and J. Dias, "Does independent component analysis play a role in unmixing hyperspectral data?" *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 1, pp. 175–187, January 2005.

[7] L. Parra, C. Spence, P. Sajda, A. Ziehe, and K.-R. Muller, "Unmixing hyperspectral data," in *Advances in Neural Information Processing (Proc. NIPS)*. MIT Press, 2000, pp. 942–948.

[8] D. Nuzillard and J.-M. Nuzillard, "Application of blind source separation to 1-D and 2-D nuclear magnetic resonance spectroscopy," *IEEE Signal Processing Letters*, vol. 5, no. 8, pp. 209–211, August 1998.

[9] R. C. Henry, "Multivariate receptor models–current practice and future trends," *Chemometrics and Intelligent Laboratory Systems*, vol. 60, no. 1-2, pp. 43 – 48, 2002.

[10] S. Abdallah and M. Plumbley, "Polyphonic transcription by non-negative sparse coding of power spectra," in *Proc. Int. Conf. Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004, pp. 318–325.

[11] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization." *Nature*, vol. 401, no. 6755, pp. 788–791, October 1999.

[12] ——, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing (Proc. NIPS)*. MIT Press, 2000, pp. 556–562.

[13] F. Y. Wang, C. Y. Chi, T. H. Chan, and Y. Wang, "Blind separation of positive dependent sources by non-negative least-correlated component analysis," in *Proc. IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, September 2006, pp. 73–78.

[14] T.-H. Chan, W.-K. Ma, C.-Y. Chi, and Y. Wang, "Blind separation of non-negative sources by convex analysis: Effective method using linear programming," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2008, pp. 3493–3496.

[15] Z. Yang and J. Laaksonen, "Multiplicative updates for non-negative projections," *Neurocomputing*, vol. 71, no. 1-3, pp. 363 – 373, 2007.

[16] M. Plumbley, "Conditions for nonnegative independent component analysis," *IEEE Signal Processing Letters*, vol. 9, no. 6, pp. 177–180, June 2002.

[17] M. Ye, "Global convergence analysis of a discrete time nonnegative ICA algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 253–256, January 2006.

[18] S. Amari and S. Douglas, "Why natural gradient?" in *Acoustics, Speech and Signal Processing, 1998. ICASSP 1998. IEEE International Conference on*, vol. 2, May 1998, pp. 1213–1216.

[19] J. Boyle and R. Dykstra, "A method for finding projections onto the intersection of convex sets in Hilbert spaces," *Lecture Notes in Statistics*, vol. 37, pp. 28–47, 1986.

[20] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[21] R. Badeau, G. Richard, and B. David, "Sliding window adaptive SVD algorithms," *IEEE Transactions on Signal Processing*, vol. 52, January 2004.

[22] P. Strobach, "Sliding window adaptive SVD using the unsymmetric householder partial compressor," *Signal Processing*, vol. 90, pp. 352–362, 2010.

Chapter 10

Energy-Based Multi-Speaker VAD

Energy-based multi-speaker voice activity detection with an ad hoc microphone array

Alexander Bertrand and Marc Moonen

Published in *Proc. of the IEEE International Conference on Audio, Speech and Signal Processing (ICASSP)*, Dallas, Texas USA, March 2010, pp. 85 - 88.

Contributions of first author

- literature study
- co-development of the energy-based VAD algorithm
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

In this paper, we propose an energy-based technique to track the power of multiple simultaneous speakers using an ad hoc microphone array with unknown microphone positions. By considering the short-term power of the microphone signals, the problem can be converted into a non-negative blind source separation (NBSS) problem. By exploiting the prior knowledge that the source signals are non-negative and well-grounded, very efficient algorithms can be used to solve this NBSS problem, based only on second order statistics. We provide simulation results that demonstrate the effectiveness of the presented algorithm.

10.1 Introduction

Many speech processing algorithms make use of a voice activity detector (VAD), i.e. an algorithm that decides whether a speech source is active or not. However, most VAD's assume that there is a single speech source, and are therefore unreliable in scenario's with multiple speakers. Furthermore, it is sometimes desirable that the VAD is able to distinguish between different speakers, e.g. in noise reduction algorithms where the noise signal is a speaker that interferes with the target speaker.

Since different speakers have different positions, the design of a multi-speaker VAD can rely on spatial information collected by multiple microphones. In [1], a far-field multi-speaker VAD is proposed for a microphone array with known microphone positions. The algorithm uses independent component analysis (ICA), K-means clustering, and beam-pattern analysis, which makes it very complex. In this paper, we use an energy-based approach that does not exploit any prior knowledge on the geometry of the array. It is suited for applications that make use of an ad hoc microphone array with widely spaced microphones (e.g. [2, 3]). This is for instance the case in video conferencing applications where each participant brings a device with built-in microphones, such as a laptop or PDA. Since most of these devices have WiFi technology, they can be linked to form an ad hoc network [2, 4]. The presented algorithm also does not assume any accurate synchronization between the microphone sampling clocks, which is very convenient, e.g. in the mentioned scenario with different devices. The VAD algorithm provides an estimate of the instantaneous power of each speech signal at each microphone.

By using short-term power measurements at the different microphones, the multi-speaker VAD problem can be converted into a blind source separation problem with non-negative sources, which can be solved efficiently with second order statistics only. We provide simulation results to demonstrate the

effectiveness of the presented algorithm.

10.2 Problem Statement and Data Model

Consider a scenario with N speakers and an ad hoc microphone array with J microphones. It is assumed that the microphones are spatially distributed such that the captured power from any speech source varies over the different microphones. We assume that the number of speakers N is known. If not, a prior step is needed to estimate N from the microphone signals, e.g. with PCA.

The N speakers produce the speech signals $\tilde{s}_n[t]$, $n = 1 \dots N$, where t denotes the sample time index. Let L denote the block length over which the instantaneous power of a signal is measured. We define the signal $s_n[k]$ as

$$s_n[k] = \frac{1}{L} \sum_{l=0}^{L-1} \tilde{s}_n[kL + l]^2 \tag{10.1}$$

i.e. $s_n[k]$ contains the instantaneous power of the signal $\tilde{s}_n$ at sample time kL (k is a frame index). The $s_n[k]$ signals are stacked in an N-dimensional vector $\mathbf{s}[k]$. In the sequel, we will use the symbol $\mathbf{s}$ without the index $[k]$ to refer to the underlying random process that generates the samples $\mathbf{s}[k]$. Similarly to (10.1), we define the instantaneous power in the j-th microphone signal as

$$y_j[k] = \frac{1}{L} \sum_{l=0}^{L-1} \tilde{y}_j[kL + l]^2 \tag{10.2}$$

where $\tilde{y}_j[t]$ denotes the j-th microphone signal. The $y_j[k]$ signals are stacked in a J-dimensional vector $\mathbf{y}[k]$.

If we assume that the signals $\tilde{s}_n$, $n = 1 \dots N$, are mutually independent, and if we neglect reverberation effects over the block edges, we can model $\mathbf{y}[k]$ according to

$$\mathbf{y}[k] \approx \mathbf{A}\mathbf{s}[k] \,, \quad \forall\, k \in \mathbb{N} \tag{10.3}$$

where $\mathbf{A}$ is a $J \times N$ mixing matrix, for which the element $[\mathbf{A}]_{jn}$ denotes the power attenuation between speaker n and microphone j. It is assumed that the mixing matrix $\mathbf{A}$ has full column rank. Notice that L yields a trade-off between time resolution and model mismatch. The larger the value of L, the better the approximation (10.3) holds, but the worse the time resolution becomes. Furthermore, if there is significant reverberation, this will also affect the approximation (10.3) (especially when L is small). However, we will demonstrate in Section 10.4 that our VAD algorithm is still able to provide satisfying results under limited reverberation.

Our goal is to find both $\mathbf{A}$ and $\mathbf{s}[k]$, which would allow us to compute the instantaneous power of each speaker at each microphone, and then to run a VAD for each speaker separately. Notice that this is a blind source separation (BSS) problem in which the source signals are non-negative. In [5], this is referred to as a non-negative independent component analysis (NICA) problem. Expression (10.3) can also be described in the frequency domain to allow for a multi-speaker VAD in separate frequency bins. However, as with all frequency domain BSS problems, a post-processing stage must then be added to resolve the permutation ambiguity between the different frequency bins. We will not take this into consideration in this paper.

Notice that we did not incorporate any noise in the data model. However, a localized noise source with non-stationary noise power, can readily be included in $\mathbf{s}$ as an additional source signal. On the other hand, diffuse noise with stationary power results in a constant noise floor, which can be easily estimated and subtracted from $\mathbf{y}[k]$. If required, noise estimation techniques, such as [6–8], can be used to track the power of a non-stationary diffuse noise. In the sequel, we assume that either noise power is subtracted from the signal $\mathbf{y}[k]$, or that localized noise sources are included in $\mathbf{s}$, so that (10.3) is satisfied. In Section 10.4, simulation results will demonstrate that the proposed VAD algorithm can still provide satisfying results when some residual noise power remains in $\mathbf{y}[k]$. The residual noise then results in a non-zero noise floor on the unmixed signals.

10.3 Solving the Non-Negative BSS Problem

10.3.1 Well-Grounded Sources

The prior knowledge on the non-negativity of the source signals in $\mathbf{s}$ can be exploited to design algorithms that are simpler compared to traditional ICA algorithms. In this paper, we exploit an additional assumption, i.e. the sources are assumed to be well-grounded [9]. This means that all sources have a non-zero pdf in any positive neighborhood of zero, i.e. $\forall\, \delta > 0$: $Pr(s_n < \delta) > 0$, for all source signals s_n, $n = 1 \ldots N$. Because speech signals typically have an on-off behavior, the signals s_n, $n = 1 \ldots N$, can be assumed to be well-grounded.

In [5], the non-negative principal component analysis (NPCA) algorithm is introduced, which solves NICA problems with well-grounded source signals. NPCA is a gradient-based learning algorithm, and its performance heavily depends on the chosen learning rate, as we will demonstrate in Section 10.4.

To avoid a step size search, we will use a multiplicative NICA (M-NICA) algorithm instead, which also exploits the well-grounded properties of the source signals [10]. M-NICA is a fixed-point type algorithm that has the facilitating

property that it does not depend on a user-defined learning rate. In the next section, we will briefly describe M-NICA. Even though the simulation results of our speaker dependent VAD are performed in a real-time context, we will describe the algorithm in batch-mode, for the sake of an easy exposition. For a detailed description of an adaptive sliding window implementation of M-NICA, we refer to [10].

10.3.2 The M-NICA Algorithm

Assuming that the source signals $\mathbf{s}$ are non-negative and well-grounded, it can be shown that it is sufficient to find an $N \times J$ unmixing matrix $\mathbf{K}$ such that the entries in the unmixed signal $\hat{\mathbf{s}} = \mathbf{K}\mathbf{y}$ are mutually uncorrelated and non-negative [9, 10]. Therefore, M-NICA is entirely based on second order statistics.

Assume we collect a $J \times M$ data matrix $\mathbf{Y}$ that contains M samples $\mathbf{y}[k]$, $k = 0 \ldots M-1$, in its columns. The goal is to find an $N \times M$ matrix $\mathbf{S} = \mathbf{K}\mathbf{Y}$ such that the rows of $\mathbf{S}$ are uncorrelated and only contain non-negative numbers. The following fixed-point type algorithm is used to generate such a matrix [10]:

1. *Initialization*:
   (a) $\forall\, n = 1 \ldots N, \forall\, m = 1 \ldots M :\ [\mathbf{S}]_{nm} \leftarrow [\mathbf{Y}]_{nm}$
   (b) Replace $\mathbf{Y}$ by its best rank N approximation by means of the singular value decomposition (SVD), i.e.

$$\{\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}\} \leftarrow \mathrm{SVD}\,(\mathbf{Y}) \tag{10.4}$$

$$\mathbf{Y} \leftarrow \overline{\mathbf{U}}\ \overline{\mathbf{\Sigma}}\ \overline{\mathbf{V}}^T \tag{10.5}$$

   where $\overline{\mathbf{\Sigma}}$ is the $N \times N$ diagonal matrix containing the N largest singular values[1] of $\mathbf{Y}$ on its diagonal, and where the corresponding left and right singular vectors are stored in the columns of $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$ respectively.
2. *Decorrelation step*:

$$\forall\, n = 1 \ldots N, \forall\, m = 1 \ldots M :$$

$$[\mathbf{S}^*]_{nm} \leftarrow [\mathbf{S}]_{nm}\, \frac{\left[\overline{\mathbf{S}}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\mathbf{S} + \mathbf{S}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\overline{\mathbf{S}} + \mathbf{\Lambda}_2\mathbf{S}\right]_{nm}}{\left[\overline{\mathbf{S}}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\overline{\mathbf{S}} + \mathbf{S}\mathbf{S}^T\mathbf{\Lambda}_1^{-1}\mathbf{S} + \mathbf{\Lambda}_2\overline{\mathbf{S}}\right]_{nm}} \tag{10.6}$$

   with

$$\overline{\mathbf{S}} = \frac{1}{M}\mathbf{S}\,\mathbf{1}_M\mathbf{1}_M^T \tag{10.7}$$

$$\mathbf{C}_s = (\mathbf{S} - \overline{\mathbf{S}})(\mathbf{S} - \overline{\mathbf{S}})^T \tag{10.8}$$

[1] Notice that, if noise were present, this step will remove some noise from the observations. In the noise-free case, $\mathbf{Y}$ has exactly N non-zero singular values.

$$\mathbf{\Lambda}_1 = \mathcal{D}\left\{\mathbf{C}_s\right\} \tag{10.9}$$

$$\mathbf{\Lambda}_2 = \mathcal{D}\left\{\left(\mathbf{\Lambda}_1^{-1}\mathbf{C}_s\right)^2\right\} \tag{10.10}$$

where $\mathbf{1}_M$ denotes an M-dimensional column vector in which each entry is 1, and where $\mathcal{D}\{\mathbf{X}\}$ denotes the operator that sets all off-diagonal elements of $\mathbf{X}$ to zero.

3. *Signal subspace projection step*:

$$\forall\, n = 1 \ldots N, \forall\, m = 1 \ldots M :$$

$$[\mathbf{S}]_{nm} \leftarrow \max\left(\left[\mathbf{S}^*\ \overline{\mathbf{V}}\ \overline{\mathbf{V}}^T\right]_{nm}, 0\right) \ . \tag{10.11}$$

4. Return to step 2.

In the decorrelation step (10.6), the elements of the matrix $\mathbf{S}$ are updated to decrease the mutual correlation between the rows of $\mathbf{S}$. Since $\mathbf{S}$ is initialized with non-negative elements, the decorrelation step (10.6) will preserve the non-negativity due to its multiplicative nature. However, the rows of the resulting matrix $\mathbf{S}$ are no longer in the signal subspace defined by the rows of $\mathbf{Y}$. Therefore, the matrix $\mathbf{S}$ is projected to the row space of $\mathbf{Y}$ in (10.11). For a more detailed derivation of the updating formulas, we refer to [10].

When a fixed point of (10.6)-(10.11) is found, the elements in each row of $\mathbf{S}$ correspond to samples of the unmixed signal $\hat{\mathbf{s}}[k]$. The mixing matrix $\hat{\mathbf{A}}$ that corresponds to $\hat{\mathbf{s}}$, can then be computed as

$$\hat{\mathbf{A}} = \mathbf{Y}\mathbf{S}^T\left(\mathbf{S}\mathbf{S}^T\right)^{-1} \ . \tag{10.12}$$

Notice that there always remains a permutation and scaling ambiguity between the columns of $\hat{\mathbf{A}}$ and the signals in $\hat{\mathbf{s}}$. However, in the multi-speaker VAD application, we are interested in the speech energy of each target speaker in each microphone signal. Let $v_{jn}[k]$ denote the speech energy of speaker n in microphone j at time instant k. Each value $v_{jn}[k]$, $j = 1 \ldots J$, $n = 1 \ldots N$, $k = 1 \ldots M$ can then be estimated as

$$\hat{v}_{jn}[k] = \left[\hat{\mathbf{A}}\right]_{jn} \hat{s}_n[k] \ . \tag{10.13}$$

10.4 Simulations

In this section, we provide simulation results for the multi-speaker VAD algorithm based on M-NICA. To compare, we also provide simulation results for the case where (10.3) is solved with NPCA, with different learning rates η (for a description of this algorithm, we refer to [5]). We simulate a cubical room

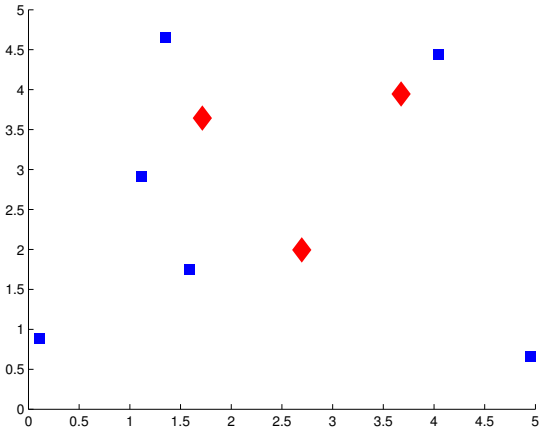


Figure 10.1: The acoustic scenario, containing $N = 3$ speakers ($\diamond$) and $J = 6$ microphones ($\square$).

(5m $\times$ 5m $\times$ 5m) with $N = 3$ randomly placed speakers ($\diamond$), all of them talking simultaneously, and $J = 6$ randomly placed microphones ($\square$), as shown in Fig. 10.1. The microphone signals are generated by means of the image method [11]. Unless stated otherwise, we compute the instantaneous power of the source signals and the microphone signals over time intervals of 30ms, which corresponds to $L = 480$ in (10.1)-(10.2), when the sampling frequency is $f_s = 16$kHz. This is the typical time duration for which a speech segment is assumed to be stationary. However, better performance can be obtained when a larger value is chosen for L, at the cost of a lower time resolution.

To produce a real-time output, a sliding window version of NPCA and M-NICA is implemented (see [10]). This means that the different iterations of the batch-mode versions of both algorithms are applied on a finite time window that shifts over the signals[2]. Samples that enter the window are first unmixed with an unmixing matrix that is computed from the previous samples in the window. The choice of the window length K introduces a trade-off: if K is chosen too small, then the independency assumption may be violated within one window length. On the other hand, a large value for K will affect the convergence time and the tracking capabilities of the VAD algorithm. In this experiment, the length of the sliding window is chosen to be $K = 200$, which is observed to provide satisfying results.

We use the mean of the signal-to-error ratios (SER) to assess the performance

[2]In our simulations, we perform one iteration for each sample shift of the window. However, to achieve faster convergence, multiple iterations can be performed in between each sample shift of the window. This is possible, since the window moves very slowly, i.e. every 30 ms.

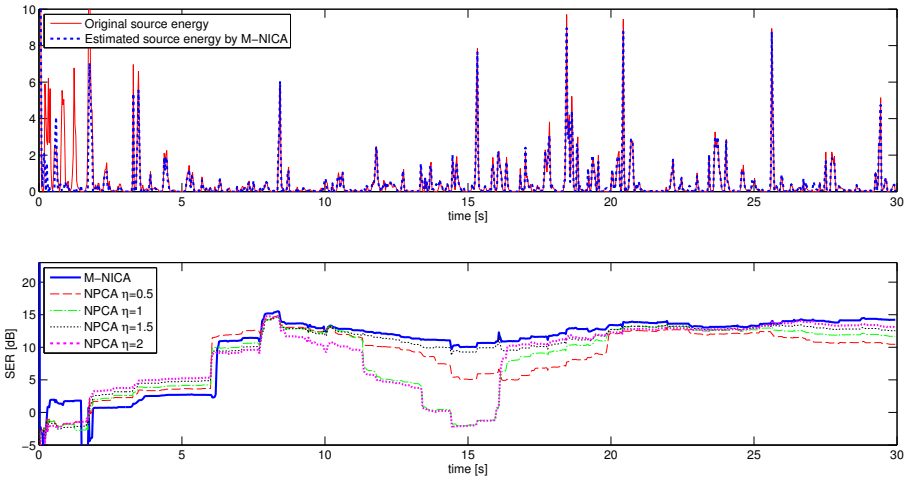


Figure 10.2: Reconstruction of the source energy in source 1 (above), and the corresponding SER (below).

of the multi-speaker VAD algorithm, i.e.

$$\text{SER} = \frac{1}{JN} \sum_{j,n} 10 \log_{10} \frac{\sum_k \hat{v}_{jn}[k]^2}{\sum_k (\hat{v}_{jn}[k] - [\mathbf{A}]_{jn} \, s_n[k])^2} \tag{10.14}$$

where $\hat{v}_{jn}[k]$ is defined by (10.13). Since we consider a sliding window implementation, the SER is computed over the K samples in the sliding window, and thus updated for each window shift.

Fig. 10.2 shows the original source energy of source 1. Furthermore, it shows the variation of the mean SER in the output of the VAD algorithm based on M-NICA and on NPCA for different values of η. It is observed that the performance of NPCA heavily depends on the choice of η. If η is chosen too small (e.g. $\eta = 0.5$), or too large (e.g. $\eta = 2$), the performance degrades significantly. The best overall performance is obtained for $\eta = 1.5$. M-NICA is observed to converge slightly slower than NPCA, but after convergence, it outperforms NPCA for any choice of η.

As mentioned in Section 10.2, reverberation affects the performance of the VAD algorithm, since approximation (10.3) then becomes less accurate. Fig. 10.3(a) plots the mean SER as a function of the reflection coefficient of the walls in the room (the SER is averaged over the last 10 seconds of the signal). For significant reverberance, the algorithm still manages to unmix the signals at a SER of approximately 8 dB, which is sufficient to make reliable VAD decisions. When L is doubled, i.e. $L = 960$, it is observed that the SER increases (at a cost of a lower time resolution).

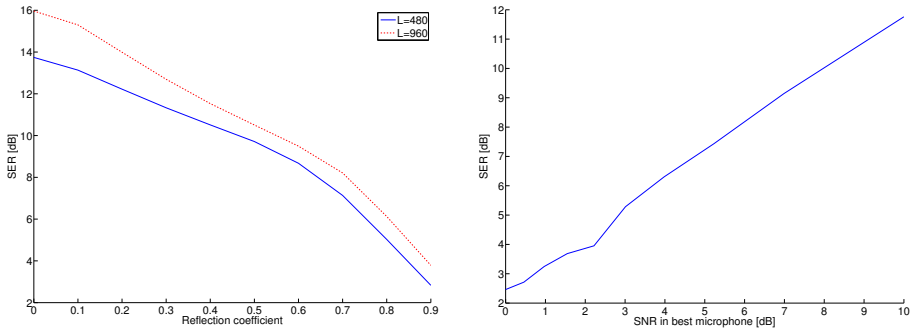


Figure 10.3: SER as a function of (a) reflection coefficient of the walls and (b) SNR.

As mentioned in Section 10.2, it is assumed that any noise power is removed from $\mathbf{y}[k]$. If some residual noise remains in $\mathbf{y}[k]$, the performance of the VAD algorithm decreases. We model residual noise by adding a stationary white noise source to each microphone signal $\tilde{y}_j[t]$, $j = 1 \ldots J$, resulting in a constant noise floor in $\mathbf{y}[k]$. Each microphone signal has an equal amount of residual noise, and no noise power is substracted from $\mathbf{y}[k]$. Fig. 10.3(b) shows the SER as a function of the signal-to-noise ratio (SNR) at the microphone with *highest* SNR. It is observed that the VAD algorithm still produces an output with satisfactory SER, as long as the SNR due to residual noise is sufficiently low. It should be noted that the decrease in SER is mainly due to a constant noise floor in the unmixed signals. The speech segments that have a higher power than this noise floor can still be detected, and are observed to be properly separated.

10.5 Conclusions

In this paper, we have presented a technique to track the power of multiple simultaneous speakers with an ad hoc microphone array with unknown microphone positions. Since the technique is energy-based, an accurate synchronization between the different microphone signals is not required. By using short-term power measurements at the different microphones, the multi-speaker VAD problem can be converted into a non-negative blind source separation (NBSS) problem, which can be solved efficiently based on second order statistics only. The effectiveness of the multi-speaker VAD has been demonstrated with adaptive sliding window simulations. The M-NICA algorithm presented here is observed to provide better overall results compared to NPCA [5], and has the additional advantage that it does not depend on a user-defined learning rate.

Bibliography

[1] P. B. S. Maraboina, D. Kolossa and R. Orglmeister, "Multi-speaker voice activity detection using ICA and beampattern analysis," in *Proc. of the European signal processing conference (EUSIPCO)*, Florence, Italy, 2006.

[2] M. Chen, Z. Liu, L.-W. He, P. Chou, and Z. Zhang, "Energy-based position estimation of microphones and speakers for ad hoc microphone arrays," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct. 2007, pp. 22–25.

[3] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435.

[4] Y. Jia, Y. Luo, Y. Lin, and I. Kozintsev, "Distributed microphone arrays for digital home and office," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2006.

[5] E. Oja and M. Plumbley, "Blind separation of positive sources using non-negative PCA," in *Proc. of the 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, Nara, Japan, April 2003.

[6] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 5, pp. 504–512, July 2001.

[7] N. Chatlani and J. J. Soraghan, "EMD-based noise estimation and tracking (ENET) with application to speech enhancement," in *Proc. of the European signal processing conference (EUSIPCO)*, Glasgow, Scotland, Aug. 2009.

[8] R. C. Hendriks, R. Heusdens, J. Jensen, and U. Kjems, "Fast noise PSD estimation with low complexity," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 3881–3884.

[9] M. Plumbley, "Conditions for nonnegative independent component analysis," *IEEE Signal Processing Letters*, vol. 9, no. 6, pp. 177–180, June 2002.

[10] A. Bertrand and M. Moonen, "Blind separation of non-negative source signals using multiplicative updates and subspace projection," *Signal Processing*, vol. 90, no. 10, pp. 2877–2890, October 2010.

[11] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," *Journal of the Acoustical Society of America*, vol. 65, pp. 943–950, April 1979.

Chapter 11

Link Failure Response and Sensor Subset Selection

Efficient sensor subset selection and link failure response for linear MMSE signal estimation in wireless sensor networks

Alexander Bertrand and Marc Moonen

Published in *Proc. of the European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, Aug. 2010, pp. 1092 - 1096.

Contributions of first author

- literature study
- co-derivation of the expressions for link failure response
- co-derivation of the expressions for utility tracking
- software implementation and computer simulations
- co-interpretation of simulation results
- text redaction and editing

Abstract

We consider two aspects of linear MMSE signal estimation in wireless sensor networks, i.e. sensor subset selection and link failure response. Both aspects are of great importance in low-delay signal estimation with high sampling frequency, where the estimator must be quickly updated in case of a link failure, and where sensor subset selection allows for a significant energy saving. Both problems are related since they require knowledge of the new optimal estimator when sensors are removed or added. We derive formulas to efficiently compute the optimal fall-back estimator in case of a link failure. Furthermore, we derive formulas to efficiently monitor the utility of each sensor signal that is currently used in the estimation, and the utility of extra sensor signals that are not yet used. Simulation results demonstrate that a significant amount of energy can be saved at the cost of a slight decrease in estimation performance.

11.1 Introduction

A wireless sensor network (WSN) consists of a large number of sensor nodes that are (usually randomly) deployed in an environment, and where each node has a wireless link to exchange data with neighbouring nodes [1]. The sensor nodes cooperate to perform a certain task such as signal estimation, detection, localization, etc. For this task, the data of the different sensors can be centralized in a so-called fusion center, or it can be partially or fully distributed over the different nodes in the network.

In this paper, we consider the case where a WSN is used for adaptive linear minimum mean squared error (MMSE) signal estimation, where the goal is to recover an unknown signal from noisy sensor observations. By using a WSN, a large area can be covered, yielding a significant amount of spatial information. This additional spatial information may result in an improved estimation performance compared to beamforming systems with small local arrays. However, WSN's often suffer from link failures, e.g. due to power shortage or interference in the wireless communication. For real-time signal estimation, the network must be able to swiftly adapt to these link failures to maintain sufficient estimation quality. In this paper, we provide an efficient procedure to compute the optimal fall-back estimators in case of a link failure, by exploiting the knowledge of the inverse sensor signal correlation matrix as used before the link failure. Due to the low complexity of the procedure, sensor nodes are able to react very quickly to link failures, even for high data rate applications such as in acoustic WSN's for speech enhancement [2, 3].

As the sensors in a WSN are usually battery-powered, energy efficiency is of great importance. To prolong the life-time of the network, it is therefore

important to only use those sensors that yield a significant contribution to the signal estimation process, while putting other sensors to sleep. This is the well known sensor subset selection problem. The sensor subset selection problem is also important in bandwidth constrained WSN's where each node can only transmit a subset of its available sensor signals. This is for instance the case in wireless binaural hearing aids with multiple microphones, where each hearing aid can only transmit a single microphone signal through the wireless link [3–5]. Notice that a quick link failure response is also an important aspect in this application.

Solving the sensor subset selection problem is generally computationally expensive due to its combinatorial nature. If the sensor signal statistics are known in advance, e.g. after an initial training phase, the sensor selection can be solved off line with unlimited power. However, in adaptive untrained WSN's the problem has to be solved during operation of the estimation algorithm. In this case, due to the limited power of a WSN, the sensor subset selection must be performed in an efficient way, generally yielding a suboptimal solution. We provide efficient closed-form formulas to compute the contribution of each sensor signal to the mean squared error (MSE) cost, i.e. the utility of each sensor signal, which can then be used in an adaptive greedy fashion to sequentially add or remove sensors in the estimation procedure. Simulation results demonstrate that a significant amount of energy can be saved in this way, at the cost of a slight decrease in estimation performance.

The paper is organized as follows. In Section 11.2, we briefly review the linear MMSE (LMSSE) signal estimation procedure, and address some of the aspects in adaptive LMMSE estimation. In Section 11.3, we derive a formula to efficiently compute the optimal fall-back estimator in case of a link failure. In Section 11.4, we describe an efficient procedure to monitor the utility of the sensor signals used in the current estimator, and to compute the potential utility of sensor signals not currently used. Simulation results are given in Section 11.5. Conclusions are drawn in Section 11.6.

11.2 Review of Linear MMSE Signal Estimation

In this section, we briefly review linear MMSE signal estimation, which is often used in signal enhancement [2–9]. We consider an ideal WSN with M sensors. Without loss of generality, we assume that all sensor signals are centralized in a fusion center. However, the results in this paper can be equally applied to the distributed case where each sensor node solves a local LMMSE problem, as

in [2–4, 8–11]. Sensor k collects observations of a complex[1] valued signal $y_k[t]$, where $t \in \mathbb{N}$ is the discrete time index. For the sake of an easy exposition, we will mostly omit the time index in the sequel. We assume that all sensor signals and the desired signal, are stationary and ergodic. In practice, the stationarity and ergodicity assumption can be relaxed to short-term stationarity and ergodicity, in which case the theory should be applied to finite signal segments that are assumed to be stationary and ergodic. We define $\mathbf{y}$ as the M-channel signal gathered at the fusion center in which all signals y_k, $\forall\, k \in \{1, \ldots, M\}$, are stacked.

The goal is to estimate a complex valued desired signal d from the sensor signal observations $\mathbf{y}$. We consider the general case where d is not an observed signal, i.e. it is assumed to be unknown, as it is the case in signal enhancement (e.g. in speech enhancement, d is the speech component in a noisy reference microphone signal). We consider LMMSE signal estimation, i.e. a linear estimator $\overline{d} = \hat{\mathbf{w}}^H \mathbf{y}$ that minimizes the MSE cost function

$$J(\mathbf{w}) = E\{|d - \mathbf{w}^H \mathbf{y}|^2\} \tag{11.1}$$

i.e.

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) \tag{11.2}$$

where $E\{.\}$ denotes the expected value operator and where the superscript H denotes the conjugate transpose operator[2]. It is noted that the above estimation procedure does not use multi-tap estimation, i.e. it does not explicitly exploit temporal correlation. However, this can be easily included by expanding $\mathbf{y}$ with delayed copies of itself. Expression (11.1) can also be viewed as a frequency domain description, such that it defines an estimator for a specific frequency bin. When (11.2) is solved for each individual frequency bin, this is equivalent to multi-tap estimation. In its multi-tap form, the solution of (11.2) is often referred to as a multi-channel Wiener filter (MWF) [6, 7].

Assuming that the correlation matrix $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$ has full rank[3], the unique solution of (11.2) is [12]:

$$\boxed{\hat{\mathbf{w}} \;=\; \mathbf{R}_{yy}^{-1}\mathbf{r}_{yd}} \tag{11.3}$$

with $\mathbf{r}_{yd} = E\{\mathbf{y}d^*\}$, where d^* denotes the complex conjugate of d. The MMSE corresponding to this optimal estimator is

$$J(\hat{\mathbf{w}}) = P_d - \mathbf{r}_{yd}^H \mathbf{R}_{yy}^{-1} \mathbf{r}_{yd} \tag{11.4}$$

[1] Throughout this paper, all signals are assumed to be complex valued to permit frequency-domain descriptions, e.g. when using a short-time Fourier transform (STFT).

[2] In the sequel, we use the superscript T to denote the normal transpose, i.e. without conjugation.

[3] This assumption is mostly satisfied in practice because of a noise component at every sensor that is independent of other sensor signals, e.g. thermal noise. If not, pseudo-inverses should be used.

$$= P_d - \mathbf{r}_{yd}^H \hat{\mathbf{w}} \tag{11.5}$$

with $P_d = |d|^2$. Based on the assumption that the signals are ergodic, $\mathbf{R}_{yy}$ can be adaptively estimated from the sensor signal observations by time averaging. Since d is assumed to be unknown, the estimation of the correlation vector $\mathbf{r}_{yd}$ has to be done indirectly, based on application-specific strategies, e.g. by exploiting the on-off behavior of the target signal (as often done in speech enhancement [2, 3, 6]), by periodic broadcasts of known training sequences, or by incorporating prior knowledge on the signal statistics in case of partially static scenarios [10]. In the sequel, we assume that both $\mathbf{R}_{yy}$ and $\mathbf{r}_{yd}$ are known, or that both can be estimated adaptively.

Notice that the inverse of $\mathbf{R}_{yy}$ is required for the computation of (11.3), rather than the matrix $\mathbf{R}_{yy}$ itself. When M is large, computing this matrix inverse is however computationally expensive, i.e. $O(M^3)$, and should be avoided in adaptive applications with high data rates. Let $\mathbf{R}_{yy}[t]$ denote the estimate of $\mathbf{R}_{yy}$ at time t. Instead of updating $\mathbf{R}_{yy}[t]$ for each new sample $\mathbf{y}[t]$, and recomputing the full matrix inversion $\mathbf{R}_{yy}^{-1}[t] = (\mathbf{R}_{yy}[t])^{-1}$, the previous matrix $\mathbf{R}_{yy}^{-1}[t-1]$ is directly updated. For example, $\mathbf{R}_{yy}$ is often estimated by means of a forgetting factor $0 < \lambda < 1$, i.e.

$$\mathbf{R}_{yy}[t] = \lambda \mathbf{R}_{yy}[t-1] + (1-\lambda)\mathbf{y}[t]\mathbf{y}[t]^H \ . \tag{11.6}$$

In this case, $\mathbf{R}_{yy}^{-1}[t]$ can be recursively updated by means of the matrix inversion lemma, a.k.a. the Woodbury identity [12], yielding

$$\mathbf{R}_{yy}^{-1}[t] = \frac{1}{\lambda}\mathbf{R}_{yy}^{-1}[t-1] - \frac{\mathbf{R}_{yy}^{-1}[t-1]\mathbf{y}[t]\mathbf{y}[t]^H\mathbf{R}_{yy}^{-1}[t-1]}{\frac{\lambda^2}{1-\lambda} + \lambda\mathbf{y}[t]^H\mathbf{R}_{yy}^{-1}[t-1]\mathbf{y}[t]} \tag{11.7}$$

which has a computational complexity of $O(M^2)$. It is noted that, when (11.7) is used to update $\mathbf{R}_{yy}^{-1}[t]$, the correlation matrix $\mathbf{R}_{yy}[t]$ itself does not need to be kept in memory.

11.3 Link Failure Response

Now assume a link failure with sensor k during operation of the estimation process. This means that the fusion center now only has access to the $(M-1)$-channel signal $\mathbf{y}_{-k}$, which is defined as the vector $\mathbf{y}$ with y_k removed. In this case, the optimal LMMSE solution is

$$\hat{\mathbf{w}}_{-k} = \mathbf{R}_{yy-k}^{-1}\mathbf{r}_{yd-k} \tag{11.8}$$

where $\mathbf{R}_{yy-k} = E\{\mathbf{y}_{-k}\mathbf{y}_{-k}^H\}$ and $\mathbf{r}_{yd-k} = E\{\mathbf{y}_{-k}d^*\}$. Hence, when the wireless link of sensor k breaks down, estimator $\hat{\mathbf{w}}$ (11.3) becomes suboptimal, and

should be replaced by (11.8). However, computing (11.8) requires knowledge of $\mathbf{R}_{yy-k}^{-1}$, which is not directly available. If $\mathbf{R}_{yy}$ were kept in memory, it is possible to invert its submatrix $\mathbf{R}_{yy-k}$ to obtain $\mathbf{R}_{yy-k}^{-1}$. However, this has a large computational cost when M is large, i.e. $O\left(M^3\right)$.

In the sequel, we derive an efficient formula to compute $\hat{\mathbf{w}}_{-k}$ without knowledge of $\mathbf{R}_{yy}$, and without explicitly computing matrix inversions. As explained in Section 11.2, we only assume that the previous estimate of $\mathbf{R}_{yy}^{-1}$ is known. For the sake of an easy exposition, but without loss of generality, we assume that $k = M$, i.e. the last element of $\mathbf{y}$ is removed. We consider a block partitioning of the inverse correlation matrix

$$\mathbf{R}_{yy}^{-1} = \left[\begin{array}{c|c} \mathbf{A}_M & \mathbf{b}_M \\ \hline \mathbf{b}_M^H & Q_M \end{array}\right] \tag{11.9}$$

where $\mathbf{A}_M$ is an $(M-1) \times (M-1)$ matrix, $\mathbf{b}_M$ is an $(M-1)$-dimensional vector, and Q_M is a real-valued scalar. We define a similar partitioning of the corresponding (and also assumed known) optimal LMMSE estimator $\hat{\mathbf{w}}$ (11.3) before the link failure with sensor M:

$$\hat{\mathbf{w}} = \left[\begin{array}{c} \mathbf{c}_M \\ \hline W_M \end{array}\right] \tag{11.10}$$

where $\mathbf{c}_M$ denotes the subvector containing the first $(M-1)$ elements of $\hat{\mathbf{w}}$, and where W_M defines the scaling that is applied to the sensor signal M in the estimation process. Similar to (11.9), we define the following block partitioning of the correlation matrix

$$\mathbf{R}_{yy} = \left[\begin{array}{c|c} \mathbf{R}_{yy-M} & \mathbf{r}_M \\ \hline \mathbf{r}_M^H & P_M \end{array}\right] \tag{11.11}$$

where $\mathbf{r}_M$ is an $(M-1)$-dimensional vector, and where P_M is a real-valued scalar, corresponding to the power of the signal y_M. By using the matrix inversion lemma, one can verify that the inverse of this block matrix is:

$$\mathbf{R}_{yy}^{-1} = \left[\begin{array}{c|c} \mathbf{R}_{yy-M}^{-1} + \alpha_M \mathbf{v}_M \mathbf{v}_M^H & -\alpha_M \mathbf{v}_M \\ \hline -\alpha_M \mathbf{v}_M^H & \alpha_M \end{array}\right] \tag{11.12}$$

with

$$\mathbf{v}_M = \mathbf{R}_{yy-M}^{-1} \mathbf{r}_M \tag{11.13}$$

$$\alpha_M = \frac{1}{P_M - \mathbf{r}_M^H \mathbf{v}_M} \, . \tag{11.14}$$

By comparing (11.9) and (11.12), we find that

$$\boxed{\mathbf{R}_{yy-M}^{-1} = \mathbf{A}_M - \tfrac{1}{Q_M} \mathbf{b}_M \mathbf{b}_M^H} \tag{11.15}$$

and therefore the optimal fall-back estimator is

$$\hat{\mathbf{w}}_{-M} = \left(\mathbf{A}_M - \frac{1}{Q_M} \mathbf{b}_M \mathbf{b}_M^H \right) \mathbf{r}_{yd-M} \ . \tag{11.16}$$

By plugging (11.9) and (11.10) into (11.3) we obtain

$$\mathbf{c}_M = \mathbf{A}_M \mathbf{r}_{yd-M} + R_{y_M d} \mathbf{b}_M \tag{11.17}$$

$$W_M = \mathbf{b}_M^H \mathbf{r}_{yd-M} + Q_M R_{y_M d} \tag{11.18}$$

where $R_{y_M d}$ denotes the last element of the correlation vector $\mathbf{r}_{yd}$. When comparing (11.16) with (11.17)-(11.18), we find with some straightforward algebraic manipulation that the optimal fall-back estimator can be readily computed as

$$\boxed{\hat{\mathbf{w}}_{-M} = \mathbf{c}_M - \tfrac{W_M}{Q_M} \mathbf{b}_M \ .} \tag{11.19}$$

Since all variables in (11.19) are directly available, this allows a very efficient computation, i.e. $O(M)$.

Remark: The above formulas can also be used in the case where an additional sensor signal becomes available. That is, formulas (11.12)-(11.14) can be used to efficiently compute the new inverse correlation matrix $\mathbf{R}_{yy}^{-1}$ when sensor M is added in the estimation process. We will return to this in Section 11.4.2.

11.4 Sensor Subset Selection

Assume that we have an optimal M-channel LMMSE estimator $\hat{\mathbf{w}}$. The goal is now to efficiently monitor the utility of each sensor signal, i.e. we wish to identify how much the MSE cost (11.1) increases when a specific sensor is removed from the signal estimation procedure (sensor deletion), or how much the MSE cost decreases if a specific additional sensor would be included in the estimator (sensor addition). We will refer to this MSE cost decrease or increase as the 'utility' of the sensor signal. To allow monitoring this utility, we want to be able to compute it in an efficient way, i.e. without explicit matrix inversions and without actually computing the optimal estimator for all possible scenarios. In the case of sensor deletion, we will show that the utility of each sensor can be monitored at a computational cost which is negligible compared to the estimator update based on (11.7). In the case of sensor addition, the cost of monitoring the potential utility of N extra sensors is more significant, i.e. N times the cost of (11.7).

11.4.1 Sensor Deletion

For sensor deletion, the goal is to monitor the contribution of each sensor to the current MSE cost. The utility of sensor k is defined as

$$U_k = J(\hat{\mathbf{w}}_{-k}) - J(\hat{\mathbf{w}}) \; . \tag{11.20}$$

The goal is to efficiently compute U_k, $\forall\, k \in \{1, \ldots, M\}$. From (11.5), and with the notations[4] introduced in Section 11.3, we find that

$$U_M = \mathbf{r}_{yd}^H \hat{\mathbf{w}} - \mathbf{r}_{yd-M}^H \hat{\mathbf{w}}_{-M} \; . \tag{11.21}$$

By using (11.19), and by using the partitioning of $\hat{\mathbf{w}}$ as defined in (11.10), we can rewrite (11.21) as

$$U_M = R_{y_M d}^* W_M + \frac{W_M}{Q_M} \mathbf{r}_{yd-M}^H \mathbf{b}_M \; . \tag{11.22}$$

From (11.18), we find that

$$\mathbf{r}_{yd-M}^H \mathbf{b}_M = W_M^* - Q_M R_{y_M d}^* \; . \tag{11.23}$$

By substituting (11.23) in (11.22), we find that

$$U_M = \frac{1}{Q_M} |W_M|^2 \; . \tag{11.24}$$

To monitor the utility of all the sensors simultaneously, i.e. the vector $\mathbf{u} = [U_1 \ldots U_M]^T$, it is thus sufficient to monitor the squared components of the current estimator $\hat{\mathbf{w}}$, normalized with the diagonal elements of the *inverted* correlation matrix $\mathbf{R}_{yy}^{-1}$, i.e.

$$\boxed{\mathbf{u} = \mathbf{\Lambda}^{-1} |\hat{\mathbf{w}}|^2} \tag{11.25}$$

with

$$\mathbf{\Lambda} = \mathcal{D}\{\mathbf{R}_{yy}^{-1}\} \tag{11.26}$$

where the operator $\mathcal{D}\{\mathbf{X}\}$ sets all off-diagonal elements of $\mathbf{X}$ to zero, and where the element-wise operator $|\mathbf{x}|^2$ replaces all elements in the vector $\mathbf{x}$ with their squared absolute value. Expression (11.25) is computationally efficient, i.e. $O(M)$. Therefore, the complexity of monitoring the utility of each sensor is negligible compared to the estimator update based on (11.7). When the utility of a certain sensor drops below a certain threshold, this sensor can be put to sleep, and the new optimal LMMSE estimator can then be readily computed as in expression (11.19). The reduced inverse correlation matrix can be readily computed with (11.15), which is then required for future estimator updates with (11.7).

[4] Again, we assume that $k = M$, without loss of generality.

11.4.2 Sensor Addition

Assume that we have an optimal MMSE estimator $\hat{\mathbf{w}}$ that linearly combines M sensor signals, and that a set of N additional sensor signals is available. Which one of these sensor signals would bring the greatest benefit to the estimator?

To use the results from Section 11.3, we assume that the current estimator is the $(M-1)$-channel estimator $\hat{\mathbf{w}}_{-M}$. The utility of adding sensor M to the estimation process, i.e. the decrease in MSE cost, is again given by (11.20). However, expression (11.25) cannot be used in this case, since W_M is not known. Indeed, this time only $\mathbf{R}_{yy-M}^{-1}$ is kept in memory, instead of $\mathbf{R}_{yy}^{-1}$. This makes the problem of sensor addition substantially different from sensor deletion.

By using (11.4), we can rewrite (11.20) as

$$U_M = \mathbf{r}_{yd}^H \mathbf{R}_{yy}^{-1} \mathbf{r}_{yd} - \mathbf{r}_{yd-M}^H \mathbf{R}_{yy-M}^{-1} \mathbf{r}_{yd-M} \ . \tag{11.27}$$

By using expression (11.12), we find that

$$\begin{aligned} \mathbf{r}_{yd}^H \mathbf{R}_{yy}^{-1} \mathbf{r}_{yd} = \quad & \mathbf{r}_{yd-M}^H \left(\mathbf{R}_{yy-M}^{-1} + \alpha_M \mathbf{v}_M \mathbf{v}_M^H \right) \mathbf{r}_{yd-M} \\ & -2\alpha_M \mathcal{R}\{\mathbf{r}_{yd-M}^H \mathbf{v}_M\} + \alpha_M |R_{y_M d}|^2 \end{aligned} \tag{11.28}$$

where $\mathcal{R}\{X\}$ denotes the real part of X. By substituting (11.28) in (11.27), we find that the utility of sensor M can be computed as

$$U_M = \alpha_M |\mathbf{r}_{yd-M}^H \mathbf{v}_M - R_{y_M d}|^2 \ . \tag{11.29}$$

The computational complexity is $O(M^2)$, which is the same order of magnitude as the computation of the estimator update based on (11.7). Notice that, as opposed to the sensor deletion case, we now do need the cross correlation between the currently used sensor signals, and the added sensor signal y_M (used in the computation of $\mathbf{v}_M$, as given in (11.13)). This cannot be circumvented because the current optimal estimator only uses $\mathbf{R}_{yy-M}^{-1}$, which indeed does not incorporate any statistics of y_M.

Let us now consider the general case where N extra sensor signals become available. Define $\mathbf{y}_c$ as the stacked vector of the M sensor signals that are currently used in the estimation process, and define $\mathbf{y}_e$ as the stacked N-channel signal that contains the N extra sensor signals that can be added to the estimation process. We redefine $\mathbf{R}_{yy}$ as

$$\mathbf{R}_{yy} = \left[\begin{array}{c|c} \mathbf{R}_{y_c y_c} & \mathbf{R}_{y_c y_e} \\ \hline \mathbf{R}_{y_c y_e}^H & \mathbf{R}_{y_e y_e} \end{array} \right] \tag{11.30}$$

where $\mathbf{R}_{y_c y_c} = E\{\mathbf{y}_c \mathbf{y}_c^H\}$, $\mathbf{R}_{y_c y_e} = E\{\mathbf{y}_c \mathbf{y}_e^H\}$, and $\mathbf{R}_{y_e y_e} = E\{\mathbf{y}_e \mathbf{y}_e^H\}$. We assume that $\mathbf{R}_{y_c y_c}^{-1}$ is kept in memory, since this was used in the computation of the current optimal estimator. We also assume that $\mathbf{R}_{y_c y_e}$ is available,

i.e. the cross correlation between the currently used sensor signals and the extra sensor signals, which can be estimated through time averaging. Finally, we assume that the power of each additional sensor signal is known, i.e. the diagonal elements of $\mathbf{R}_{y_e y_e}$.

Similar to (11.29), we can compute the vector $\mathbf{u} = [U_1 \dots U_N]^T$, which gives the utility of each additional sensor signal:

$$\boxed{\mathbf{u} = \mathbf{\Sigma}^{-1} |\mathbf{V}^T \mathbf{r}^*_{y_c d} - \mathbf{r}_{y_e d}|^2} \tag{11.31}$$

where

$$\mathbf{V} = \mathbf{R}^{-1}_{y_c y_c} \mathbf{R}_{y_c y_e} \tag{11.32}$$

$$\mathbf{\Sigma} = \mathcal{D}\{\mathbf{R}_{y_e y_e}\} - \mathcal{D}\{\mathbf{R}^H_{y_c y_e} \mathbf{V}\} \tag{11.33}$$

and where $\mathbf{r}_{y_c d} = E\{\mathbf{y}_c d^*\}$ and $\mathbf{r}_{y_e d} = E\{\mathbf{y}_e d^*\}$. The computational complexity of (11.32) is the dominant part, which makes the total computational complexity $O(M^2 N)$.

Let $U_k = \max_{i \in \{1,\dots,N\}} U_i$, which means that sensor k will be selected as providing the most useful additional sensor signal. To incorporate sensor signal y_k in the estimation procedure, the inverse correlation matrix $\mathbf{R}^{-1}_{y_c y_c}$ should be replaced with $\mathbf{R}^{-1}_{y_c y_c + k} = E\{\left[\mathbf{y}_c^T y_k\right]^T \left[\mathbf{y}_c^H y_k^*\right]\}^{-1}$, which can be computed similarly to (11.12), i.e.

$$\boxed{\mathbf{R}^{-1}_{y_c y_c + k} = \left[\begin{array}{c|c} \mathbf{R}^{-1}_{y_c y_c} + \frac{1}{S_k}\mathbf{v}_k \mathbf{v}_k^H & -\frac{1}{S_k}\mathbf{v}_k \\ \hline -\frac{1}{S_k}\mathbf{v}_k^H & \frac{1}{S_k} \end{array}\right]} \tag{11.34}$$

where $\mathbf{v}_k$ denotes the k-th column of $\mathbf{V}$, and where S_k denotes the k-th diagonal element of $\mathbf{\Sigma}$. This has computational complexity $O(M^2)$, which is the same as the complexity of an estimator update according to (11.7). The new optimal LMMSE estimator can then be computed as

$$\boxed{\hat{\mathbf{w}}_{+k} = \mathbf{R}^{-1}_{y_c y_c + k} \begin{bmatrix} \mathbf{r}_{y_c d} \\ R_{y_k d} \end{bmatrix}} \tag{11.35}$$

where $R_{y_k d}$ denotes the k-th entry in $\mathbf{r}_{y_e d}$.

11.4.3 Greedy Sensor Subset Selection

The formulas (11.25) and (11.31) can be readily used in a greedy approach to efficiently determine a subset of sensor signals that yields a good estimator. This can be done in two different ways (with generally different end results). In the case of sensor addition, one starts by selecting the single sensor signal

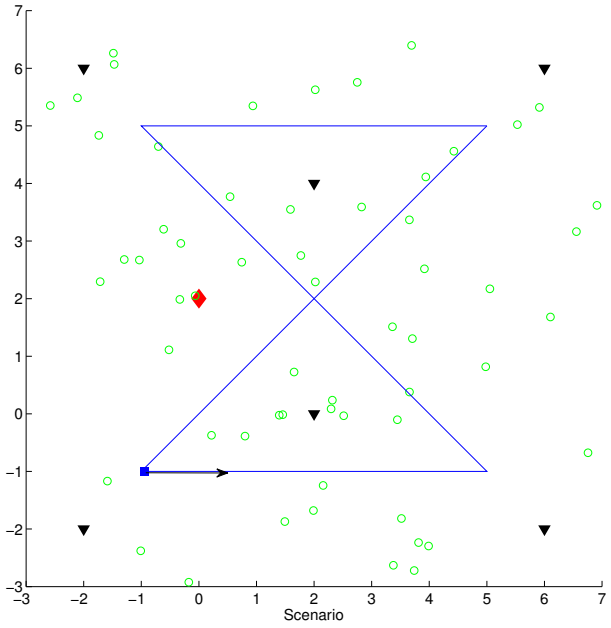


Figure 11.1: The simulated scenario, containing $M = 60$ sensors ($\circ$) with one reference sensor ($\diamond$), 6 noise sources ($\bigtriangledown$) and one moving target source ($\square$).

which results in the best single-channel estimator, and then in each cycle the sensor with highest utility is added to the estimation process (forward mode). In the case of sensor deletion, one starts by computing the optimal estimator using all sensor signals, and then in each cycle the sensor with lowest utility is deleted (backward mode). An adaptive greedy sensor subset selection (AGSSS) algorithm is described in more detail in the next section.

11.5 Simulations

In this section, we present simulation results of an adaptive LMMSE signal estimation algorithm with adaptive greedy sensor subset selection. The scenario is depicted in Fig. 11.1. This is a toy scenario, and we do not attempt to model any practical setting or application. All signals are sampled with a sampling rate of 8kHz. The target source ($\square$) moves at a speed of 0.5 m/s over the path indicated by the straight lines, and stops for 5 seconds at each corner. The target source signal is white and has a Gaussian distribution. There are six localized white Gaussian noise sources ($\bigtriangledown$) present, each with 25% of the power of the target source[5]. The WSN contains $M = 60$ randomly

[5]This is an arbitrary choice that yields practical SNR's at the sensors.

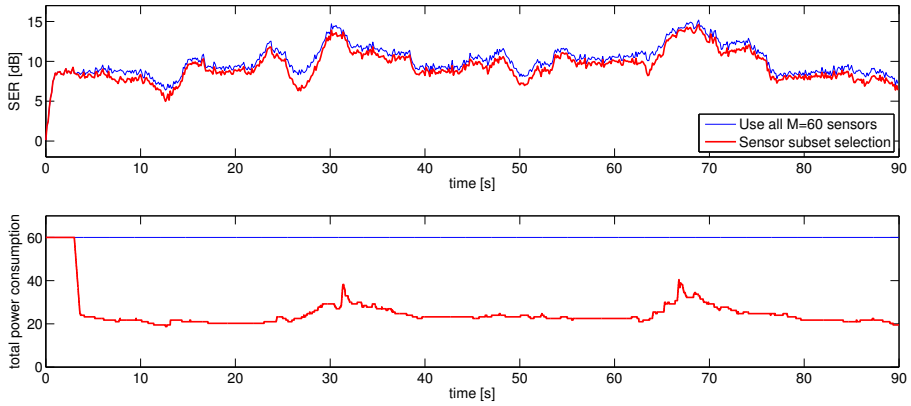


Figure 11.2: SER vs. time (above), and the corresponding total power consumed in the WSN (below).

placed sensors ($\circ$), with one reference sensor ($\diamond$). The goal is to estimate the target source signal as it is sensed by this reference sensor (denoted by d). In addition to the spatially correlated noise, independent white Gaussian sensor noise, with 5% of the power of the target source, is added to each sensor signal. The individual signals originating from the target sources and the noise sources that are collected by a specific sensor are attenuated in power and summed. The attenuation factor of the signal power is $\frac{1}{r}$, where r denotes the distance between the source and the sensor. We assume that there is no time delay in the transmission path between the sources and the sensors[6]. The estimation performance will be assessed based on the instantaneous signal-to-error ratio, computed over $L = 1000$ samples:

$$SER[t] = 10\log_{10}\left(\frac{\sum_{k=t-L+1}^{t} d[k]^2}{\sum_{k=t-L+1}^{t}(d[k]-\overline{d}[k])^2}\right) . \tag{11.36}$$

The inverse correlation matrix $\mathbf{R}_{yy}^{-1}$ is updated according to (11.7) with a forgetting factor $\lambda = 0.9995$. The correlation vector $\mathbf{r}_{yd}$ is updated with the same forgetting factor. We use the clean desired signal d in the estimation of $\mathbf{r}_{yd}$, to isolate estimation errors. Notice that in practice, application-specific techniques are required to estimate this vector if d is not directly available[7] (see e.g. [2, 3]). During the first 3 seconds, the estimation algorithm estimates the

[6]Since there are no time delays, the spatial information is purely energy based in this case. Therefore, the fusion center cannot perform any beamforming towards specific locations by exploiting different delay paths between sources and sensors.

[7]In some applications, the signal d is directly available at certain moments in time. For example, in communications applications, known training sequences can be used to estimate $\mathbf{r}_{yd}$ during periodic training intervals.

required statistics of all sensor signals, and computes the optimal M-channel LMMSE estimator $\hat{\mathbf{w}}$ (11.3). After 3 seconds, an adaptive greedy sensor subset selection (AGSSS) algorithm starts running simultaneously with the adaptive LMMSE estimation process.

In the AGSSS, the utility of each currently used sensor signal is tracked using (11.25). If a sensor's utility drops below 1% of the MSE cost of the current estimator (computed with (11.5)), the sensor is put to sleep, and the inverse correlation matrix and the estimator are updated according to (11.15) and (11.19), respectively. Notice that this corresponds to a decrease in SER of maximum $10\log_{10}(1.01) = 0.043$dB for each sensor that is removed. The sensors that are put to sleep transmit their sensor signal only 25% of the time, reducing their power consumption with 75 %. The reason why sleeping sensors still transmit data, is to estimate the required statistics to compute their utility, based on (11.31). Once their utility exceeds 5% of the MSE cost of the current estimator, they are added again to the estimation process. This corresponds to an increase in SER of at least $-10\log_{10}(0.95) = 0.22$dB for each sensor that is added. The inverse correlation matrix and the estimator are updated according to (11.34) and (11.35), respectively.

The instantaneous SER of the resulting time-varying estimator is shown in Fig. 11.2, together with a plot of the total power consumption summed over all sensors. The active sensors have a power consumption of 1, and sleeping sensors have a power consumption of 0.25 (these numbers are unitless since they are not based on actual physical power consumption). The SER and power consumption of the optimal estimator that uses all $M = 60$ sensors is also added as a reference, which we will refer to as the full estimator. We observe that, due to the sensor subset selection, the SER slightly drops compared to the full estimator (on average, this is a decrease of 0.56 dB). However, due to the power saving of the sleeping sensors, the total average power consumption is only 41 % of the total power consumption of the full estimator. The average number of active sensors is 13.

11.6 Conclusions

In this paper, we have considered two aspects in linear MMSE signal estimation in wireless sensor networks, i.e. sensor subset selection and link failure response. We have first derived an efficient formula to compute the optimal fall-back estimator when the wireless link of one of the sensors fails. High efficiency is achieved by exploiting the knowledge of the inverse correlation matrix as used before the link failure. We have then derived an efficient formula to monitor the utility of each sensor signal in the current estimation process, which can be used for sensor deletion. We have also derived a formula to efficiently compute the

potential utility of sensors that are not yet used in the estimation process, which can then be used for sensor addition. Both formulas can be used to perform an adaptive greedy sensor subset selection procedure. Simulation results of this greedy procedure in an adaptive LMMSE estimation algorithm demonstrate that a significant amount of energy can be saved, at the cost of a slight decrease in estimation performance.

Bibliography

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2033–2036, 2001.

[2] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435.

[3] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, pp. 38–51, Jan. 2009.

[4] S. Srinivasan, "Noise reduction in binaural hearing aids: Analyzing the benefit over monaural systems," *The Journal of the Acoustical Society of America*, vol. 124, no. 6, pp. 353–359, 2008.

[5] S. Srinivasan and A. C. den Brinker, "Rate-constrained beamforming in binaural hearing aids," *EURASIP Journal on Advances in Signal Processing, vol. 2009, Article ID 257197, 9 pages, 2009. doi:10.1155/2009/257197.*

[6] 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, Sep 2002.

[7] J. Chen, J. Benesty, Y. Huang, and S. Doclo, "New insights into the noise reduction Wiener filter," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1218 –1234, July 2006.

[8] A. Bertrand and M. Moonen, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan*, April 2009.

[9] ——, "Distributed adaptive node-specific MMSE signal estimation in sensor networks with a tree topology," *Proc. of the European signal processing conference (EUSIPCO), Glasgow - Scotland*, August 2009.

[10] ——, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part I: sequential node updating," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5277–5291, 2010.

[11] ——, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – part II: simultaneous & asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5292–5306, 2010.

[12] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.

Part V

Conclusions

Chapter 12

Conclusions and Suggestions for Future Research

12.1 Summary and Conclusions

In this thesis, we have considered several open estimation problems in the domain of wireless sensor networks (WSNs). In almost all proposed methods, the goal was to obtain the same estimation performance as in a centralized algorithm where all the data is assumed to be available in a fusion center. We have distinguished between two different types of distributed estimation problems: signal estimation and parameter estimation.

In the context of distributed signal estimation in WSNs, we have proposed a blind adaptive distributed signal estimation algorithm in which each node estimates a different signal. This algorithm was referred to as 'distributed adaptive node-specific signal estimation' (DANSE). In **Chapter 2**, the DANSE algorithm for fully connected WSNs has been described for the case where nodes update their local estimation variables in a sequential round-robin fashion. If the desired signals of all the nodes share a common low-dimensional signal subspace, it has been shown that DANSE significantly reduces the exchange of data between nodes, while still obtaining an optimal estimator in each node, as if all nodes have access to all the sensor signal observations in the network. A practical adaptive implementation of DANSE has been described and simulated, demonstrating the tracking capabilities of the algorithm in a dynamic scenario. It has been observed that the DANSE algorithm is more robust to

estimation errors in the correlation matrices, compared to its centralized equivalent. A modified DANSE algorithm has been studied in **Chapter 3**, where an improved tracking performance is obtained by letting nodes update simultaneously or asynchronously. We have pointed out that the convergence and optimality of the DANSE algorithm can no longer be guaranteed in this case. We have then modified the DANSE algorithm with a relaxation operation to restore convergence and optimality under simultaneous and asynchronous updating. Convergence of the new algorithm is proven if the relaxation parameter satisfies certain conditions. Simulations have demonstrated that a simplified version of the new algorithm can also be used, with a lower computational load, while maintaining convergence. Since all nodes can estimate the required statistics in parallel, the new algorithm adapts faster than the original DANSE algorithm, especially so when the number of nodes is large. In **Chapter 4**, both versions of the DANSE algorithm (sequential and simultaneous node-updating) have then been applied in a speech enhancement context. To this end, the algorithm has been extended to a more robust version, to avoid numerically ill-conditioned quantities that often arise in such practical settings. This new version has been referred to as R-DANSE, and its convergence has been proven. Batch-mode experiments have shown that R-DANSE significantly outperforms DANSE in practical settings. Additional tests have been performed to quantify the influence of several parameters, such as the DFT size and TDOA's or delays within the communication link.

In **Chapter 5**, the DANSE algorithm has been extended to operate in WSNs with a tree topology, hence relaxing the constraint that the network needs to be fully connected. It is argued that feedback is to be avoided, when a straightforward modification of DANSE is applied for operation in simply connected networks, since it harms the convergence and optimality properties of the algorithm. Direct feedback can be avoided easily, whereas indirect feedback is more difficult to remove in a network topology that has cycles. A tree topology is then a natural choice, since it has no cycles, for which the T-DANSE algorithm has been derived. A condition is given on the updating order of the nodes to guarantee convergence to the optimal estimators. Simulations have shown that the condition on the updating order of the nodes is sufficient but not necessary, although convergence is faster if the condition is satisfied.

Finally, in **Chapter 6**, the DANSE algorithm has been extended with node-specific linear constraints, yielding an optimal node-specific linearly-constrained minimum variance beamformer in each node. Convergence and optimality of the algorithm has been formally proven. We have experimentally verified that convergence can be obtained when nodes update simultaneously, if relaxation is applied, similar to the unconstrained DANSE algorithm. We have provided simulation results that demonstrate the effectiveness of the algorithm for speech enhancement (and separation) in a wireless acoustic sensor network.

In the second part of this thesis, we have tackled distributed linear regression

problems, based on distributed parameter estimation techniques. In particular, we have focused on the case where the data or regression matrix is noisy, in which case a least-squares estimator is biased. To reduce this bias, we have proposed two novel methods. In **Chapter 7**, we have derived a distributed version of the well-known total least squares (TLS) estimation technique for ad hoc sensor networks, that computes the centralized TLS solution without gathering the data in a fusion center. This yields an unbiased estimate if the regressor noise is white. To facilitate the use of the dual based subgradient algorithm (DBSA), we have transformed the TLS problem to an equivalent convex semidefinite program (SDP) that satisfies the convergence conditions of DBSA, and yields the same solution as in the original problem. Even though we have made a detour through SDR and SDP theory, the resulting D-TLS algorithm relies on solving local TLS-like problems at each node, rather than on computationally expensive SDP optimization techniques. The algorithm is flexible and fully distributed, i.e. it does not make any assumptions on the network topology and nodes only share data with their direct neighbors through local broadcasts. Due to the flexibility and the uniformity of the network, there is no single point of failure, which makes the algorithm robust to sensor failures. We have provided Monte-Carlo simulations that demonstrate the effectiveness of the algorithm. In **Chapter 8**, we have proposed another method, that can also cope with colored regressor noise, which is based on a bias-compensated recursive least squares algorithm with diffusion adaptation in an ad hoc WSN. This algorithm has been analyzed in an adaptive filtering context, and we have demonstrated that the cooperation between nodes indeed reduces the bias, and furthermore reduces the variance of the local parameter estimates at each node.

In the last part of this thesis, we have proposed two supporting techniques that can be used in WSNs for (acoustic) signal estimation. In **Chapter 9**, we have proposed a multiplicative non-negative independent component analysis (M-NICA) algorithm, that performs a blind separation of non-negative well-grounded source signals. The M-NICA algorithm is based on multiplicative update rules which preserve non-negativity, together with a subspace projection based correction step. It has the facilitating property that it does not depend on a user-defined learning rate, as opposed to gradient based techniques such as the NPCA algorithm, where a proper choice for the learning rate is crucial to provide satisfying results. This algorithm has then been used in **Chapter 10** to derive an energy-based multi-speaker voice activity detection algorithm, that aims to track the individual speech power of multiple speakers talking simultaneously. Finally, in **Chapter 11** we have considered two aspects in linear MMSE signal estimation in wireless sensor networks, i.e. sensor subset selection and link failure response. We have first derived an efficient formula to compute the optimal fall-back estimator when the wireless link of one of the sensors fails. High efficiency is achieved by exploiting the knowledge of the inverse correlation matrix as used before the link failure. We have then derived an efficient formula to monitor the utility of each sensor signal in the estima-

tion process. Simulation results of a greedy procedure based on this utility measure in an adaptive LMMSE estimation algorithm have demonstrated that a significant amount of energy can be saved, at the cost of a slight decrease in estimation performance.

12.2 Suggestions for Future Research

In this section, we briefly address some possible future research directions that build further on the techniques proposed in this thesis.

12.2.1 DANSE with Distributed Acoustic Parameter Estimation

When using DANSE in a speech enhancement context, the aim is to estimate a target speech signal, while reducing background noise. However, in some applications, denoising is not a goal as such, but is required as a preprocessing step. For instance, in automatic speech recognition, speech features have to be estimated from the denoised speech signal. Since these features are usually heavily influenced by sound reflections (reverberation), a dereverberation algorithm is also required. In adverse scenarios, such as large reverberant rooms with background noise and long microphone-speaker distances, the estimation of certain speech features (e.g., the auto-regressive (AR) coefficients) is indeed very difficult or impossible to achieve with traditional single-microphone setups, i.e., the AR estimates are often heavily deteriorated by residual noise or reverberation.

However, with a WASN and the DANSE algorithm already in place, also the dereverberation and the AR estimation stage can benefit from cooperation between nodes. The use of WASNs for acoustic parameter estimation (such as AR coefficients, or acoustic room impulse responses (RIR) for dereverberation) is a novel idea that opens many possibilities. By cooperation between multiple microphone nodes in a WASN, it is possible to obtain robust and stable AR estimates in noisy scenarios, independent of the position of the speaker (this avoids that the speaker needs to wear or carry a microphone). The BC-RLS algorithm described in Chapter 8 is able to reduce the bias on the AR estimates that results from the noise due to the large microphone-speaker distance. For the distributed estimation of RIRs for dereverberation, one can rely on the D-TLS algorithm derived in Chapter 7, combined with the techniques presented in [1]. In this case, each node broadcasts its raw microphone signals to its neighbors, such that in each node, a local blind RIR identification problem can be solved. However, based on similar techniques as in Chapter 7, the local RIR estimates can be coupled with those of their neighbors by means of consensus-constraints and network-wide norm constraints.

12.2.2 Nodes with Different Interests: a Game-Theoretic Framework

The DANSE algorithm has the remarkable property that, even though the signals communicated between the different nodes are significantly compressed, the optimal estimators are obtained as if all raw sensor signals are made available to all nodes. This is a surprising theoretical result, especially since each node actually acts 'selfishly', i.e., it does not take the estimation problem of other nodes into account. The reason why an optimal solution is yet obtained is due to the fact that there is a common latent desired source signal that implicitly couples the node-specific estimation problems with each other. In other words: the nodes have the same implicit interest, i.e., each node estimates a different version of the same source signal.

An interesting question is how DANSE can be modified for scenarios where there is no common desired source, i.e., in the case where nodes have different (conflicting) interests. One can indeed imagine scenarios where multiple devices aim to observe or record different sound sources, and so where a desired source for device A can be an interfering source for device B and vice versa. Examples include simultaneous hands-free telephony with interfering speakers, speech communication systems in noisy environments such as airports or factories, audio surveillance of large rooms with multiple relevant sound sources, or ambient intelligent environments with multiple users. Since the current proof of convergence and optimality of DANSE fully depends on a data model with common desired sources for the different nodes, a radically new approach is needed.

The word 'selfish', has been used to specify the behavior of the nodes in DANSE, and in fact hints towards so-called 'non-cooperative game theory', which is often used in economics. We have recently demonstrated that DANSE can indeed be analyzed in this game-theoretic framework, and that it can be viewed as a so-called 'strategic form game' [2]. The case where there are common desired sources can be treated as a special case, for which common game theory theorems can be used to provide an alternative proof of convergence and optimality. Furthermore, we have demonstrated that, in the case where the nodes have different interests, there always exists a parameter setting such that DANSE is in equilibrium, i.e., where the fusion rules in the different nodes have converged. This equilibrium point always corresponds to a so-called 'Nash equilibrium', which is one of the most widely used concepts in game theory [3].

However, the bad news is that Nash-equilibria are generally sub-optimal, which is due to the uncooperative behavior of the players (or the nodes, in our case). Almost always, there exist (unknown) distributed solutions that yield better signal estimates at each of the nodes, but that are not equilibrium solutions of the DANSE algorithm. To avoid convergence to a Nash-equilibrium, nodes need to take each others interests into account, instead of only focusing on their

own node-specific estimation problem. The DANSE algorithm therefore needs to be redesigned to stimulate cooperative behavior.

One specific branch of game theory, namely 'coalitional game theory', may provide a solution here [4]. It offers an analytic framework as well as mathematical tools to design and analyze algorithms that focus on collaboration between nodes. In this case, nodes can form a coalition with each other, such that each node in the coalition has an advantage. This yields equilibria that are better than the Nash-equilibria in non-cooperative games. Coalitional game theory recently knew some successes in the field of communication networks. The challenge is to reformulate and generalize coalitional game-theoretic concepts such that they can be used in the context of estimation problems in WASNs. In this way, new and better algorithms can be designed and analyzed, and more general scenarios can be addressed.

12.2.3 Joint Design of Application Layer Signal Estimation Algorithms and Network Layer Resource Allocation

Signal estimation in ad hoc deployed WASNs has challenging requirements, i.e., high data rates, hard real-time deadlines, distributed computations, multi-hop communications, nodes with different interests, etc. It is therefore important to design both the application layer and the network layer jointly. The challenging open question is then: for given sensor signal observations, given network resources (bandwidth, power, and available links), and taking into account the different interests of the nodes, what is the optimal way to propagate the data over the network? This is a network routing problem and a resource allocation problem, where the 'utility' for every user is hidden in the application layer. This is very different from resource allocation problems in traditional multi-user networks, where every user merely aims to achieve the highest possible bit-rate, and where there is no relation between the application layers of the different users [5, 6].

For example, the number of bits that is allocated to each communicated audio signal is an important design parameter that can be steered by the application layer, depending on the relevance of the communicated signal. Furthermore, it is important to determine which signals are useful for which subset of nodes. For certain nodes, some microphone signals may be more relevant than others (e.g. when they are close to a desired source), some microphone signals may only be useful if they can be combined with specific signals in other nodes, and other signals may not be useful at all. Furthermore, there will be cases where relevant signals can only be combined properly if the communication or network delay is sufficiently small, and so these should then be given a higher priority.

By far the most challenging complication to this network/communication layer resource allocation problem, is that the utility function is actually defined in the application layer, i.e. whether the transmitted (possibly compressed) audio signals will actually be useful to the receiving nodes from a signal enhancement perspective.

Bibliography

[1] Y. A. Huang and J. Benesty, "Adaptive multi-channel least mean square and newton algorithms for blind channel identification," *Signal Processing*, vol. 82, pp. 1127–1138, August 2002.

[2] A. Bertrand and M. Moonen, "Distributed signal estimation in sensor networks where nodes have different interests," *Internal Report ESAT-SISTA*, 2011.

[3] J. Nash, "Non-cooperative games," *The Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[4] W. Saad, H. Zhu, M. Debbah, A. Hjorungnes, and T. Basar, "Coalitional game theory for communication networks," *IEEE Signal Processing Magazine*, vol. 26, pp. 77–97, Sep. 2009.

[5] R. Cendrillon, W. Yu, M. Moonen, J. Verlinden, and T. Bostoen, "Optimal multiuser spectrum balancing for digital subscriber lines," *IEEE Transactions on Communications*, vol. 54, no. 5, pp. 922 – 933, May 2006.

[6] R. Cendrillon, J. Huang, M. Chiang, and M. Moonen, "Autonomous spectrum balancing for digital subscriber lines," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4241 –4257, 2007.

Publication List[1]

International Journal Papers

1. A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435
2. A. Bertrand and M. Moonen, "Blind separation of non-negative source signals using multiplicative updates and subspace projection," *Signal Processing*, vol. 90, no. 10, pp. 2877-2890, 2010
3. 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.
4. A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks – Part II: simultaneous & asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5292 - 5306, 2010.
5. A. Bertrand and M. Moonen, "Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2196 - 2210, 2011.
6. A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2320 - 2330, 2011.
7. A. Bertrand and M. Moonen, "Distributed node-specific LCMV beamforming in wireless sensor networks," Submitted for publication, 2011.
8. A. Bertrand, M. Moonen and A. H. Sayed "Diffusion bias-compensated RLS estimation over adaptive networks," Submitted for publication, 2011.

[1] An up-to-date publication list can be accessed online via `http://homes.esat.kuleuven.be/~abertran`, where the full-text of most publications, as well as additional material (such as presentation slides, sound fragments, and MATLAB software code) is available for download.

International Conference Papers

1. A. Bertrand, K. Demuynck, V. Stouten and H. Van hamme, "Unsupervised learning of auditory filter banks using non-negative matrix factorisation", *Proc. of the IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*, Las Vegas, Nevada USA, Apr. 2008, pp. 4713 - 4716.
2. A. Bertrand and M. Moonen, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network", *Proc. of the IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*, Taipei, Taiwan, Apr. 2009, pp. 2053 - 2056.
3. A. Bertrand and M. Moonen, "Distributed adaptive node-specific MMSE signal estimation in sensor networks with a tree topology", *Proc. of the European signal processing conference (EUSIPCO)*, Glasgow, Scotland, Aug. 2009, pp. 794 - 798.
4. A. Bertrand and M. Moonen, "Energy-based multi-speaker voice activity detection with an ad hoc microphone array", *Proc. of the IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*, Dallas, Texas USA, March 2010, pp. 85 - 88.
5. A. Bertrand and M. Moonen, "Efficient sensor subset selection and link failure response for linear MMSE signal estimation in wireless sensor networks", *Proc. of the European signal processing conference (EUSIPCO)*, Aalborg, Denmark, Aug. 2010, pp. 1092 - 1096.
6. A. Bertrand, J. Callebaut and M. Moonen, "Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks", *Proc. of the International Workshop on Acoustic Echo and Noise Control (IWAENC)*, Tel Aviv, Israel, Aug. 2010.
7. A. Bertrand and M. Moonen, "Distributed LCMV beamforming in wireless sensor networks with node-specific desired signals", *Proc. of the IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*, Prague, Czech Republic, May 2011.
8. A. Bertrand, M. Moonen and A.H. Sayed, "Diffusion-based bias-compensated RLS for distributed estimation over adaptive sensor networks", Submitted for publication, 2011.
9. J. Szurley, A. Bertrand, M. Moonen, P. Ruckebusch, I. Moerman, "Utility based cross-layer collaboration for speech enhancement in wireless acoustic sensor networks", Submitted for publication, 2011.

Technical Reports

1. A. Bertrand and M. Moonen, "Distributed signal estimation in sensor networks where nodes have different interests", *Internal Report K.U. Leuven ESAT-SCD*, 2011.

Curriculum Vitae

Alexander Bertrand was born in Roeselare, Belgium, in 1984. He received the M.Sc. degree (summa cum laude) in Electrical Engineering from Katholieke Universiteit Leuven (K.U. Leuven), Belgium, in 2007. Since 2007, he is pursuing the Ph.D. degree under the supervision of Prof. M. Moonen, at the Department of Electrical Engineering (ESAT), K.U. Leuven. In 2010, he was a visiting researcher at the Adaptive Systems Laboratory, University of California, Los Angeles (UCLA), under the supervision of Prof. A. H. Sayed.

He received a Ph.D. scholarship of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) during 2008-2011, and an FWO (Research Foundation - Flanders) travel grant for a Visiting Research Collaboration at UCLA in 2010. His research interests are in multi-channel signal processing, ad hoc sensor arrays, wireless sensor networks, distributed signal enhancement, speech enhancement, and distributed estimation.