



**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ**

**ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ**

# **Εύρωστοι Προσαρμοστικοί Αλγόριθμοι Μηχανικής Εκμάθησης για Κατανεμημένη Επεξεργασία Σήματος**

**Συμεών Ν. Χουβαρδάς**



**Ευρωπαϊκή Ένωση**  
Ευρωπαϊκό Κοινωνικό Ταμείο



**ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ  
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ**  
*επένδυση στην κοινωνία της γνώσης*  
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ  
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



**ΑΘΗΝΑ**

**ΣΕΠΤΕΜΒΡΙΟΣ 2013**





**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCES  
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS**

**PROGRAM OF POSTGRADUATE STUDIES**

**PhD THESIS**

**Robust Adaptive Machine Learning Algorithms for  
Distributed Signal Processing**

**Symeon N. Chouvardas**



**ATHENS**

**SEPTEMBER 2013**



**PhD DISSERTATION**

**Robust adaptive machine learning algorithms  
for distributed signal processing**

**Symeon N. Chouvardas**

**ADVISOR: Sergios Theodoridis, Professor NKUA**

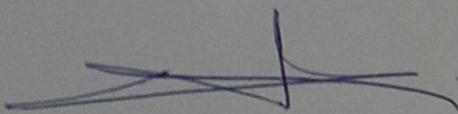
**THREE-MEMBER ADVISING COMMITTEE:**

**Sergios Theodoridis, Professor NKUA**

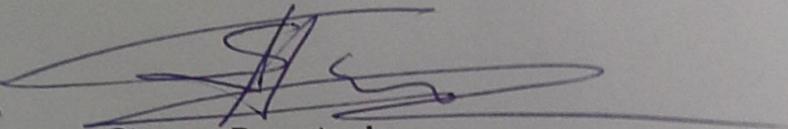
**Stavros Perantonis, Senior Researcher, NCSR "Demokritos"**

**Nicholaos Kalouptsidis, Professor NKUA**

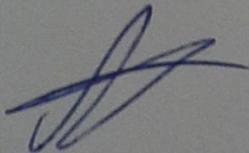
**SEVEN-MEMBER EXAMINATION COMMITTEE**



**Sergios Theodoridis**  
Professor NKUA



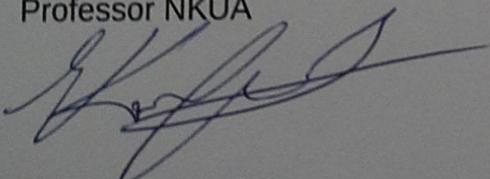
**Stavros Perantonis**  
Senior Researcher, NCSR "Demokritos"



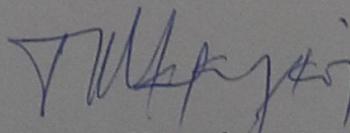
**Nicholaos Kalouptsidis**  
Professor NKUA



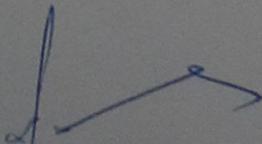
**Kostas Berberidis**  
Professor University of Patras



**Eleftherios Kofidis**  
Assistant Professor University of  
Piraeus



**Petros Maragos**  
Professor NTUA



**Athanasios Rontogiannis**  
Senior Researcher National  
Observatory of Athens

**Examination Date: 19/9/2013**



**ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ**

**Εύρωστοι Προσαρμοστικοί Αλγόριθμοι Μηχανικής Εκμάθησης για  
Κατανεμημένη Επεξεργασία Σήματος**

**Συμεών Ν. Χουβαρδάς**

**ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Σέργιος Θεοδωρίδης Καθηγητής ΕΚΠΑ**

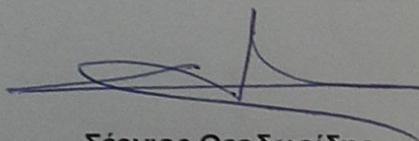
**ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:**

**Σέργιος Θεοδωρίδης Καθηγητής ΕΚΠΑ**

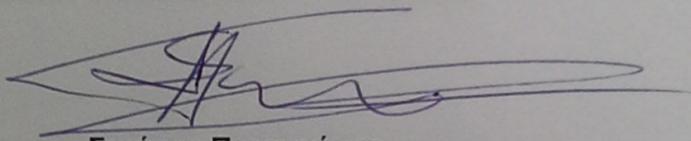
**Σταύρος Περαντώνης Ερευνητής Α, ΕΚΕΦΕ "ΔΗΜΟΚΡΙΤΟΣ"**

**Νικόλαος Καλουπτσίδης Καθηγητής, ΕΚΠΑ**

**ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ**



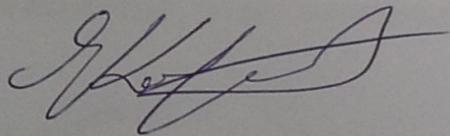
**Σέργιος Θεοδωρίδης**  
Καθηγητής ΕΚΠΑ



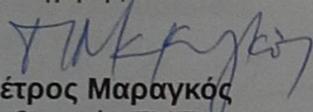
**Σταύρος Περαντώνης**  
Ερευνητής Α, ΕΚΕΦΕ "ΔΗΜΟΚΡΙΤΟΣ"



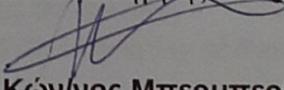
**Νικόλαος Καλουπτσίδης**  
Καθηγητής ΕΚΠΑ



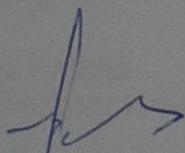
**Ελευθέριος Κοφίδης**  
Επκ. Καθηγητής ΠΑΠΕΙ



**Πέτρος Μαραγκός**  
Καθηγητής ΕΜΠ



**Κών/νος Μπερμπερίδης**  
Καθηγητής Παν/μίου Πατρών



**Αθανάσιος Ροντογιάννης**  
Ερευνητής Αστεροσκοπείου

**Ημερομηνία Εξέτασης: 19/9/2013**



## ΠΕΡΙΛΗΨΗ

Κατανεμημένα δίκτυα τα οποία απαρτίζονται από έναν μεγάλο αριθμό κόμβων, π.χ. Δίκτυα Ασύρματων Αισθητήρων, προσωπικοί υπολογιστές, φορητοί υπολογιστές, έξυπνα τηλέφωνα, κλπ., οι οποίοι συνεργάζονται με σκοπό την επίτευξη ενός κοινού στόχου, αποτελούν μια υποσχόμενη τεχνολογία η οποία βρίσκει εφαρμογή σε πολλά μοντέρνα προβλήματα. Τυπικά παραδείγματα τέτοιων εφαρμογών είναι τα εξής: κατανεμημένη επίβλεψη περιβάλλοντος, εύρεση ακουστικής πηγής, εκτίμηση φάσματος, κλπ. Συνεργατικοί μηχανισμοί δύνανται να βελτιώσουν σημαντικά την διαδικασία εκμάθησης, μέσω της οποίας οι κόμβοι επιτυγχάνουν τον κοινό στόχο τους.

Η παρούσα διατριβή μελετά το πρόβλημα της προσαρμοστικής μάθησης σε κατανεμημένα δίκτυα, εστιάζοντας στο πρόβλημα της κατανεμημένης εκτίμησης παραμέτρων. Ένα σύνολο από κόμβους λαμβάνουν πληροφορία, η οποία σχετίζεται με συγκεκριμένες παραμέτρους, και η εκτίμηση των εν λόγω παραμέτρων αποτελεί τον στόχο μας. Προς αυτήν την κατεύθυνση, οι κόμβοι λαμβάνουν υπόψη τόσο τις τοπικές μετρήσεις, όσο και την πληροφορία η οποία λαμβάνεται από την συνεργασία με τους υπόλοιπους κόμβους του δικτύου. Στα πλαίσια της παρούσας διατριβής, η συνεργασία μεταξύ των κόμβων ακολουθεί την φιλοσοφία της κατανεμημένης βελτιστοποίησης μέσω διάχυσης και οι προτεινόμενοι αλγόριθμοι ανήκουν στην οικογένεια APSM. Αρχικά, εύρωστοι αλγόριθμοι με βάση τον APSM προτείνονται. Ο στόχος είναι η «εναρμόνιση» της πληροφορίας, η οποία λαμβάνεται από την γειτονιά, με την τοπική πληροφορία. Η εν λόγω «εναρμόνιση» επιτυγχάνεται μέσω προβολής της πληροφορίας της γειτονιάς πάνω σε ένα κυρτό σύνολο, το οποίο κατασκευάζεται με βάση τοπικές μετρήσεις. Στην συνέχεια, αντιμετωπίζεται σενάριο κατά το οποίο ένα υποσύνολο των κόμβων του δικτύου δυσλειτουργεί και παράγει μετρήσεις, οι οποίες έχουν υποβαθμιστεί σημαντικά από τον θόρυβο. Για την επίλυση του εν λόγω προβλήματος γίνεται χρήση της συνάρτησης κόστους Huber, η οποία είναι εύρωστη στην ύπαρξη ακραίων τιμών θορύβου. Επιπλέον, μελετήθηκε το πρόβλημα της προσαρμοστικής εκτίμησης αραιών διανυσμάτων στα πλαίσια της κατανεμημένης μάθησης. Οι κόμβοι του δικτύου αναζητούν άγνωστο, αραιό διάνυσμα, το οποίο αποτελείται από μικρό αριθμό μη μηδενικών συντελεστών. Περιορισμοί σταθμισμένης  $l_1$  νόρμας καθώς και προβολές μεταβλητής μετρικής, οι οποίες ευνοούν αραιές λύσεις χρησιμοποιούνται. Τέλος, προτείνονται αλγόριθμοι, οι οποίοι οδηγούν σε μείωση της πληροφορίας που αποστέλεται στο δίκτυο, περιορίζοντας τις εκτιμήσεις να βρίσκονται πάνω σε έναν Krylov υπόχωρο.

Οι προτεινόμενοι αλγόριθμοι έχουν υψηλή απόδοση ενώ ταυτόχρονα οι απαιτούμενοι πόροι εύρους ζώνης και η πολυπλοκότητα παραμένουν σε λογικά επίπεδα.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:** Επεξεργασία Σήματος

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:** Μηχανική εκμάθηση, προσαρμοστική μάθηση, κατανεμημένοι αλγόριθμοι, προβολές



## Abstract

Distributed networks comprising a large number of nodes, *e.g.*, Wireless Sensor Networks, Personal Computers (PC's), laptops, smart phones, etc., which cooperate with each other in order to reach a common goal, constitute a promising technology for several applications. Typical examples include: distributed environmental monitoring, acoustic source localization, power spectrum estimation, etc. Sophisticated cooperation mechanisms can significantly benefit the learning process, through which the nodes achieve their common objective.

In this dissertation, the problem of adaptive learning in distributed networks is studied, focusing on the task of distributed estimation. A set of nodes sense information related to certain parameters and the estimation of these parameters constitutes the goal. Towards this direction, nodes exploit locally sensed measurements as well as information springing from interactions with other nodes of the network. Throughout this dissertation, the cooperation among the nodes follows the diffusion optimization rationale and the developed algorithms belong to the APSM algorithmic family.

First, robust APSM-based techniques are proposed. The goal is to “harmonize” the spatial information, received from the neighborhood, with the locally sensed one. This “harmonization” is achieved by projecting the information of the neighborhood onto a convex set, constructed via the locally sensed measurements. Next, the scenario, in which a subset of the node set is malfunctioning and produces measurements heavily corrupted with noise, is considered. This problem is attacked by employing the Huber cost function, which is resilient to the presence of outliers. In the sequel, we study the issue of sparsity-aware adaptive distributed learning. The nodes of the network seek for an unknown sparse vector, which consists of a small number of non-zero coefficients. Weighted  $\ell_1$ -norm constraints are embedded, together with sparsity-promoting variable metric projections. Finally, we propose algorithms, which lead to a reduction of the communication demands, by forcing the estimates to lie within lower dimensional Krylov subspaces. The derived schemes serve a good trade-off between complexity/bandwidth demands and achieved performance.

---

**Subject Area:** Signal Processing.

**Keywords:** Machine Learning, adaptive learning, distributed algorithms, projections.

*To my family.*



# Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisor Prof. Sergios Theodoridis for offering me the opportunity to pursue my PhD degree at the University of Athens. I am really grateful for his continuous guidance and support throughout these years. With his deep knowledge on the fields of signal processing and machine learning, and, of course, his good sense of humor, working with him was a real pleasure. I am really proud of being his student.

In addition, I would like to thank my dissertation co-advisors Prof. Nicholas Kalouptsidis and Dr. Stavros Perantonis. I owe a deep gratitude to Prof. Nicholas Kalouptsidis for his guidance, collaboration and the fruitful discussions on the field of Compressed Sensing. Due thanks also go to the rest of my dissertation committee: Prof. Konstantinos Berberidis, Assistant Prof. Eleftherios Kofidis, Prof. Petros Maragkos and Dr. Athanasios Rontogiannis.

I would also like to thank Assistant Prof. Konstantinos Slavakis for his support and for sharing with me his expertise on the field of convex optimization. Furthermore, I would like to express my gratitude to Dr. Ioannis Kopsinis for the collaboration and for the discussions we had on the field of Adaptive Learning. Special thanks go to Dr. Gerasimos Mileounis for the perfect cooperation we had on the field of sparsity-aware learning. I would also like to thank the current and former members of the lab: Dr. Theodoros Giannakopoulos, Dr. Alexandros Katsiotis, Dr. Konstantinos Xenoulis, Dr. Konstantinos Rizogiannis, Dr. Stylianos Tzikopoulos, Evangelos Logaras and Dimitrios Manatakis. I could not forget my good friends Anastasios Kagkadis and Aristofanis Sakarellos for their endless encouragement and support throughout these years.

I would like to dedicate this dissertation to my parents, Niko and Katerina, my brother Michalis, my sister Antonia for always being there for me, and my niece Maria-Katerina for bringing so much happiness to our family.



# List of Publications

## Journal Publications

- S. Chouvardas, K. Slavakis, and S. Theodoridis. Adaptive robust distributed learning in diffusion sensor networks. *IEEE Transactions on Signal Processing*, 59(10):4692–4707, 2011.
- S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis. A sparsity promoting adaptive algorithm for distributed learning. *IEEE Transactions on Signal Processing*, 60(10):5412–5425, Oct. 2012.
- S. Chouvardas, K. Slavakis, and S. Theodoridis. Trading off complexity with communication costs in distributed adaptive learning via Krylov subspaces for dimensionality reduction. *IEEE Journal of Selected Topics in Signal Processing*, 7(2):257–273, 2013.
- Symeon Chouvardas, Konstantinos Slavakis, Sergios Theodoridis, and Isao Yamada. Stochastic analysis of hyperslab-based adaptive projected subgradient method under bounded noise. *To appear in IEEE Signal Processing Letters*, 2013.

## Conference Publications

- S. Chouvardas, K. Slavakis, and S. Theodoridis. A novel adaptive algorithm for diffusion networks using projections onto hyperslabs. In *CIP*, pages 393–398. IEEE, 2010 (**best student paper award**).
- Symeon Chouvardas, Konstantinos Slavakis, and Sergios Theodoridis. Trading off communications bandwidth with accuracy in adaptive diffusion networks. In *ICASSP*, pages 2048–2051. IEEE, 2011.

- 
- Symeon Chouvardas, Konstantinos Slavakis, Yannis Kopsinis, and Sergios Theodoridis. Sparsity-promoting adaptive algorithm for distributed learning in diffusion networks. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1084–1088. IEEE, 2012.
  - Symeon Chouvardas, Gerasimos Mileounis, Nikolaos Kalouptsidis, and Sergios Theodoridis. A greedy sparsity-promoting LMS for distributed adaptive learning in diffusion networks. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013.
  - Symeon Chouvardas, Gerasimos Mileounis, Nikolaos Kalouptsidis, and Sergios Theodoridis. Training-Based and Blind Algorithms for Sparsity-Aware Distributed Learning. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2013.

# Contents

<b>1</b>	<b>Introduction</b>	<b>31</b>
1.1	Motivation . . . . .	31
1.2	Related Work . . . . .	34
1.3	Objectives . . . . .	36
1.4	Outline . . . . .	38
<b>2</b>	<b>Basic Concepts of Distributed Signal Processing</b>	<b>41</b>
2.1	Brief Historical Remarks and Recent Research Trends in Distributed Signal Processing and Wireless Sensor Networks . . . . .	41
2.1.1	History of Research in WSNs . . . . .	41
2.1.2	Research trends and Applications involving Distributed Learning . . . . .	43
2.2	Basic Principles of Distributed Learning and Diffusion optimization . . . . .	45
2.2.1	Motivation . . . . .	45
2.2.2	Networks and Neighborhoods . . . . .	46
2.2.3	Distributed Learning via Diffusion Optimization . . . . .	48
2.3	Applications of Distributed Learning and Diffusion optimization . . . . .	53
2.3.1	Distributed Learning met in nature . . . . .	53
2.3.2	Distributed Averaging . . . . .	56
2.3.3	Spectral Sensing for Cognitive Radios . . . . .	61
<b>3</b>	<b>Adaptive Filtering</b>	<b>65</b>
3.1	The LMS algorithm . . . . .	65
3.1.1	Convergence of the Diffusion based LMS . . . . .	68
3.2	Recursive Least Squares Algorithm . . . . .	71

# CONTENTS

---

3.2.1	Diffusion Recursive Least Squares . . . . .	74
3.3	The Adaptive Projected Subgradient Method . . . . .	76
3.3.1	Basic Concepts of Convex Analysis and the POCS Algorithm. . . . .	78
88		
3.4	Numerical Examples . . . . .	94
<b>4</b>	<b>Adaptive Robust Algorithms for Distributed Learning</b>	<b>97</b>
4.1	Diffusion Adaptive Algorithm Using Projections onto Hyperplanes . . . . .	98
4.2	Theoretical Analysis . . . . .	101
4.3	Introducing Robustness to Cope with a Failure of Nodes . . . . .	103
4.4	Numerical Examples . . . . .	106
	<b>Appendices</b>	<b>111</b>
	<b>Appendix A Consensus Matrix and the Consensus Subspace</b>	<b>111</b>
	<b>Appendices</b>	<b>113</b>
	<b>Appendix B Proof of Theorem 1</b>	<b>113</b>
B.1	Proof of Theorem 1.1 . . . . .	113
B.2	Proof of Theorem 1.2 . . . . .	117
B.3	Proof of Theorem 1.3 . . . . .	119
B.4	Proof of Theorem 1.4 . . . . .	120
<b>5</b>	<b>A Sparsity–Promoting Projection–Based Algorithm for Distributed Learning</b>	<b>123</b>
5.1	Sparsity–Aware Learning . . . . .	123
5.1.1	Sparsity–Promoting Adaptive Algorithms . . . . .	125
5.2	Set–theoretic Estimation Approach and Variable Metric Projections . . . . .	125
5.3	Proposed Algorithmic Scheme . . . . .	129
5.4	Numerical Examples . . . . .	132
	<b>Appendices</b>	<b>140</b>

<b>Appendix C Basic Concepts Of Convex Analysis Employing Weighted Inner Products</b>	<b>140</b>
<b>Appendices</b>	<b>141</b>
<b>Appendix D Variable Metric Projection onto the Weighted <math>\ell_1</math> Ball</b>	<b>141</b>
<b>Appendices</b>	<b>142</b>
<b>Appendix E Proof of Theorem 1</b>	<b>142</b>
E.1 Monotonicity . . . . .	142
E.2 Asymptotic optimality . . . . .	145
E.3 Asymptotic Consensus . . . . .	149
E.4 Strong Convergence . . . . .	150
<b>6 Dimensionality Reduction in Distributed Adaptive Learning via Krylov Subspaces</b>	<b>151</b>
6.1 Problem Formulation . . . . .	152
6.1.1 Krylov Subspaces and the Reduced Rank Wiener Solution . . . . .	153
6.1.2 Set-theoretic estimation: the reduced rank case . . . . .	155
6.2 Proposed Scheme . . . . .	157
6.2.1 Enhancing the Information Flow . . . . .	158
6.2.2 The Algorithmic Scheme . . . . .	160
6.3 Whitening the input . . . . .	164
6.4 Numerical Examples . . . . .	170
<b>Appendices</b>	<b>177</b>
<b>Appendix F Projection Operators Onto Subspaces</b>	<b>177</b>
<b>Appendices</b>	<b>178</b>
<b>Appendix G Proof of Claim 1</b>	<b>178</b>
<b>Appendices</b>	<b>179</b>

# CONTENTS

---

<b>Appendix H Proof of Claim 2</b>	<b>179</b>
<b>Appendices</b>	<b>180</b>
<b>Appendix I Proof of Theorems 1 and 2</b>	<b>180</b>
I.1 Monotonicity . . . . .	180
I.2 Asymptotic Optimality . . . . .	182
I.3 Asymptotic Consensus . . . . .	183
I.4 Strong Convergence . . . . .	185
<b>7 Conclusions and Future Work</b>	<b>187</b>
7.1 Summary . . . . .	187
7.2 Future Work . . . . .	188
<b>List of Abbreviations</b>	<b>191</b>
<b>List of Symbols</b>	<b>193</b>
<b>Bibliography</b>	<b>195</b>

# List of Tables

4.1	Convergence Properties of Adaptive Distributed Algorithms . . . . .	103
6.1	Steady State distances. . . . .	172
6.2	Squared Distance from the Consensus Subspace. . . . .	172



# List of Figures

1.1	(a) Illustration of a centralized network with an FC. Circles denote the nodes and the square denotes the FC. (b) A decentralized network. . . . .	34
2.1	Illustration of an ad-hoc network. . . . .	47
2.2	Illustration of a node, the target source and the direction vectors. . . . .	54
3.1	Illustration of a hyperslab and the projection of a point onto it. . . . .	77
3.2	Illustration of a gradient and two subgradients of a convex function. Note that in $\mathbf{w}_1$ the function is differentiable, so there exists a unique supporting hyperplane, which is tangent to the graph of the function, whereas in $\mathbf{w}_2$ the function is not differentiable and there exist more than one hyperplanes, that support the graph of the function. . . . .	79
3.3	The projection and the relaxed projection operators. . . . .	80
3.4	(a) The geometry of a halfspace. Its boundary is a hyperplane. (b) A closed ball $B_{[\mathbf{c},\delta]}$ . . . . .	81
3.5	Sequential formulation of the POCS algorithm. Each iteration takes us closer to the intersection of the convex sets. . . . .	82
3.6	Parallel formulation of the POCS algorithm. Each iteration takes us closer to the intersection of the convex sets. . . . .	83
3.7	Geometrical illustration of the algorithm. The point $\mathbf{w}_n$ is projected onto $\mathcal{C}_n, \mathcal{C}_{n-1}$ , a convex combination of these projection is computed. The occurring vector from this step is projected onto the constraint set. . . . .	85
3.8	(a) A cost function whose 0-th level set is a hyperslab. (b) A cost function and the supporting hyperplane generated by the differential. . . . .	86

## LIST OF FIGURES

---

3.9	MSD performance for the first experiment. . . . .	95
3.10	MSD performance for the second experiment. . . . .	95
3.11	MSD curves for the 2nd experiment. Only a single iteration is plotted, since Theorem 4 does not involve the expectation operator, where the averaging of independent iterations is necessary. . . . .	96
4.1	Illustration of an iteration for the case of $q = 2$ . The aggregate is projected onto an external hyperslab and the result, $\mathbf{z}_{k,n}$ , is used in the adaptation step. Note that $\mathbf{z}_{k,n}$ is projected onto $S_{k,n}$ and $S_{k,n-1}$ , and the projections are combined together. The update estimate, $\mathbf{w}_{k,n+1}$ lies closer to the intersection of the hyperslabs, compared to $\phi_{k,n}$ . Note, also, that the hyperslab $S'_{k,n}$ contains $S_{k,n}$ . . . . .	99
4.2	Illustration of the cost functions $\tilde{\Theta}(\cdot)$ , $\hat{\Theta}(\cdot)$ . The aggregate, $\phi_{k,n}$ , is projected onto $H'_{k,n}$ , which is the intersection of the hyperplane associated with the subgradient at $\hat{\Theta}(\phi_{k,n})$ with the space $\mathbb{R}^m$ , to provide $\mathbf{z}_{k,n}$ . In the sequel, $\mathbf{z}_{k,n}$ is used into the adaptation step. . . . .	105
4.3	(a) MSE for experiment 1. (b) MSD for experiment 1. (c) The statistics of the network's regressors. . . . .	108
4.4	(a) MSE for experiment 2. (b) MSD for experiment 2. (c) The statistics of the network's regressors. . . . .	109
4.5	(a) MSE for experiment 3. (b) MSD for experiment 3. (c) The statistics of the network's regressors. . . . .	109
4.6	(a) MSE for experiment 4 by considering all the nodes of the network. (b) MSD for experiment 4. (c) Average MSE computed over the healthy nodes. . . . .	110
5.1	Illustration of a hyperslab, the standard metric projection of a vector $\mathbf{w}$ onto it, denoted by $P_{S_n}(\mathbf{w})$ , and the variable metric projection onto it. . . . .	127
5.2	Illustration of a weighted $\ell_1$ ball (solid line magenta) and an unweighted $\ell_1$ ball (dashed line blue). . . . .	128

5.3	Geometrical interpretation of the algorithm. The number of hyperslabs onto which $\phi_{k,n}$ is projected, using variable metric projections, is $q = 2$ . The result of these two projections, which are illustrated by the dash dotted black line, is combined (red line) and the result is projected (solid black line) onto the sparsity promoting weighted $\ell_1$ ball, in order to produce the next estimate. . . . .	133
5.4	MSD for the experiment 1. . . . .	134
5.5	MSD for the experiment 2. . . . .	135
5.6	MSD for the experiment 3. . . . .	137
5.7	MSD for the experiment 4. . . . .	137
5.8	Squared distance from the consensus subspace, for experiment 5. . . . .	139
6.1	Illustration of a hierarchical network with $L = 5$ . The solid lines denote the simple communication links, whereas the dashed-dotted ones the hierarchical communication links. . . . .	158
6.2	(a) Geometrical illustration of the algorithm for $q = 1$ . The aggregate $\phi_{k,n}$ , which belongs in the subspace, is projected onto the intersection of the subspace and the hyperslab, generated by the measurement data. (b) The algorithmic scheme in the reduced dimension space, i.e., $\mathbb{R}^D$ . . . . .	162
6.3	Illustration of $K_D(\mathcal{R}', \mathbf{r}')$ , $\hat{M}$ and the connection between points that belong to them. . . . .	167
6.4	Average MSE for the first experiment. . . . .	171
6.5	Average EMSE for the second experiment. . . . .	173
6.6	Average MSE for the third experiment. . . . .	174
6.7	Average MSE for the fourth experiment. . . . .	175
6.8	Average MSE for the fifth experiment. . . . .	175
6.9	Average MSE for the sixth experiment. . . . .	176



## **Preface**

This research has been co-financed by the European Union (European Social Fund-ESF) and Greek national funds through the Operational Program “Education and Lifelong Learning” of the National Strategic Reference Framework (NSRF)- Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.



# Chapter 1

## Introduction

*This dissertation deals with the problem of adaptive learning in distributed networks. In the introduction chapter, we present the motivation of this study, the related work, the objectives, as well as an outline for this work.*

### 1.1 Motivation

Distributed networks comprising a number of connected nodes, *e.g.*, Personal Computers (PC's), laptops, smart phones, surveillance cameras and microphones, wireless sensor networks etc., which exchange information in order to reach a common goal, are envisioned to play a central role in many applications. Typical examples of emergent applications involving distributed networks are: distributed environmental monitoring, acoustic source localization, power spectrum estimation, target tracking, surveillance, traffic control, patient monitoring and hospital surveillance, just to name a few [2, 12, 36, 55, 111]. All the previously mentioned applications share in common the fact that the nodes are deployed over a geographic region providing spatial diversity to the obtained measurements. Henceforth, the development of algorithms and node cooperation mechanisms, which exploit the information diversity over time and space, so that a common objective to be reached, becomes essential.

In this dissertation, the problem of distributed processing is studied with a focus on the distributed estimation task. A number of nodes, which are spread over a geographic region, sense information related to certain parameters; the estimation of these parameters comprises our goal. The main idea behind distributed processing is that the nodes exchange

information among them and make decisions/computations in a *collaborative* way instead of working individually, using solely the information that is locally sensed. It is by now well established, that the cooperation among the nodes leads to better results compared to the case where they act as individual learners, see for example [28, 88, 138]. The need to develop node cooperation mechanisms is increased due to the presence of noise in the majority of applications. More specifically, the measurements observed at each node are corrupted by noise, and this fact adds further uncertainty on the obtained estimates of the unknown target parameters. This uncertainty can be reduced via the cooperation of the nodes.

The network topology is dictated by several parameters, such as, geographical constraints, privacy constraints, etc., and determines the type of cooperation among the nodes. In the sequel, we present two popular network topologies. Depending on the existence or absence of a central node, also known as Fusion Center (FC), the networks are classified in two main subcategories, the centralized and the decentralized, respectively.

- In the *centralized* networks, all the nodes, but the central one, collect information related to the parameters to be estimated and then they transmit it to the FC. The latter processes the data, and the resulting estimate is communicated back to all the nodes. This topology is illustrated in Fig. 1.1. The estimates computed in centralized networks are optimal in the sense that all the available information is used for their computation. Distributed networks, which include an FC, encounter the following limitations. First of all, the existence of a central node is not always possible due to geographical constraints and due to the large amount of energy and communications bandwidth, which is needed by this topology, see *e.g.*, [55]. Furthermore, networks obeying the centralized topology lack robustness, since if the FC fails, which could happen for instance if in a Wireless Sensor Network (WSN) setup the battery runs out, then the network collapses. It should also be pointed out that since the FC makes all the essential computations, its computational power has to be considerably high. However, in applications such as WSNs, one usually employs cheap sensors of limited computational power and henceforth the existence of an FC is not feasible. Last but not least, in many cases, *e.g.*, medical applications, Internet studies, etc., privacy has to be preserved. That is, the nodes are not “willing” to exchange raw data but only their learning results [48].

- In the decentralized topology (Fig. 1.1), the FC does not exist, and each node of the network performs computations by exploiting: a) the local measurements which are sensed from the environment, b) the information which is received from the rest of the nodes, with which communication is possible. Each node is able to transmit and receive data coming from a subset of nodes; these comprise its *neighborhood*. In decentralized networks, the following issues have to be taken into consideration:
  - *Performance*: A performance close to the optimal, that is the one associated with the centralized networks, which use all the available data, has to be achieved. In other words, despite the fact that direct communication among some of the nodes cannot be established, sophisticated cooperation mechanisms have to be developed, in order to “push” the performance to be as close as possible to the ideal scenario.
  - *Robustness to possible failures*: As it has been already mentioned, a major drawback of the centralized topology is that if the FC fails then the network collapses. Decentralized networks have to be constructed so as to be robust against possible node failures.
  - *Bandwidth and complexity constraints*: The amount of transmitted information has to be as small as possible, in order to keep the bandwidth low. Furthermore, since in decentralized networks a central processing unit with powerful computational capabilities is not present and usually cheap processing units comprise the nodes, low-complexity schemes have to be developed.
  - *Adaptivity*: In many applications, such as, source localization, spectrum sensing, etc, the nodes of the network are tasked to estimate non-stationary parameters, *i.e.*, parameters which vary with time. Batch estimation algorithms, which use all the available training data simultaneously, cannot attack such problems. To this end, adaptive techniques have to be developed, where the data are observed sequentially, one per (discrete) time instance, and operate in an online fashion for updating and improving the estimates.

In decentralized networks, in which the nodes are tasked to estimate adaptively an unknown parameter, there are two cooperation strategies which can be adopted. The *diffusion*

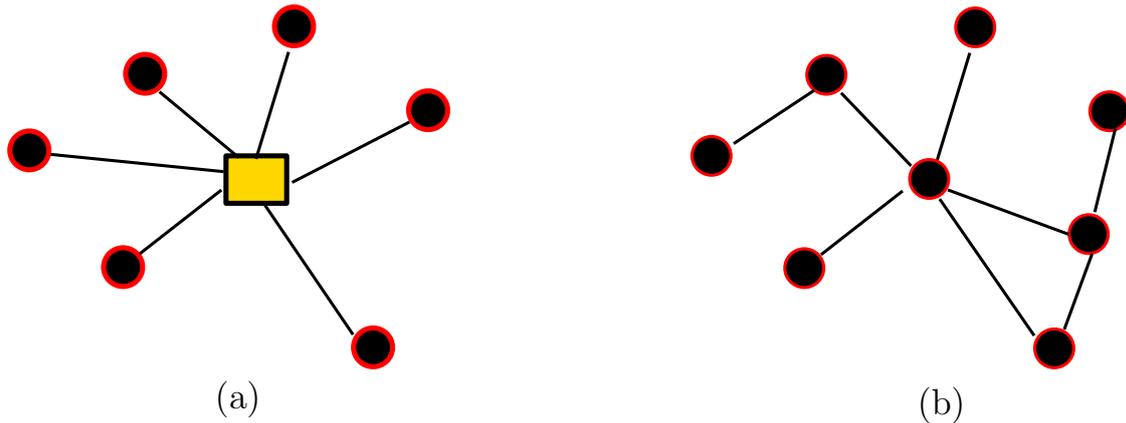


Figure 1.1: (a) Illustration of a centralized network with an FC. Circles denote the nodes and the square denotes the FC. (b) A decentralized network.

and the *consensus*-based ones. In the former, the available information at each node, which comprises the node's local information, as well as the information received by the neighborhood, is diffused by adopting a proper cooperation protocol. In the latter, constraints are embedded, which force the estimates, which are obtained at the nodes, to be the same. These concepts will be explained in more detail in Chapter 2.

In this work, adaptive distributed algorithms for robust parameter estimation are developed. The diffusion rationale is adopted and the algorithms serve a good trade-off between the complexity/bandwidth demands and performance.

## 1.2 Related Work

Decentralized processing has attracted a big interest during the 80's and the 90's, *e.g.*, [10, 135]. Nevertheless, the task of distributed adaptive filtering was first studied in 2006 by Lopes and Sayed, [92, 93]. In these works, an *incremental* network is considered, *i.e.*, a network where each node is able to communicate with one neighbor and the nodes lie in a cyclic pattern (see also Chapter 2). The algorithm, which is employed for the unknown parameter estimation, is the Least Mean Squares (LMS). Algorithms for adaptive learning in incremental networks have been proposed in the context of the Recursive Least Squares (RLS) algorithm, *e.g.*, [121], and the Affine Projection Algorithm (APA), *e.g.*, [89, 90].

Apart from the incremental topology, the task of adaptive distributed learning has been studied in the context of the so-called *ad-hoc* networks. In these networks, each node is

able to communicate with a subset of the node set. As it will become clear later on, the ad-hoc topology has an easier practical implementation compared to the incremental one. For this reason, such networks have been extensively studied in the literature over the past years. The first study was carried out by Lopes and Sayed, [94]. The authors there proposed and studied a diffusion based LMS. The theoretical properties of the diffusion LMS, in a unified way, were presented in [28]. Diffusion RLS based schemes have been proposed in the work [27], and diffusion based Kalman filtering has been studied in [29]. A diffusion based algorithm, which follows the set-theoretic viewpoint has been proposed in the context of the diffusion-based Adaptive Projected Subgradient Method (APSM) in [31].

Another family of algorithms, proposed in the literature, is the *consensus*-based ones. The main objective in these methodologies is to force the nodes to consensus; or in other words to converge to the same estimate. The classical non-adaptive consensus based estimation algorithms, *e.g.*, [115], required two time scales: a time scale in which the nodes average their estimates, and a second one for updating them using the locally sensed information. This is impractical in real-time adaptive learning. However, fully adaptive consensus-based LMS schemes have been proposed in the studies: [98, 123] and consensus based RLS algorithms in [97, 98]. The paper in [136] carries out a comparative study both in experimental as well as theoretical terms, in which the diffusion-based LMS schemes were proved to outperform, in terms of convergence speed, their consensus counterparts.

Recently, the topic of sparsity-aware adaptive distributed learning has drawn a considerable attention. The goal is the estimation of a possibly time-varying unknown parameter vector, which is assumed to be sparse, *i.e.*, it has a small number of non-zero coefficients. An LMS based algorithm for sparsity-aware learning can be found in [49, 50].

Adaptive algorithms in ad-hoc networks, which adopt the diffusion rationale, have been proposed for modeling the behavior of bee swarms, fish schools, bird formations, mobile networks, etc, see *e.g.*, [25, 88, 138, 139]. Another issue, which is dealt in the studies [23, 24] is the hierarchy of the network. More specifically, different transmission capabilities are assigned to a subset of the node set, depending on their position on the network and the goal is to enhance the network flow.

All the previously mentioned adaptive distributed algorithms consider the case of linear parameter estimation. More precisely, it is assumed that the data obtained at each node

are related via an unknown vector, which is the one to be estimated from the nodes of the network. The study in [32] proposes a diffusion-based technique for minimizing general cost functions. Finally, [34] treats the case where nodes are tasked to minimize general cost functions in a scenario where the communication among them is non-ideal.

### 1.3 Objectives

The main objective of this dissertation is to develop algorithms in the context of adaptive estimation in distributed networks. The diffusion rationale is adopted and the proposed algorithms belong to the APSM algorithmic family. Our effort can be divided in the following research areas:

- *Study and development of adaptive techniques, which are tasked to “harmonize”, i.e., to bring “close”, the spatial information, received from the neighborhood, with the locally sensed one.* A challenge, which arises in distributed learning, is that despite the fact that the nodes share a common objective, the statistics of the locally sensed information may differ significantly from node to node. In adaptive distributed learning, this can lead to slow convergence speed of the developed algorithms. This motivated us to propose a technique, [40,41], through which this problem is overcome. More specifically, following the diffusion rationale, the information coming from the network is fused under some certain protocol. In order to harmonize this information with the locally sensed data, the fused information is projected onto a set, which is constructed by exploiting the local measurements. We observed experimentally that this enhances significantly the performance of the algorithm.
- *Development of schemes which are robust to cope with possible node failures:*

The scenario where a subset of the node set is malfunctioning is often met in real world applications involving distributed networks. Therefore, there is a strong need to develop techniques, which bypass the information transmitted from these nodes and rely on the information coming from the “healthy” ones. More analytically, the scenario in which a number of nodes sense measurements, which are heavily corrupted by noise, was considered in [41]. The problem is successfully solved by adopting the

Huber cost function, [70], which is drawn from the “armory” of the robust statistics theory. Employing this cost function, the contribution of the corrupted nodes is reduced significantly and the performance of the scheme is enhanced, compared to the case where classical cost functions are used for the parameter estimation.

- *Sparsity-aware adaptive distributed learning:* The adaptive estimation of sparse vectors in ad-hoc networks is an issue of significant importance, since a plethora of signals met in nature adhere to sparse representations. Typical examples of such signals, which are also met in the distributed estimation problems, are sparse echo signals, sparse wireless channels, just to name a few, *e.g.*, [19]. As it is by now well documented in the literature, *e.g.*, [4, 21, 82], the estimation of sparse vectors is aided by the use of the so-called  $\ell_1$  norm constraints (see Chapter 5). A more sophisticated route is the use of weighted  $\ell_1$  norm constraints [22]. Having the weighted  $\ell_1$  norm constraints as a kick-off point, we developed an APSM based algorithm for distributed learning, which exploits the sparsity of the unknown vector to be estimated, *e.g.*, [39, 45]. For further performance enhancement, the notion of proportionality, *e.g.*, [8, 54], is also adopted. To this end, different weights are assigned to different coefficients of the computed estimates and the goal is to lead the small coefficients to diminish faster. A theoretical analysis is carried out, and numerical examples verify the powerful performance of the proposed scheme.
- *Adaptive schemes, which reduce the communication demands, in scenarios where a large amount of information has to be disseminated over the network:* In adaptive distributed learning, the collaboration among the nodes of the network, through information exchange, turns out to be beneficial to the performance of the algorithms, compared to the case where each one operates individually (see for example [28]). This comes at the price of increased bandwidth demands, since the nodes have to transmit information to their neighbors. As it will become clear throughout this dissertation, the number of transmitted coefficients is mainly dictated by the dimensionality of the vector to be estimated. Hence, in scenarios where this dimensionality is large, the bandwidth demands become a burden. Motivated by this fact, in order to reduce the number of transmitted coefficients, we propose in [42, 43], a technique for dimensionality reduc-

tion, which is based on the following philosophy. Instead of seeking for the unknown vector, we seek for the projection of it onto a *lower dimensional subspace*. Through this procedure (see Chapter 6), the nodes transmit a number of coefficients, which is equal to the dimension of the subspace. The involved subspaces are the so-called *Krylov* subspaces, named after the Russian applied mathematician and naval engineer Alexei Krylov, who published a study on this issue in 1931, [83]. The Krylov subspaces were originally proposed for solving large systems of linear equations avoiding matrix inversions, *e.g.*, [117]. Nevertheless, they have been also used in the adaptive filtering task, *e.g.*, [154, 155]. Employing Krylov subspaces for dimensionality reduction in adaptive learning, provides a good trade-off between the accuracy of the algorithms and the reduction, in the sense that even if the subspace is of a significantly reduced dimension, compared to the original space in which the unknown vector lies, the performance of the algorithm is not seriously degraded; this is due to the optimality associated with the Krylov dimensionality reduction technique.

## 1.4 Outline

The present thesis is organized as follows:

- *Chapter 2* presents the basic concepts of distributed signal processing.
- *Chapter 3* reviews classical adaptive algorithms, *i.e.*, the LMS and the RLS, as well as their distributed counterparts. Furthermore, the basic notions and ideas of the main algorithmic tool, which will be used in this study, *i.e.*, the APSM, are discussed.
- *Chapter 4* proposes an APSM based distributed algorithm, which follows the diffusion rationale. The main goal of this scheme is the harmonization between the locally sensed data, and the information observed from the neighborhood. Moreover, a distributed APSM, which is robust to deal with malfunctioning nodes is studied. The algorithm employs the Huber cost function.
- *Chapter 5* develops a sparsity-promoting APSM based algorithm, for adaptive learning in distributed networks. Sparsity is enforced via weighted  $\ell_1$  norm constraints and variable metric projections.

- *Chapter 6* proposes a technique, which reduces the amount of transmitted information, throughout the network. The presented scheme builds upon concepts of the APSM algorithmic family and the dimensionality reduction takes place via the Krylov subspace rationale.
- *Chapter 7* summarizes the main conclusions of this dissertation and presents possible directions for future research.



# Chapter 2

## Basic Concepts of Distributed Signal Processing

*In this chapter, the basic concepts and principles of distributed learning will be discussed. We first present a short overview of the research in distributed signal processing, which was mainly developed in the context of the wireless sensor network applications. Furthermore, we discuss the problem of distributed optimization and we provide the necessary background on networks. In the sequel, we describe how the diffusion rationale is employed in the decentralized optimization task. Finally, we present some distributed learning problems and we apply the previously described tools for their solution.*

### 2.1 Brief Historical Remarks and Recent Research Trends in Distributed Signal Processing and Wireless Sensor Networks

#### 2.1.1 History of Research in WSNs

The need to develop decentralized algorithms began to grow in the 60's. This was a direct consequence of several facts. The most important one, however, stemmed from the emergent need for exploitation of the WSNs technology. As with many technologies, the research in WSNs was driven by military applications. During the Cold War, a network of acoustic

sensors, named Sound Surveillance System (SOSUS), was deployed on the ocean bottom, for tracking Soviet submarines. This network was the ancestor of many modern WSNs used for acoustic Signal Processing applications, *e.g.*, the distributed echo cancellation system used in Hands-Free technologies, [85], the video conferencing noise canceler [84], etc. Nowadays, this system remains active and it is used for monitoring seismic and animal activity in the ocean, [37, 106]. Another important distributed signal processing application, which was developed during the Cold War, was the Airborne Warning and Control System (AWACS). This system consisted of multiple radars, acting as nodes of the network, which sensed information coming from the air and warned for possible hazards.

In the decade of the 80's, the Defense Advanced Research Projects Agency (DARPA) gave a boost in the distributed signal processing research field. The DARPA project was based on the Arpanet [99], which was the predecessor of the Internet. In a nutshell, the goal of this project was to study whether the basic principles of the Arpanet could be extended to WSNs. The proposed network consisted of cheap sensors, which were spatially distributed and collaborated with each other. A number of research challenges rose via the DARPA, such as: artificial intelligence issues, distributed signal processing techniques, routing methodologies, distributed software algorithms, just to name a few. These issues were summarized and discussed in the workshop [1]. The application, which demonstrated the results of the DARPA project, was the Distributed Acoustic Tracking problem.

In 1984, researchers at the Massachusetts Institute of Technology (MIT) studied distributed signal processing techniques for helicopter tracking applications [104]. The approach involved human heuristic methodologies. A generalization of this research, for multiple target tracking in a decentralized fashion, was studied at the same time in the Advanced Decision Systems (ADS). A multiple hypothesis tracking algorithm was developed in order to overcome limitations and problems, *e.g.*, false alarms, missing detections etc, [38].

The research in the early 90's focused, mainly, on military applications. A celebrated example was the network-centric warfare. In this application, sensors collected information, collaborated with each other and transmitted commands to the respective weapon/shooters *e.g.*, [3]. WSN's employed for military purposes in the decade of the 90's were also the Cooperative Engagement Capability developed by the U.S. Navy, the Advanced Deployable System the Remote Battlefield Sensor System, etc, (see also [37]).

### 2.1.2 Research trends and Applications involving Distributed Learning

In this section, we will briefly discuss typical recent real-life applications, which involve WSNs and decentralized processing.

Exploiting the technology advances of the 21st century, distributed signal processing and WSNs were put at the heart of a number of emergent applications. Two factors led to this direction. The first one was the advances in processors, which allowed the use of cheap and small sensors with considerable computational capabilities. The second one was the development of sophisticated communication protocols, such as the IEEE 802.11 standard, which increased significantly the available bandwidth. Celebrated examples of such applications include: the environmental and habitat monitoring, the forest fire detection, medical applications, such as the Body Area Network (BAN), spectrum allocation for cognitive radios and the traffic control.

- In environmental and habitat monitoring, a large number of sensors are scattered over a geographic region and the goal is to track climatic trends, measure the population of certain species, temperature/humidity and so on. The sensors measure acoustic image signals. Environmental monitoring applying WSNs takes place in the Center of Embedded Network Sensing project, [37], and in the System for the Vigilance of the Amazon, [71]. The latter project provides information regarding environmental issues and drug trafficking activities.

Possible anomalies of the region, onto which the sensors are deployed, may block the line of sight among them. Henceforth, a centralized network, which requires that all the nodes are able to communicate with the FC, can not be employed in environmental monitoring scenarios. This fact together with the need to consume a small bandwidth, brings out the need for the cooperation among the nodes of the network in an essentially decentralized mode of operation.

A very important environmental oriented application is the *forest fire detection*. Wireless sensors deployed in a forest are tasked to detect and provide crucial information about the origin of a fire, so as to prevent it from spreading uncontrollably. Usually, these sensors operate with solar energy and collaborate with each other so as to

overcome obstacles, such as large trees and rocks, which block their communication.

- WSNs play an important role in medical applications. A celebrated example is the Body Area Network (BAN), *e.g.* [74], which is envisioned to be widely used in the future. BAN is a network of sensors, which are placed in the body of a patient, and the goal is the early detection of hazardous conditions and their prevention if possible. Medical applications, in which the BAN contributes, include stroke rehabilitation, brain injury and surgery rehabilitation, blood pressure and temperature monitoring, etc.

Tracking activities that take place inside hospitals is another important medical based application, in which distributed processing is met. More specifically, patients have sensors attached to them. These provide information about the heart rate, the blood pressure, the temperature and so on. The information is sent to central data bases, which keep the medical history of the patient. Doctors also carry sensors, which give information about their location in the hospital. Last but not least, sensors can be embedded to medication, so as to reduce the probability of wrong prescription. More specifically, sensors are attached to the patients and the medications and if a patient happens to be allergic to an ingredient of a drug, then the interaction between them will prevent the patient from taking that drug.

- Radio spectrum allocation is an emergent application, which is envisioned to be at the forefront of research in the distributed signal processing field. The main idea is that a number of primary users transmit to certain bands, at specific time instances, whereas the rest, known as secondary users, collaborate with each other in order to find the unoccupied bands. The importance of this task is increased by the fact that modern devices, *e.g.*, smart phones or tablets, exchange a large amount of data and henceforth each one has to find the *spectrum holes* so as to transmit the essential information. The term spectrum hole stands for a band of frequencies, which is assigned to a primary user, but it is not being used by this user, at a certain time instance, *e.g.*, [64, 118]. A spectrum sensing technique has been proposed in [112]. This technique requires centralized operations, since an FC is considered. Fully decentralized approaches for spectrum sensing have been proposed in the works [7, 25]. This problem

will be discussed in more detail in section 2.3.3.

- Another important application, in which distributed signal processing in WSNs takes place is the so called Traffic Control. The concept is the following. Cheap sensors, which estimate the vehicle traffic, are deployed in road intersections. These sensors exchange information with each other and produce a global estimate for the traffic of a region. This knowledge can be exploited so as to inform the drivers about roads to be avoided, or to control the traffic lights. Another viewpoint to this problem was proposed in [56]. There, instead of deploying static sensors in intersections, the sensors are instead placed in the vehicles. When the vehicles meet, they exchange information regarding the location or the density of traffic jams. Notice that this application is totally decentralized, since the vehicles act as the nodes of the network.

## 2.2 Basic Principles of Distributed Learning and Diffusion optimization

In this section, the basic principles of distributed learning will be reviewed, under the diffusion viewpoint. This section serves as an introduction to the basic “tools” and notions, which will be used in this dissertation.

### 2.2.1 Motivation

A network consisting of  $N$  nodes is considered, and the goal is the estimation of an unknown, but common to all parameter vector  $\mathbf{w}_* \in \mathbb{R}^m$ . As it will become clear later on, this approach is general and many applications can be seen as special cases of the distributed estimation task. In the “ideal” scenario, where a central node is present, the estimation of the unknown vector can be carried out via a minimization of a properly chosen global cost function, *i.e.*,

$$\mathbf{w}_* = \arg \min_{\mathbf{w}} J_{\text{glob}}(\mathbf{w}). \tag{2.1}$$

The term global indicates that this cost function contains information coming from all the nodes of the network. On the contrary, in decentralized learning, where an FC is not present,

each node has access to the information obtained only by itself as well as its neighborhood; *i.e.*, the nodes of the network with which communication is possible. Fortunately, if each node cooperates with its neighbors and these also cooperate with their own neighbors, the goal of obtaining a solution at each node, which will be as close as possible to the global one, becomes feasible. This is exactly the main idea of decentralized learning; develop proper cooperation techniques, taking into consideration the constraints of the network and allowing only partial knowledge at each node, so as to reach a solution, which will be as close as possible to the centralized one. In the next sections, we will shed light on the cooperation techniques, and we will discuss methodologies for solving a special case of the optimization problem (2.1) in a fully decentralized fashion.

### 2.2.2 Networks and Neighborhoods

We consider an ad-hoc network, which is illustrated in Fig. 2.1 The node set is denoted by  $\mathcal{N} := \{1, 2, \dots, N\}$  and we assume that each node, say  $k$ , is able to communicate and exchange information with a subset of the node set, namely  $\mathcal{N}_k$ . For example, in Fig. 2.1 the neighborhood of node 1 comprises the following nodes:  $\mathcal{N}_1 = \{1, 3, 5\}$ . Throughout this dissertation, the following assumptions regarding the network are considered

- Each node is a neighbor of itself, or in mathematical terms  $k \in \mathcal{N}_k, \forall k \in \mathcal{N}$ .
- Symmetry is assumed, *i.e.*,  $l \in \mathcal{N}_k \Leftrightarrow k \in \mathcal{N}_l, \forall k, l \in \mathcal{N}$ .
- The network is assumed to be *connected*, or in other words, for any two nodes of the network, there exists at least one path connecting them. If the nodes are able to communicate, then the path is the direct link connecting them, otherwise the path is multihop, *i.e.*, it contains more than one nodes. For example, nodes 1 and 3 communicate directly, whereas nodes 1 and 7 communicate via the path  $1 \rightarrow 3 \rightarrow 7$ .

The number of neighbors of node  $k$ , which coincides with the cardinality of  $\mathcal{N}_k$ , *i.e.*,  $|\mathcal{N}_k|$ , defines the degree of this node. For example, node 3 has degree equal to 4.

As it has been already stated, each node receives information, which is transmitted by its neighborhood. Let us describe how this information is exploited. Fix an arbitrary node, say  $k$ . For each  $k$  a nonnegative weight is assigned to each neighbor which scales appropriately

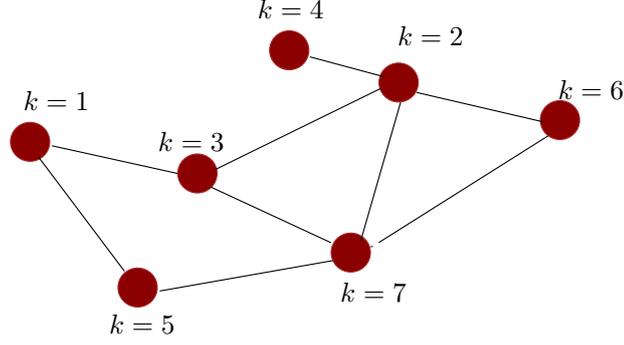


Figure 2.1: Illustration of an ad-hoc network.

the data coming from it. In the sequel, an aggregate of all these scaled data is computed. This information *fusion* is at the heart of the diffusion optimization rationale. The weights, which are also known as *combination weights*, will be denoted by  $c_{k,l}$ ,  $\forall k \in \mathcal{N}$ ,  $\forall l \in \mathcal{N}_k$ , where the subscripts  $k, l$  imply that the information transmitted by node  $l$  to node  $k$ , is weighted via  $c_{k,l}$ . Some typical examples of combination rules are:

- **Averaging Rule:**

$$c_{k,l} = \begin{cases} \frac{1}{|\mathcal{N}_k|}, & \text{if } l \in \mathcal{N}_k, \\ 0, & \text{otherwise} \end{cases}$$

- **Metropolis Rule:**

$$c_{k,l} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_k|, |\mathcal{N}_l|\}}, & \text{if } l \in \mathcal{N}_k, \text{ and } l \neq k \\ 1 - \sum_{l \in \mathcal{N}_k \setminus k} c_{k,l}, & \text{if } l = k \\ 0, & \text{otherwise.} \end{cases}$$

- **Relative Degree:**

$$c_{k,l} = \begin{cases} \frac{|\mathcal{N}_l|}{\sum_{j \in \mathcal{N}_k} |\mathcal{N}_j|}, & \text{if } l \in \mathcal{N}_k \\ 0, & \text{otherwise.} \end{cases}$$

All the previous combination rules share in common the properties:  $c_{k,l} = 0$ ,  $\forall l \notin \mathcal{N}_k$ ,  $c_{k,l} > 0$ ,  $\forall l \in \mathcal{N}_k$  and  $\sum_{l \in \mathcal{N}_k} c_{k,l} = 1$ . Notice that the combination weights provide us with information regarding the network, since if  $c_{k,l} = 0$  then  $l \notin \mathcal{N}_k$ , *i.e.*, node  $l$  does not belong to the neighborhood of node  $k$ . If  $l \in \mathcal{N}_k$  then the value of  $c_{k,l}$  is positive. Since

$c_{k,l} > 0$ ,  $\forall l \in \mathcal{N}_k$  and  $\sum_{l \in \mathcal{N}_k} c_{k,l} = 1$ , then each node computes a *convex* combination of the data coming from the neighborhood.

It is worth pointing out that in the Relative Degree rule, larger weights are assigned to those of the nodes, which have a large degree. More sophisticated approaches, which assign larger weights to the nodes which have smaller noise variances, and henceforth their information is more “reliable” compared to the noisier ones, have been proposed in the studies [72, 132].

**Remark 1.** *The topology of a network is determined by the deployment of the sensor/nodes over the geographical region. A commonly adopted topology is the incremental one, e.g., [90, 93]. Each sensor is able to exchange information with only one node and consequently the nodes constitute a ring pattern. Although this topology requires small communications bandwidth, it is not practical to be applied in networks with many nodes. This is a direct consequence of the fact that the construction and the maintenance of such a network requires a Hamiltonian Cycle, which is an NP hard task, e.g., [107]. Moreover, in a possible node failure, the network collapses. For these reasons, in many applications adopting the ad-hoc topology is preferable.*

### 2.2.3 Distributed Learning via Diffusion Optimization

In this subsection, the basic notions of diffusion optimization will be reviewed. Our goal is to explain in short, the principles that govern the cooperation among the nodes, and how these can be beneficial to the learning procedure.

As we have already stated, the ultimate goal is the estimation of an unknown vector, say  $\mathbf{w}_* \in \mathbb{R}^m$ , which is common to the nodes of the network. Throughout this dissertation, the linear regression model will be adopted, *i.e.*, we consider that each node,  $k$ , at each discrete time instance  $n$ , has access to a scalar  $d_{k,n} \in \mathbb{R}$  and a vector  $\mathbf{u}_{k,n} \in \mathbb{R}^m$ , which are related via

$$d_{k,n} = \mathbf{u}_{k,n}^T \mathbf{w}_* + v_{k,n}. \quad (2.2)$$

The term  $v_{k,n}$  stands for the additive noise process. As we will see later on, the linear system is very general and many real-world applications can be modeled as special cases of (2.2). Several techniques and criteria have been proposed for the estimation of  $\mathbf{w}_*$ , when the

measurements obey (2.2). For illustrative purposes here, we adopt the Mean Square Error (MSE) criterion, *e.g.*, [63, 120]:

$$J_k(\mathbf{w}) = \mathbb{E}[d_{k,n} - \mathbf{u}_{k,n}^T \mathbf{w}]^2. \quad (2.3)$$

Firstly, we will describe the methodology for the minimization of  $J_k(\mathbf{w})$  in the scenario where each node operates individually; *i.e.*, when it does not cooperate with its neighbors. The random processes,  $d_{k,n}$ ,  $\mathbf{u}_{k,n}$ , are assumed to be zero-mean and jointly-wide sense stationary, *e.g.*, [120]. Let us define the following second order quantities:

$$\mathbf{R}_k = \mathbb{E}\mathbf{u}_{k,n}\mathbf{u}_{k,n}^T \quad (2.4)$$

$$\mathbf{p}_k = \mathbb{E}\mathbf{u}_{k,n}d_{k,n} \quad (2.5)$$

$$\sigma_{d,k}^2 = \mathbb{E}d_{k,n}^2 \quad (2.6)$$

If we expand (2.3) we obtain

$$J_k(\mathbf{w}) = \sigma_{d,k}^2 - 2\mathbf{w}^T \mathbf{p}_k + \mathbf{w}^T \mathbf{R}_k \mathbf{w}. \quad (2.7)$$

The cost function  $J_k(\mathbf{w})$  is differentiable and convex (see also Chapter 3), hence to attain its minimum value, the gradient  $\nabla J_k(\mathbf{w})$  must equal to zero. The resulting relation is the following:

$$\mathbf{w}_o = \mathbf{R}_k^{-1} \mathbf{p}_k, \quad (2.8)$$

where the autocorrelation matrix  $\mathbf{R}_k$  is assumed to be positive definite and henceforth invertible. It has been verified, *e.g.*, [120], that the optimum point,  $\mathbf{w}_o$ , obtained by the minimization of  $J_k(\mathbf{w})$  coincides with the unknown vector  $\mathbf{w}_*$ . The minimum value of the loss function equals:

$$J_{k,\min} = J(\mathbf{w}_*) = \sigma_{d,k}^2 - \mathbf{p}_k^T \mathbf{R}_k^{-1} \mathbf{p}_k. \quad (2.9)$$

Despite the fact that the minimization of  $J_k(\mathbf{w})$  leads to the unknown solution, knowledge on the statistics  $\mathbf{R}_k$ ,  $\mathbf{p}_k$  is required, which is not always feasible. However, this problem can be overstepped, as it will become clear in the next chapters, if one resorts to adaptive techniques, which estimate  $\mathbf{w}_*$  using the measurement data ( $d_{k,n}$ ,  $\mathbf{u}_{k,n}$ ) obtained at each time

instance. Adaptive techniques can also be accommodated in scenarios where the unknown vector and/or the statistics undergo changes.

So far, we have considered the case where each node uses local knowledge only, *i.e.*, the statistics  $\mathbf{R}_k$ ,  $\mathbf{p}_k$ , so as to estimate  $\mathbf{w}_*$ . In this scenario, the fact that all the nodes try to estimate *the same* unknown vector is not taken into consideration. However, it has been verified that the cooperation among the nodes is beneficial to the performance of the developed algorithms, *e.g.*, [28]. To this end, we will shed some light on how the nodes can collaboratively estimate the unknown vector. First of all, let us define the global cost function:

$$J_{\text{glob}}(\mathbf{w}) = \sum_{k \in \mathcal{N}} J_k(\mathbf{w}). \quad (2.10)$$

It is obvious that this cost function is the “optimum” one in the sense that it contains information coming from the whole network and its minimization requires the existence of an FC. Now, let us define the local cost at each node, given by:

$$J_{k,\text{loc}}(\mathbf{w}) = \sum_{l \in \mathcal{N}_k} c_{k,l} J_l(\mathbf{w}), \quad (2.11)$$

where  $c_{k,l}$  are the weights, which are defined in subsection 2.2.2. Notice that since  $c_{k,l} = 0$ ,  $l \notin \mathcal{N}_k$ , then each node uses information coming from its neighborhood. Hence, the minimization of (2.11) can take place in a fully decentralized way. Indeed, the global cost is related to the local costs as follows:

$$J_{\text{glob}}(\mathbf{w}) = \sum_{l \in \mathcal{N}} \sum_{k \in \mathcal{N}} c_{k,l} J_l(\mathbf{w}) \quad (2.12)$$

$$= \sum_{k \in \mathcal{N}} \sum_{l \in \mathcal{N}} c_{k,l} J_l(\mathbf{w}) \quad (2.13)$$

$$= \sum_{k \in \mathcal{N}} J_{k,\text{loc}}(\mathbf{w}) \quad (2.14)$$

Let us now reformulate the local optimization problem, which involves the minimization of  $J_{k,\text{loc}}(\mathbf{w})$ , as follows:

$$\min_{\mathbf{w}} J_{k,\text{loc}}(\mathbf{w}) + \lambda \frac{1}{2} \|\mathbf{w} - \mathbf{w}_*\|^2. \quad (2.15)$$

The reasoning of the extra term is to lead the obtained solution to lie “close” to the unknown vector, pretending that it is “known”. Its contribution is weighted by the regularization parameter  $\lambda$ . A classical technique, which can be employed for the minimization of this constrained optimization problem is the steepest–descent iterative method (see for example [120]). Employing the steepest–descent, in order to find a solution to the problem described in (2.15), the following iterative scheme occurs:

$$\mathbf{w}_{k,n+1} = \mathbf{w}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (\mathbf{p}_l - \mathbf{R}_l \mathbf{w}_{k,n}) + \mu_k \lambda (\mathbf{w}_* - \mathbf{w}_{k,n}), \quad (2.16)$$

where  $\mathbf{w}_{k,n}$  denotes the estimate obtained at the  $n$ -th iteration at node  $k$ , and  $\mu_k$  is the step–size. Obviously,  $\mathbf{w}_*$  is unknown and it cannot be employed in the iterative scheme of (2.16). So, we substitute it with a combination of the most recent estimates, which are obtained by the neighborhood. Thus, we reformulate (2.16) to obtain:

$$\boldsymbol{\psi}_{k,n} = \mathbf{w}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (\mathbf{p}_l - \mathbf{R}_l \mathbf{w}_{k,n}) \quad (2.17)$$

$$\mathbf{w}_{k,n+1} = \boldsymbol{\psi}_{k,n} + \mu_k \lambda \left( \sum_{l \in \mathcal{N}_k \setminus k} b_{k,l} \boldsymbol{\psi}_{l,n} - \boldsymbol{\psi}_{k,n} \right), \quad (2.18)$$

where for the weights  $b_{k,l}$ , it holds  $\sum_{l \in \mathcal{N}_k \setminus k} b_{k,l} = 1$ . Notice that, in contrast to (2.16), equation (2.18) uses  $\boldsymbol{\psi}_{k,n}$  instead of  $\mathbf{w}_{k,n}$  in the term included due to the constraint, this is because the former is generally a better estimate of  $\mathbf{w}_*$  compared to the latter. Assume a sufficiently small  $\mu_k$ , which guarantees that  $0 < (1 - \mu_k \lambda) < 1$ , and define  $a_{k,k} := 1 - \mu_k \lambda$ ,  $\forall k \in \mathcal{N}$  and  $a_{k,l} = \mu_k \lambda b_{k,l}$ . It is obvious that the weights  $a_{k,l}$  satisfy the properties  $a_{k,l} > 0, l \in \mathcal{N}_k$ ,  $a_{k,l} = 0, l \notin \mathcal{N}_k$  and  $\sum_{l \in \mathcal{N}_k} a_{k,l} = 1$ . Plugging the weights  $a_{k,l}$  in equations (2.17), (2.18), we obtain:

$$\boldsymbol{\psi}_{k,n} = \mathbf{w}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (\mathbf{p}_l - \mathbf{R}_l \mathbf{w}_{k,n}) \quad (2.19)$$

$$\mathbf{w}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{k,l} \boldsymbol{\psi}_{l,n}. \quad (2.20)$$

Equations (2.19), (2.20), describe the celebrated *Adapt–Combine* (A–C) Diffusion strategy.

This cooperation strategy is named this way because in step (2.19) each node performs a learning operation, based on the statistical quantities obtained by the neighborhood, and in step (2.20) the nodes combine and fuse the previously computed estimates. In Chapter 3, we will study adaptive techniques, which follow the A–C cooperation strategy.

Equations (2.16) can be rewritten as follows:

$$\boldsymbol{\psi}_{k,n} = \mathbf{w}_{k,n} + \mu_k \lambda (\mathbf{w}_* - \mathbf{w}_{k,n}) \quad (2.21)$$

$$\mathbf{w}_{k,n+1} = \boldsymbol{\psi}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (\mathbf{p}_l - \mathbf{R}_l \mathbf{w}_{k,n}). \quad (2.22)$$

Using similar arguments as in the A–C diffusion cooperation strategy we conclude that:

$$\boldsymbol{\psi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{w}_{l,n} \quad (2.23)$$

$$\mathbf{w}_{k,n+1} = \boldsymbol{\psi}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (\mathbf{p}_l - \mathbf{R}_l \boldsymbol{\psi}_{k,n}). \quad (2.24)$$

Equations (2.23) and (2.24) constitute the *Combine–Adapt* (C–A) Diffusion strategy. It can be readily seen that in this cooperation strategy, each node fuses the information received by its neighbors and then uses this aggregate in the adaptation step. In [136], the A–C strategy is compared to the C–A one, in terms of computational complexity and performance. There, it was proved that if one employs an LMS–based algorithm in the adaptation step then the A–C strategy outperforms the C–A one.

**Remark 2.** *In the diffusion optimization strategy, the cooperation among the nodes of the network can be seen as a “combination” of the information obtained by the neighborhood, which is determined by the weights  $c_{k,l}$ ,  $a_{k,l}$ . According to the previous discussion, the combination of the estimates of the neighboring nodes, is the consequence of a constraint, the goal of which is to lead them to be “close” to each other. Another viewpoint is the one that runs across the so-called consensus–based algorithms. According to this rationale, constraints which force the estimates at the nodes of the network to be the same, are employed. Usually, the resulting optimization problem can be solved via the Alternating Direction Minimization Multipliers, e.g., [10]. Consensus–based algorithms have been adopted in several distributed learning tasks, such as, distributed adaptive filtering, e.g., [98, 123], distributed linear estima-*

tion, e.g., [115], collaborating spectrum sensing, e.g., [7], just to name a few. Nevertheless, it has been shown in [136], that in the distributed adaptive learning task, the diffusion based algorithms outperform the consensus based ones.

**Remark 3.** The combination weights  $c_{k,l}$  in (2.20) and (2.24) indicate that the nodes exchange the input/output statistics, i.e.,  $\mathbf{p}_k$ ,  $\mathbf{R}_k$ , or as we will see later on, in the adaptive operation mode, their respective measurements,  $d_{k,n}$ ,  $\mathbf{u}_{k,n}$ . However, this information exchange and exploitation increases the bandwidth demands as well as the computational complexity. To this end, in many applications, it is assumed that  $c_{k,k} = 1$ ,  $\forall k \in \mathcal{N}$  and  $c_{k,l} = 0$ ,  $\forall k \in \mathcal{N}$ ,  $\forall l \in \mathcal{N}_k \setminus k$ , or in other words, each node exploits its local statistical (or measurement) information, and combines only the estimates of its neighborhood.

## 2.3 Applications of Distributed Learning and Diffusion optimization

In this section, we will discuss three distributed learning paradigms, in order to illustrate how the previously presented arguments, regarding the diffusion philosophy, can be employed to attack practical applications. The first one is the so-called target localization, in which we consider a network, where the nodes are interested in estimating the location of a certain target. The second application is the distributed averaging one. More specifically, each node of the network has access to a certain value and the nodes are tasked to estimate the average of all these values. Finally, we will describe the problem of collaborative spectrum sensing, in which a number of nodes, the so-called Secondary Users (SUs), wish to estimate unoccupied spectrum bands so as to transmit information.

### 2.3.1 Distributed Learning met in nature

As we have already discussed, at the heart of distributed learning is the concept of collaborative decision making. That is, the nodes make decisions exploiting: a) the information which is locally known and b) the information received by their neighbors. The roots of this philosophy stem from basic principles of the nature. A celebrated example is the first communities, in which ancient people organized, so as to fulfill collaboratively their basic

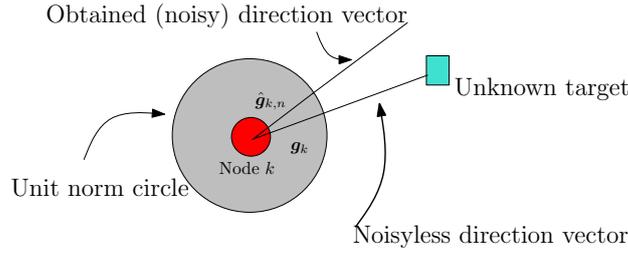


Figure 2.2: Illustration of a node, the target source and the direction vectors.

needs for food, sheltering, etc. Many paradigms drawn from the nature can be modeled as decentralized networks, with nodes whose purpose is to achieve the same goal. Networks consisting of living organisms are also known as *biological networks*. In the sequel, we present an example of a biological network, the nodes of which have the goal of estimating a specific target; we will explain in brief how the diffusion rationale can be employed in this problem.

### Target Localization

Consider a network consisted of  $N$  nodes, whose goal is to estimate and track the location of a specific target. The nodes can possibly represent fish schools, which seek for a nutrition source, *e.g.*, [138], bee swarms, which search for their hive, *e.g.*, [88], bacteria seeking for nutritive sources, *e.g.*, [33]. The location of the unknown target, say  $\mathbf{w}_*$ , is assumed to belong to the two-dimensional space, with definition  $\mathbf{w}_* = [w_*^{(1)}, w_*^{(2)}]^T$ , where  $w_*^{(1)}$  and  $w_*^{(2)}$  are its coordinates. The position of each node is denoted by  $\mathbf{w}_k = [w_k^{(1)}, w_k^{(2)}]^T$ , and the true distance between node  $k$  and the unknown target equals to

$$r_k = \|\mathbf{w}_* - \mathbf{w}_k\|. \quad (2.25)$$

The vector, whose direction points from the node  $k$  to the unknown source, is given by:

$$\mathbf{g}_k = \frac{\mathbf{w}_* - \mathbf{w}_k}{\|\mathbf{w}_* - \mathbf{w}_k\|}. \quad (2.26)$$

Obviously, the distance equation can be rewritten in terms of the direction vector as follows:

$$r_k = \mathbf{g}_k^T (\mathbf{w}_* - \mathbf{w}_k). \quad (2.27)$$

### 2.3 Applications of Distributed Learning and Diffusion optimization

---

Usually, it is assumed that  $k$  senses the distance and the direction vectors via noisy observations. Following a similar rationale as in [138], the noisy distance scalar can be modeled as follows:

$$\hat{r}_{k,n} = r_k + v_{k,n}, \quad (2.28)$$

where  $n$  stands for the discrete time instance and  $v_{k,n}$  for the additive noise term. The noise in the direction vector is a consequence of two effects: a) a deviation occurring along the direction, which is perpendicular to  $\mathbf{g}_k$  and b) a deviation that takes place along the parallel direction of  $\mathbf{g}_k$  (see Fig. 2.2). All in one, the noisy direction vector, occurring at time instance  $n$ , can be written as:

$$\hat{\mathbf{g}}_{k,n} = \mathbf{g}_k + v_{k,n}^\perp \mathbf{g}_k^\perp + v_{k,n}^\parallel \mathbf{g}_k, \quad (2.29)$$

where  $v_n^\perp$  is the noise corrupting the unit norm perpendicular direction vector, *i.e.*,  $\mathbf{g}_k^\perp$  and  $v_n^\parallel$  is the noise occurring at the parallel direction vector. Taking into consideration the noisy terms, (2.28) is given by

$$\hat{r}_{k,n} = \hat{\mathbf{g}}_{k,n}^T (\mathbf{w}_* - \mathbf{w}_k) + \eta_{k,n}, \quad (2.30)$$

where

$$\eta_{k,n} = v_{k,n} - v_{k,n}^\perp \mathbf{g}_k^{\perp T} (\mathbf{w}_* - \mathbf{w}_k) - v_{k,n}^\parallel \mathbf{g}_k^T (\mathbf{w}_* - \mathbf{w}_k). \quad (2.31)$$

Equation (2.30) can be further simplified if one recalls that by construction  $\mathbf{g}_k^{\perp T} (\mathbf{w}_* - \mathbf{w}_k) = \mathbf{0}_2$ . Moreover, typically the contribution of  $v_{k,n}^\perp$  is assumed to be significantly larger than the contribution of  $v_{k,n}^\parallel$ . Henceforth, taking into consideration these two arguments, (2.31) can be rewritten as:

$$\eta_{k,n} \approx v_{k,n}. \quad (2.32)$$

If one defines  $d_{k,n} := \hat{r}_{k,n} + \hat{\mathbf{g}}_{k,n}^T \mathbf{w}_k$  and combines (2.30) with (2.32) the following model results:

$$d_{k,n} \approx \hat{\mathbf{g}}_{k,n}^T \mathbf{w}_* + v_{k,n}. \quad (2.33)$$

Equation (2.33) is a special case of the previously described *linear model*. Henceforth, the “tools” discussed in previous sections can be employed for the estimation of the unknown target source, *i.e.*,  $\mathbf{w}_*$ . The study in [138] proposes an LMS-based algorithm for the compu-

tation of this vector. As it will become clear later on, this algorithm is suitable for environments, in which the position of the source is changing, and, henceforth, the nodes have to be able to track these alterations. It is worth pointing out that the cooperation principles, which are described in subsection 2.2.2, are adopted in [138], so that the nodes estimate the position of the unknown source in a collaborative way. More specifically, each node minimizes the cost function  $J_{k,loc}(\mathbf{w}) = \sum_{l \in \mathcal{N}_k} c_{k,l} J_l(\mathbf{w})$ , where  $J_l(\mathbf{w}) = \mathbb{E}(d_{l,n} - \mathbf{w}^T \hat{\mathbf{g}}_{l,n})^2$ . Then the A–C or the C–A diffusion scheme can be employed for the computation of  $\mathbf{w}_*$ . Indeed, it has been verified that the information exchange and fusion enhances significantly the ability of the nodes to estimate and track the target source.

### 2.3.2 Distributed Averaging

A problem of great importance met in distributed signal processing, is the *distributed averaging* or *distributed consensus* task. The general concept can be summarized as follows. Given an  $N$  node network, where each node has access to a scalar quantity,  $x_k, \forall k \in \mathcal{N}$ , the goal is to reach *consensus* to the average value,  $w_*$  or in other words, the nodes are interested in estimating:

$$w_* = \frac{1}{N} \sum_{k \in \mathcal{N}} x_k. \quad (2.34)$$

The distributed consensus problem is met in numerous applications, such as: data fusion [119], vehicle formation [57], distributed inference [77], just to name a few. A straightforward solution to this problem is to employ a FC, which collects the scalar values from the nodes of the network, computes  $w_*$  and transmits it back to the nodes. Nevertheless, as it has been already stated, this solution presents many drawbacks. We are interested, henceforth, to solve the problem in a fully decentralized way. Another possible technique is the flooding one. According to this, the nodes transmit their values to the neighborhood and in the sequel, the neighbors transmit the information to their neighbors and so on. This procedure is completed when every node has access to all the data and is able to compute the desired average value. The flooding methodology is not robust due to the fact that in large networks the nodes have to store a large number of values. Moreover, if the network is sparse, *i.e.*, the number of links of the network is small, if the latter represented as a graph, then a significant delay may occur. Finally, this technique consumes a large amount of the bandwidth resources,

since nodes with many neighbors have to transmit their values to the other nodes, with which communication is possible.

Another more efficient route is to resort to techniques, which compute  $w_*$ , exploiting solely information coming from the neighborhood, *e.g.*, [51, 146, 147]. More specifically, each node collects data from the neighborhood, fuses them under a certain rule and the goal is to achieve consensus to the average value. Consider the following iterative scheme at each node

$$x_{k,n} = \sum_{l \in \mathcal{N}} c_{k,l} x_{l,n-1}, \quad (2.35)$$

with initial values  $x_{k,0} = x_k, \forall k \in \mathcal{N}$ . The combination weights  $c_{k,l}$  can be chosen with respect to several rules, *e.g.*, the Metropolis rule presented in subsection 2.2.2. If one recalls the property  $c_{k,l} > 0, \forall l \in \mathcal{N}_k, c_{k,l} = 0, \forall l \notin \mathcal{N}_k$ , (2.35) trivially implies that each node exploits neighborhood information, which is in line with the basic principles of decentralized processing. Gathering all  $x_{k,n}$ 's in a vector  $\mathbf{x}_n = [x_{1,n}, \dots, x_{N,n}]^T \in \mathbb{R}^N$ , then (2.35) can be written equivalently for the whole network as follows:

$$\mathbf{x}_n = \mathbf{W} \mathbf{x}_{n-1}, \quad (2.36)$$

where  $\mathbf{W}$  is an  $N \times N$  matrix with entries given by  $c_{k,l}$ . Consider that the matrix  $\mathbf{W}$  satisfies the following properties:

1.  $\mathbf{1}_N^T \mathbf{W} = \mathbf{1}_N^T$ , where  $\mathbf{1}_N \in \mathbb{R}^N$  is the vector of ones.
2.  $\mathbf{W} \mathbf{1}_N = \mathbf{1}_N$ .
3.  $\tau(\mathbf{W} - \mathbf{1}_N \mathbf{1}_N^T / N) < 1$ , where  $\tau(\cdot)$  stands for the maximum eigenvalue of the respective matrix.

Property 1 implies that the vector of ones is a left eigenvector of  $\mathbf{W}$ , which in turn indicates that  $\mathbf{1}_N^T \mathbf{x}_n = \mathbf{1}_N^T \mathbf{x}_{n-1} = \dots = \mathbf{1}_N^T \mathbf{x}_0$ . The last relation shows that the sum of the individual values, therefore their average, is preserved at each step. From Property 2, it can be seen that  $\mathbf{1}_N$  is a right eigenvector of  $\mathbf{W}$  and it is a fixed point of (2.36). Finally, combining the previous properties with Property 3, it can be proved, *e.g.*, [146], that 1 is an eigenvalue of the combination matrix  $\mathbf{W}$ . Methodologies for constructing the combination matrix so as to fulfill these properties in a fully decentralized way, have been proposed in, *e.g.*, [146].

**Theorem 1.** Assume that the matrix  $\mathbf{W}$  satisfies Properties 1-3. Then it holds that [146]:

$$\lim_{n \rightarrow \infty} \mathbf{x}_n = \mathbf{w}_*, \quad (2.37)$$

with  $\mathbf{w}_* = [w_*, \dots, w_*]^T \in \mathbb{R}^N$ . In words, each node gains access to the desired average value.

This theorem is a direct consequence of the fact that the assumptions regarding the combination coefficients ensure that

$$\lim_{n \rightarrow \infty} \mathbf{W}^n = \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T. \quad (2.38)$$

Another important class of distributed consensus algorithms is the one of *randomized gossip algorithms*, e.g., [16,51]. The goal remains the same, i.e., the nodes desire to compute the average value, as in the classical consensus averaging algorithms. The difference lies in the fact that at each time instance, a single node exchanges and fuses information with a single neighbor. Moreover, this procedure takes place in a stochastic way, since a random node of the network and a random neighbor of this node, are picked at each time instance. Fix a certain time instance  $n$  and consider that node  $k$  is chosen at random, to exchange information with node  $l$ . At the end of this gossip round,  $k$  and  $l$  will have access to the values  $x_{k,n} = \frac{x_{k,n-1} + x_{l,n-1}}{2}$ . This procedure can be written compactly for the whole network as follows:

$$\mathbf{x}_n = \mathbf{W}_n \mathbf{x}_{n-1}, \quad (2.39)$$

where the matrix  $\mathbf{W}_n$ , which is time-varying in this scenario, has two non-zero entries, at the positions  $k, l$  and  $l, k$ , and they equal to  $1/2$ . Despite the fact that the desired behavior is the same as in classical distributed consensus averaging, i.e.,  $\lim_{n \rightarrow \infty} \mathbf{x}_n = \mathbf{w}_*$ , the analysis is different due to the stochastic nature of the matrix  $\mathbf{W}_n$ . First of all, assume that the combination matrices are selected independently and  $\mathbb{E} \mathbf{W}_n = \mathbb{E} \mathbf{W}_0 = \mathbb{E} \mathbf{W}$ ,  $\forall n \in \mathbb{Z}$ , where the time instance is suppressed. Employing the expectation operator in (2.39) we have:

$$\mathbb{E} \mathbf{x}_n = \mathbb{E} \left( \prod_{i=0}^{n-1} \mathbf{W}_i \right) \mathbf{x}_0 = (\mathbb{E} \mathbf{W})^n \mathbf{x}_0, \quad (2.40)$$

where the second equality holds since the matrices are selected independently. Notice that Properties 1,2 hold true for the matrix  $\mathbf{W}_n$ ,  $\forall n \in \mathbb{N}$ , and, hence, they hold for  $\mathbb{E}\mathbf{W}$ . If Property 3 is valid for the matrix  $\mathbb{E}\mathbf{W}$ , *i.e.*,  $\tau(\mathbb{E}\mathbf{W} - \mathbf{1}_N \mathbf{1}_N^T / N) < 1$  then

$$\lim_{n \rightarrow \infty} \mathbb{E}\mathbf{x}_n = \mathbf{w}_*, \quad (2.41)$$

*i.e.*, the nodes reach consensus in the mean. Unfortunately, the convergence in the mean property provides us with little information about the behavior of the algorithm in terms of convergence speed. To this end, the  $\check{\epsilon}$ -averaging time is introduced. Given a positive  $\check{\epsilon} > 0$  and an initial vector  $\mathbf{x}_0$  the  $\check{\epsilon}$ -averaging time equals to:

$$T_{\text{av}}(\check{\epsilon}) = \sup_{\mathbf{x}_0} \inf_n \left\{ \mathbb{P} \left( \frac{\|\mathbf{x}_n - \mathbf{w}_*\|}{\|\mathbf{x}_0\|} \geq \check{\epsilon} \right) \right\} \leq \check{\epsilon}. \quad (2.42)$$

The  $\check{\epsilon}$ -averaging time indicates the earliest time index  $n$ , in which the distance between the networkwise vector  $\mathbf{x}_n$  and  $\mathbf{w}_*$  is smaller than  $\check{\epsilon}$  with probability greater than  $1 - \check{\epsilon}$ . This measure is very useful because it helps us to bound the number of iterations, which are needed in order to achieve a certain performance (dictated by  $\check{\epsilon}$ ).

**Theorem 2.** *For any randomized consensus algorithm, which converges in the mean, the  $\check{\epsilon}$ -averaging time is bounded by [16]:*

$$T_{\text{av}}(\check{\epsilon}) \leq \frac{3 \log \check{\epsilon}^{-1}}{1 - \tau_2(\mathbb{E}\mathbf{W})}, \quad (2.43)$$

where  $\tau_2(\mathbb{E}\mathbf{W})$  is the second largest eigenvalue of the matrix  $\mathbb{E}\mathbf{W}$ .

Taking a closer look at (2.43), it can be obtained that the desired accuracy is dictated by: a) the distance  $\check{\epsilon}$  and b) the parameter  $\tau_2(\mathbb{E}\mathbf{W})$ . The first factor indicates that a smaller distance between the desired vector and the obtained one, requires a larger number of iterations, which is an expected behavior. The second factor, *i.e.*, the second largest eigenvalue contains information about the topology of the network and it has been employed in several works, *e.g.*, [14, 58]. For this eigenvalue it holds that  $\tau_2(\mathbb{E}\mathbf{W}) \leq 1$ . In order to guarantee asymptotic consensus, the topology of the network has to be properly chosen so as to ensure that the eigenvalue will be strictly less than 1. Furthermore, from (2.43), it

can be seen that if  $\tau_2(\mathbb{E}\mathbf{W})$  is small, then the  $\check{\epsilon}$ -averaging time requires a smaller number of iterations, in order to achieve a certain accuracy  $\check{\epsilon}$ .

Apart from the network topology, the value of  $\tau_2(\mathbb{E}\mathbf{W})$  is determined by the probability distribution at the edges of the network [15]. Recall from the previous discussion that at each time instance a random node of the network, say  $k$ , and a random neighbor of this node,  $l$ , are chosen. We consider that the edge  $k, l$  is chosen with probability equal to  $P_{k,l}$ . In order to accelerate the convergence speed, via reducing the  $\check{\epsilon}$ -averaging time, [15] proposes the following optimization problem:

$$\text{minimize } \tau_2(\mathbb{E}\mathbf{W}) \quad (2.44)$$

$$\text{s.t. } \sum_{k \in \mathcal{N}, l \in \mathcal{N}_k} P_{k,l} = 1 \quad (2.45)$$

$$P_{k,l} \geq 0. \quad (2.46)$$

As it has been shown in [15], this problem is convex and can be solved using semi-definite programming techniques, *e.g.*, [17, 87].

**Remark 4.** *In many applications involving WSNs, exchanging real valued scalars may be prohibited due to bandwidth limitations. This limitation can be overstepped if a quantization process takes place and, henceforth, the value of a transmitted coefficient is drawn from a finite alphabet. Obviously in this scenario, there are no guarantees that the nodes will reach consensus to the average value, due to the quantization distortion. Nevertheless, in the studies [9, 81, 86], the quantized consensus algorithms are shown to converge to a value, the distance of which from the true average, is bounded by known constants.*

**Remark 5.** *In the previous discussion, perfect communication among the nodes was assumed. This means that the values transmitted throughout the network are not corrupted with noise. However, this is not always the case. In numerous applications the communication among the nodes is non-ideal and it has to be taken into consideration. Usually, the non-ideal communication is modeled via additive noise or packet loss. In the first scenario, each node receives the value sent by its neighbor corrupted by additive noise, whereas in the second it is assumed that certain packets do not arrive to the receiver nodes. In [78, 114, 144] independent failures in space are considered, *i.e.*, failures that occur independently at the*

nodes of the network, and in time, that is link failure events are temporary independent. However, all these studies consider a noiseless environment. Noisy information exchange has been considered in [62, 69]. Consensus is guaranteed by employing decreasing weight sequences. Finally, the study in [79] assumes both packet losses as well as additive noise corruption.

### 2.3.3 Spectral Sensing for Cognitive Radios

In this section, we will describe the basic principles of Spectral Sensing in Cognitive Radio systems. Moreover, we will explain how the basic issues of linear estimation using diffusion optimization can be employed in this paradigm.

Cognitive radios comprise two types of users: the primary users (PU) and the secondary users (SU). Secondary users need to detect unoccupied frequency bands, *i.e.*, frequency bands in which primary users are not active, and transmit information in these. This is important since if a SU sends information in a band occupied by a PU, then harmful interference effects may occur, *e.g.*, [64, 118]. A possible methodology, in order to carry out the spectral sensing task at each SU, is the following. Each SU estimates the aggregated power spectrum, which is transmitted by the PUs, and afterwards locates the unused frequency bands. Consider an environment with  $P$  primary users and  $S$  secondary ones. Let us define  $S_p(e^{j\omega})$  to be the power spectrum transmitted by PU  $p$ ,  $p = 1, \dots, P$ , where  $j = \sqrt{-1}$  and  $\omega \in [-\pi, \pi]$  is the normalized angular frequency. It is assumed that the power spectrum can be represented as a linear combination of  $B$  functions  $f_i(e^{j\omega})$  as follows:

$$S_p(e^{j\omega}) = \sum_{i=1}^B v_{p,i} f_i(e^{j\omega}), \quad p = 1, \dots, P, \quad (2.47)$$

where  $v_{p,i}$  denotes the coefficients of the basis expansion for user  $p$ . A typical example of the basis function is the Gaussian pulse, which is given by  $f_i(e^{j\omega}) := e^{-\frac{(\omega-\omega_i)^2}{\sigma_i^2}}$ , where the parameters  $\omega_i$ ,  $\sigma_i^2$  are assumed to be known. Fix a certain PU, say  $p$ , and collect all the coefficients  $v_{p,i}$  in a vector:  $\mathbf{v}_p := [v_{p,1}, v_{p,2}, \dots, v_{p,B}]^T \in \mathbb{R}^B$ , as well as the basis functions in the following vector  $\mathbf{f}_\omega := [f_1(e^{j\omega}), f_2(e^{j\omega}), \dots, f_B(e^{j\omega})]^T \in \mathbb{R}^B$ . Then, (2.47) can be written

compactly as follows:

$$S_p(e^{j\omega}) = \mathbf{f}_\omega^T \mathbf{v}_p. \quad (2.48)$$

Now, let us denote with  $q_{p,k}$  the path loss scalar from PU  $p$  to SU  $k$ . It is assumed that a training procedure takes place so that the path loss coefficients are known to the system. These coefficients are allowed to be slowly time-varying and hence the training stage, through which they are estimated, has to be repeated at regular time intervals. Each SU  $k$  senses information coming from all active PUs. Thus the power spectrum which arrives at SU  $k$  is given by:

$$\begin{aligned} S_k(e^{j\omega}) &= \sum_{i=1}^P q_{i,k} S_i(e^{j\omega}) + \sigma_k^2 \\ &= \sum_{i=1}^P q_{i,k} \mathbf{f}_\omega^T \mathbf{v}_i + \sigma_k^2 \\ &= \mathbf{u}_{k,\omega}^T \mathbf{w}_* + \sigma_k^2, \end{aligned} \quad (2.49)$$

where  $\sigma_k^2$  stands for the receiver noise power,  $\mathbf{u}_{k,\omega} = [q_{1,k} \mathbf{f}_\omega^T, \dots, q_{P,k} \mathbf{f}_\omega^T]^T \in \mathbb{R}^{BP}$ ,  $\mathbf{w}_* = [\mathbf{v}_1^T, \dots, \mathbf{v}_P^T]^T \in \mathbb{R}^{BP}$ . At each time instance, node  $k$  senses the power spectrum  $S_k(e^{j\omega})$  at  $R$  different frequency samples,  $\omega_1, \omega_2, \dots, \omega_R$ , which lie in the interval  $[-\pi, \pi]$ . To this end, the following model rises

$$d_{k,r}(n) = S_k(e^{j\omega_r}) = \mathbf{u}_{k,\omega_r} \mathbf{w}_* + \sigma_k^2 + v_{k,r}(n), \quad r = 1, \dots, R, \quad (2.50)$$

where  $v_{k,r}(n)$  denotes the sampling noise. Gathering the  $R$  measurements which are collected at node  $k$  at time instance  $n$ , we obtain:

$$\mathbf{d}_{k,n} = \mathbf{U}_{k,n} \mathbf{w}_* + \mathbf{v}_{k,n}, \quad (2.51)$$

where  $\mathbf{d}_{k,n} = [d_{k,1}(n) - \sigma_k^2, \dots, d_{k,R}(n) - \sigma_k^2]^T \in \mathbb{R}^R$ ,  $\mathbf{v}_{k,n} = [v_{k,1}(n), \dots, v_{k,R}(n)]^T \in \mathbb{R}^R$  and the matrix  $\mathbf{U}_{k,n} = [\mathbf{u}_{k,\omega_1}^T, \mathbf{u}_{k,\omega_2}^T, \dots, \mathbf{u}_{k,\omega_R}^T]^T$ , of dimension  $R \times BP$ .

Recall that the SUs wish to estimate the aggregate power spectrum of the PUs, in order

to decide whether to transmit or not. This is equivalent to estimating  $\mathbf{w}_*$  since:

$$\sum_{p=1}^P S_p(e^{j\omega}) = (\mathbf{1}_P \otimes \mathbf{f}_\omega) \mathbf{w}_*, \quad (2.52)$$

where with  $\otimes$  we denote the Kronecher product. The model in (2.51) is a slight modification of the linear system, which was discussed in Section 2.2.3. The difference is that in the current model at each time instance, each node obtains  $R$  measurements instead of one. Nevertheless, the adopted methodology through which a solution is obtained, follows a similar philosophy with the one presented in section 2.2.3.

First of all let us consider the case where each node estimates  $\mathbf{w}_*$  using only local measurements. The following cost function is defined:

$$J(\mathbf{w}) = \|\mathbf{d}_{k,n} - \mathbf{U}_{k,n} \mathbf{w}\|^2. \quad (2.53)$$

Under similar assumptions regarding the statistics of  $\mathbf{d}_{k,n}$  and  $\mathbf{U}_{k,n}$  as the ones presented in 2.2.3, and following similar steps, the minimization of (2.53) gives us the following equations:

$$\mathbf{p}_{\mathbf{d}_k \mathbf{U}_k} = \mathbf{R}_{\mathbf{U}_k} \mathbf{w}_* \Leftrightarrow \mathbf{w}_* = \mathbf{R}_{\mathbf{U}_k}^{-1} \mathbf{p}_{\mathbf{d}_k \mathbf{U}_k}, \quad (2.54)$$

where  $\mathbf{p}_{\mathbf{d}_k \mathbf{U}_k} = \mathbb{E} \mathbf{U}_{k,n}^T \mathbf{d}_{k,n}$  and  $\mathbf{R}_{\mathbf{U}_k} = \mathbb{E} \mathbf{U}_{k,n}^T \mathbf{U}_{k,n}$ .

The cost function (2.53) can be minimized at each node via diffusion based optimization, *e.g.*, [122]; that is, the nodes exchange information (estimates and statistics) and fuse them with respect to the previously discussed principles. In more practical scenarios, where the statistics are not a-priori known and/or undergo changes, adaptive algorithms can be employed so as to estimate the desired vector.

**Remark 6.** *Another route, followed in [26, 30], is to resort to diffusion based distributed detection techniques. In distributed detection, e.g., [140, 141], each node, which in the certain paradigm is a SU, decides whether a PU is present or not based on a hypothesis testing. In principle, each SU performs computations and if a computed measure is above a certain threshold, which is determined by the noise statistics, then it decides that the frequency bands are occupied. On the contrary, if the computed value is smaller than this threshold*

*then the bands are considered unoccupied. An LMS-based scheme is employed in [26] and a set-theoretic one in [30]. Finally, a different viewpoint, proposed in the study [7], is to adopt a parsimonious model for the power spectrum density and resort to sparsity promoting techniques for its estimation.*

# Chapter 3

## Adaptive Filtering

*In this chapter, the task of adaptive filtering will be discussed. More specifically, the major characteristics of the LMS and the RLS algorithms will be presented. Furthermore, their distributed/diffusion counterparts will be discussed. Finally, the set-theoretic based APSM will be described together with its theoretical properties.*

### 3.1 The LMS algorithm

Our kick off point is the linear system, which was discussed in chapter 2. Recall that the goal is the estimation of a vector  $\mathbf{w}_* \in \mathbb{R}^m$ , exploiting measurements obeying the following model:

$$d_n = \mathbf{u}_n^T \mathbf{w}_* + v_n, \quad n \in \mathbb{Z}, \quad (3.1)$$

where  $d_n \in \mathbb{R}$ ,  $\mathbf{u}_n \in \mathbb{R}^m$  and  $v_n \in \mathbb{R}$  is the noise process. Notice that the node subscript is omitted here, since we deal with a non-distributed adaptive filtering problem. As we have already discussed in chapter 2, the estimation of  $\mathbf{w}_*$  is achieved via the minimization of the following cost function:

$$J(\mathbf{w}) = \mathbb{E}[d_n - \mathbf{u}_n^T \mathbf{w}]^2. \quad (3.2)$$

Solving (3.2) we obtain  $\mathbf{w}_*$  as

$$\mathbf{w}_* = \mathbf{R}^{-1} \mathbf{p}, \quad (3.3)$$

where  $\mathbf{R}$  and  $\mathbf{p}$  are the autocorellation matrix and the crosscorellation vector, respectively. A possible way for the computation of  $\mathbf{w}_*$ , instead of solving (3.3) directly, is to resort to iterative techniques, such as the steepest–descent method. The resulting recursion is:

$$\begin{aligned}\mathbf{w}_{n+1} &= \mathbf{w}_n - \mu \nabla J(\mathbf{w}_n) \\ &= \mathbf{w}_n + \mu (\mathbf{p} - \mathbf{R}\mathbf{w}_n),\end{aligned}\tag{3.4}$$

where  $\mu$  is the constant step–size and  $\mathbf{w}_n$  is the estimate of  $\mathbf{w}_*$  at the  $n$ -th iteration. For a properly chosen step–size, it can be proved that  $\mathbf{w}_n \rightarrow \mathbf{w}_*$ ,  $n \rightarrow \infty$ , *e.g.*, [120].

In practice, the statistics  $\mathbf{R}$  and  $\mathbf{p}$  are rarely available and the gradient, *i.e.*,  $\nabla J(\mathbf{w}_n)$ , which requires knowledge of these quantities, cannot be computed exactly. A possible “escape” route for such cases is the so–called *stochastic gradient* rationale. This belongs to the more general family of *stochastic approximation* [116]; expected values are replaced by their instantaneous measurements, *e.g.*, [120]. The celebrated LMS algorithm belongs to the family of the stochastic gradient schemes, where the following approximations are employed:  $\mathbf{R}_n \approx \mathbf{u}_n \mathbf{u}_n^T$  and  $\mathbf{p}_n \approx d_n \mathbf{u}_n^T$ . In this case, the gradient of the cost function is approximated by:

$$\nabla J(\mathbf{w}_n) \approx \mathbf{u}_n \mathbf{u}_n^T \mathbf{w}_n - d_n \mathbf{u}_n = (\mathbf{u}_n^T \mathbf{w}_n - d_n) \mathbf{u}_n.\tag{3.5}$$

and the resulting recursion becomes:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu (d_n - \mathbf{u}_n^T \mathbf{w}_n) \mathbf{u}_n.\tag{3.6}$$

The LMS scheme of (3.6) is one of the most commonly used adaptive filtering algorithms, due to its simplicity and its robustness to deal with different learning scenarios. The LMS algorithm was originally proposed by Widrow and Hoff in [143]. Finally, it is worth pointing out that the theoretical properties of the LMS have been extensively studied in the literature, *e.g.*, [52,63,120], albeit under a number of assumptions. Although LMS is structurally simple, it is a nonlinear time varying algorithm, which makes its analysis a formidable task.

Now, let us describe how the classical LMS can be generalized so as to address distributed adaptive filtering problems. Our starting point will be the A–C and the C–A diffusion schemes presented in Chapter 2. Recall that the recursion for the A–C diffusion algorithm

is given by

$$\boldsymbol{\psi}_{k,n} = \mathbf{w}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (\mathbf{p}_l - \mathbf{R}_l \mathbf{w}_{k,n}) \quad (3.7)$$

$$\mathbf{w}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{k,l} \boldsymbol{\psi}_{l,n}, \quad (3.8)$$

whereas the C–A one is given by:

$$\boldsymbol{\psi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{w}_{l,n} \quad (3.9)$$

$$\mathbf{w}_{k,n+1} = \boldsymbol{\psi}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (\mathbf{p}_l - \mathbf{R}_l \mathbf{w}_{k,n}). \quad (3.10)$$

Using the instantaneous measurement approximation, *i.e.*,  $\mathbf{p}_l - \mathbf{R}_l \mathbf{w}_{k,n} \approx (d_{k,n} - \mathbf{u}_{k,n}^T \mathbf{w}_{k,n}) \mathbf{u}_{k,n}$ , we obtain the A–C diffusion LMS:

$$\boldsymbol{\psi}_{k,n} = \mathbf{w}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (d_{l,n} - \mathbf{u}_{l,n}^T \mathbf{w}_{k,n}) \mathbf{u}_{l,n} \quad (3.11)$$

$$\mathbf{w}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{k,l} \boldsymbol{\psi}_{l,n}, \quad (3.12)$$

and the C–A diffusion LMS:

$$\boldsymbol{\psi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{w}_{l,n} \quad (3.13)$$

$$\mathbf{w}_{k,n+1} = \boldsymbol{\psi}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (d_{l,n} - \mathbf{u}_{l,n}^T \mathbf{w}_{k,n}) \mathbf{u}_{l,n}. \quad (3.14)$$

The A–C diffusion LMS was first proposed in [29] and the C–A one in [94].

As we have already discussed, under proper assumptions regarding the step–size, the gradient descent iterative scheme, which uses the statistical quantities  $\mathbf{p}_k$ ,  $\mathbf{R}_k$ , converges asymptotically to  $\mathbf{w}_*$ . In the LMS and in its distributed counterparts, where an approximation of the statistics is used, the algorithm converges in a different sense. More specifically, the algorithm converges *in the mean*; that is, the expected value of the estimates tends asymptotically to the unknown vector  $\mathbf{w}_*$ . In the sequel, the convergence behavior of the LMS will be summarized along its main points.

### 3.1.1 Convergence of the Diffusion based LMS

Let us first write in a more compact form the equations of the A–C and the C–A diffusion LMS as follows:

$$\begin{cases} \boldsymbol{\phi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{1,k,l} \mathbf{w}_{l,n}, \\ \boldsymbol{\psi}_{k,n+1} = \boldsymbol{\phi}_{k,n} + \mu_k \sum_{l \in \mathcal{N}_k} c_{k,l} (d_{l,n} - \mathbf{u}_{l,n}^T \boldsymbol{\phi}_{k,n}) \mathbf{u}_{l,n}, \\ \mathbf{w}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{2,k,l} \boldsymbol{\psi}_{l,n+1}, \end{cases} \quad (3.15)$$

where the properties satisfied by the coefficients  $a_{1,k,l}$ ,  $c_{k,l}$  and  $a_{2,k,l}$  are given in Chapter 2. Note that the A–C LMS occurs if we let  $\mathbf{A}_1 = \mathbf{I}_m$  in (4.5), where  $\mathbf{A}_1$  is an  $N \times N$  matrix, with entries  $a_{1,k,l}$ , whereas the C–A LMS results if we let  $\mathbf{A}_2 = \mathbf{I}_m$ , where  $\mathbf{A}_2$  is an  $N \times N$  matrix consisting of the weights  $a_{2,k,l}$ . Let us define the following vectors:

$$\tilde{\mathbf{w}}_{k,n} = \mathbf{w}_* - \mathbf{w}_{k,n}, \quad \tilde{\boldsymbol{\psi}}_{k,n} = \mathbf{w}_* - \boldsymbol{\psi}_{k,n}, \quad \tilde{\boldsymbol{\phi}}_{k,n} = \mathbf{w}_* - \boldsymbol{\phi}_{k,n}.$$

Moreover, the following networkwise vectors will turn to be useful:

$$\underline{\tilde{\mathbf{w}}}_n = \begin{bmatrix} \mathbf{w}_* - \mathbf{w}_{1,n} \\ \vdots \\ \mathbf{w}_* - \mathbf{w}_{N,n} \end{bmatrix}, \quad \underline{\tilde{\boldsymbol{\psi}}}_n = \begin{bmatrix} \mathbf{w}_* - \boldsymbol{\psi}_{1,n} \\ \vdots \\ \mathbf{w}_* - \boldsymbol{\psi}_{N,n} \end{bmatrix}, \quad \underline{\tilde{\boldsymbol{\phi}}}_n = \begin{bmatrix} \mathbf{w}_* - \boldsymbol{\phi}_{1,n} \\ \vdots \\ \mathbf{w}_* - \boldsymbol{\phi}_{N,n} \end{bmatrix}.$$

as well as:

$$\mathbf{A}_1 = \mathbf{A}_1 \otimes \mathbf{I}_m, \quad \mathbf{A}_2 = \mathbf{A}_2 \otimes \mathbf{I}_m, \quad \mathbf{P} = \mathbf{C} \otimes \mathbf{I}_m,$$

where the matrix  $\mathbf{C}$  is defined in a similar way as  $\mathbf{A}_1$ ,  $\mathbf{A}_2$ , with entries the weights  $c_{k,l}$ . Let now

$$\mathbf{D}_n = \text{diag} \left\{ \sum_{l \in \mathcal{N}_1} c_{1,l} \mathbf{u}_{l,n} \mathbf{u}_{l,n}^T, \dots, \sum_{l \in \mathcal{N}_N} c_{N,l} \mathbf{u}_{l,n} \mathbf{u}_{l,n}^T \right\},$$

and

$$\mathbf{g}_n = \mathbf{P}^T [\mathbf{u}_{1,n}^T v_{1,n}, \dots, \mathbf{u}_{N,n}^T v_{N,n}]^T.$$

Then, (4.5) can be written for the whole network as follows

$$\begin{cases} \tilde{\underline{\phi}}_n = \mathcal{A}_1 \tilde{\underline{w}}_n \\ \tilde{\underline{\psi}}_{n+1} = \tilde{\underline{\phi}}_n - \mathbf{M} [\mathbf{D}_n \tilde{\underline{\phi}}_n + \mathbf{g}_n] \\ \tilde{\underline{w}}_{n+1} = \mathcal{A}_2 \tilde{\underline{\psi}}_{n+1}, \end{cases} \quad (3.16)$$

where the  $Nm \times Nm$  matrix  $\mathbf{M}$  equals to

$$\mathbf{M} = \text{diag}\{\mu_1 \mathbf{I}_m, \dots, \mu_n \mathbf{I}_m\}.$$

The set of equations in (3.16) can be written compactly as follows:

$$\tilde{\underline{w}}_{n+1} = \mathcal{A}_2 [\mathbf{I}_m - \mathbf{M} \mathbf{D}_n] \mathcal{A}_1 \tilde{\underline{w}}_n - \mathcal{A}_2 \mathbf{M} \mathbf{g}_n. \quad (3.17)$$

Finally, we define

$$\begin{aligned} \mathcal{D} &= \mathbb{E}[\mathbf{D}_n] = \text{diag} \left\{ \sum_{l \in \mathcal{N}_1} c_{1,l} \mathbf{R}_l, \dots, \sum_{l \in \mathcal{N}_N} c_{N,l} \mathbf{R}_l \right\} \\ \mathcal{G} &= \mathbf{P}^T \text{diag} \{ \sigma_1^2 \mathbf{R}_1, \dots, \sigma_N^2 \mathbf{R}_N \} \mathbf{P}. \end{aligned}$$

### Assumptions 1.

1. *The input vector and the noise process have zero means. Moreover, they are independent.*
2. *The input vectors are temporarily and spatially independent. In system identification applications, the input vectors are assumed to have a sliding window form, i.e.,  $\mathbf{u}_{k,n} := [u_{k,n}, u_{k,n-1}, \dots, u_{k,n-m+1}]$ . Obviously, the temporal independence does not hold due to the sliding window form, since any two successive vectors will have  $m - 1$  common coefficients. Nevertheless, this assumption, which is known as independence assumption, is often adopted in the study of LMS-based algorithms, e.g., [98, 100, 120], because it simplifies significantly the analysis.*

3. Assume that the combination matrices  $\mathcal{A}_1, \mathcal{A}_2$  satisfy  $\mathbf{1}_{Nm}^T \mathcal{A}_i = \mathbf{1}_{Nm}^T, i = 1, 2$ .
4. Regarding the step-sizes, we assume that:

$$0 < \mu_k < \frac{2}{\tau \left( \sum_{l \in \mathcal{N}_k} c_{k,l} \mathbf{R}_l \right)} \quad (3.18)$$

**Theorem 3.** Consider that the previously mentioned assumptions hold true. Then, the mean value of the estimates at the nodes of the network converge asymptotically to the unknown solution, that is:

$$\lim_{n \rightarrow \infty} \mathbb{E}[\mathbf{w}_{k,n}] = \mathbf{w}_*, \quad \forall k \in \mathcal{N}.$$

*Proof.* First of all, it is easy to obtain that under assumption 1.2, the input vector,  $\mathbf{u}_{k,n}$  is independent of the past estimates,  $\mathbf{w}_{l,j}, l \in \mathcal{N}, j = 1, \dots, n-1$ . According to this, if we employ the expectation operator in (3.17), and take into consideration that  $\mathbb{E}[\mathbf{g}_n] = \mathbf{0}_{Nm}$ , because the noise is independent of the input and their expected value equals to zero, we have

$$\mathbb{E}[\tilde{\mathbf{w}}_{n+1}] = \mathcal{A}_2 [\mathbf{I}_m - \mathcal{M}\mathcal{D}_n] \mathcal{A}_1 \mathbb{E}[\tilde{\mathbf{w}}_n] = (\mathcal{A}_2 (\mathbf{I}_m - \mathcal{M}\mathcal{D}_n) \mathcal{A}_1)^n \mathbb{E}[\tilde{\mathbf{w}}_0]. \quad (3.19)$$

It has been proved, *e.g.*, [122], that if the step-sizes  $\mu_k$  satisfy assumption 1.2 and the matrices  $\mathcal{A}_1, \mathcal{A}_2$  satisfy assumption 1.3, then

$$\lim_{n \rightarrow \infty} (\mathcal{A}_2 [\mathbf{I}_m - \mathcal{M}\mathcal{D}_n] \mathcal{A}_1)^n = O_{Nm},$$

where  $O_{Nm}$  denotes the zero matrix of dimension  $Nm \times Nm$ . Combining the last equation with (3.19), it can be readily obtained that

$$\lim_{n \rightarrow \infty} \mathbb{E}[\tilde{\mathbf{w}}_{n+1}] = \mathbf{0}_{Nm} \Leftrightarrow \lim_{n \rightarrow \infty} \mathbb{E}[\mathbf{w}_{k,n}] = \mathbf{w}_*.$$

□

**Remark 7.** The complexity of the LMS-based algorithms is linear with respect to the dimensionality of the unknown vector. More specifically, in the non-distributed LMS, each iteration requires  $2m$  additions and  $2m + 1$  multiplications. The low-complexity of this scheme is one

of its major advantages.

**Remark 8.** Another important advantage of the LMS is its powerful tracking ability, i.e., the LMS is able to track possible changes, which take place on the unknown parameter  $\mathbf{w}_*$ . This is a direct consequence of the fact that the LMS exploits only the most recent data; this is due to the choice of  $\mu$ , which is either constant or it does not allow to be vanishingly small as the iterations progress, [64, 120].

**Remark 9.** As it has been already stated in the previous chapters, another class of adaptive schemes for distributed learning is that of the consensus based algorithms. The recursion of the consensus-based LMS, see for example [80, 105]<sup>1</sup>, is the following:

$$\mathbf{w}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{w}_{l,n} + \mu_k (d_{k,n} - \mathbf{u}_{k,n}^T \mathbf{w}_{k,n}) \mathbf{u}_{k,n}. \quad (3.20)$$

It has been proved, [122, 137] that the mean stability of this algorithm is sensitive to the choice of the combination coefficients, in contrast to the diffusion-based schemes. Another viewpoint of the LMS algorithm following the consensus rationale has been proposed in [98, 123]. The treatment there is different and the philosophy adopted there is that of the Alternating Direction Method of Multipliers, [10]. The resulting scheme consists of updates associated not only with the estimates of the unknown parameter vector but, also, of updates for a set of Lagrange multipliers.

## 3.2 Recursive Least Squares Algorithm

Another celebrated algorithm, which is commonly used for adaptive filtering is the RLS one. The RLS algorithm is a recursive implementation of the Least Squares (LS) method. Given the measurement pairs  $(d_n, \mathbf{u}_n)$ ,  $n = 0, \dots, K$ , the classical LS problem seeks an  $m \times 1$  vector that minimizes the following cost function, e.g., [120]:

$$J_{LS}(\mathbf{w}) = \sum_{n=0}^K (d_n - \mathbf{w}^T \mathbf{u}_n)^2, \quad (3.21)$$

---

<sup>1</sup>In these studies, a diminishing step size was assumed so as to guarantee convergence to  $\mathbf{w}_*$ . Obviously, in this case, the algorithms do not have the ability to track changes. Following [122], a constant step-size has been considered here, to cope with adaptivity.

or in matrix-vector form:

$$J_{LS}(\mathbf{w}) = \|\mathbf{d}_K - \mathbf{U}_K \mathbf{w}\|^2, \quad (3.22)$$

with  $\mathbf{d}_K = [d_0, d_1, \dots, d_K]^T$  and  $\mathbf{U}_K = [\mathbf{u}_0^T, \mathbf{u}_1^T, \dots, \mathbf{u}_K^T]^T$ . The vector that minimizes (3.21) is given by (see for example [64, 120]):

$$\mathbf{w}_{LS} = (\mathbf{U}_K^T \mathbf{U}_K)^{-1} \mathbf{U}_K^T \mathbf{d}_K \quad (3.23)$$

Employing the conventional LS in an online/adaptive scenario is prohibited since: a) the number of involved measurements increases as the time advances and a large amount of data has to be stored, b) solving (3.21) requires the inversion of an  $m \times m$  matrix at each time instance and c) all the measurements are equally weighted. This does not pose problems in a stationary environment when the system has to be solved once. However, if  $\mathbf{w}_*$  is time varying, then measurements corresponding to the remote past have a strong influence on the more recent time instants. This is taken into account by the exponentially weighted LS cost function defined as

The cost function of the exponentially weighted (non-distributed) RLS is given by:

$$J_{LS}(\mathbf{w}) = \sum_{n=0}^K \zeta^{K-n} (d_n - \mathbf{w}^T \mathbf{u}_n)^2. \quad (3.24)$$

The factor  $\zeta \in (0, 1]$ , which is widely known as *forgetting factor*, reduces the contribution of the past values, for  $\zeta < 1$ . In the case where  $\zeta = 1$ , all the measurements are equally weighted; such a choice is suitable in stationary environments. From the first order optimality condition, the minimizer of (3.24) satisfies:

$$\sum_{n=0}^K \zeta^{K-n} \mathbf{u}_n \mathbf{u}_n^T \mathbf{w}_{LS} = \sum_{n=0}^K \zeta^{K-n} \mathbf{u}_n d_n. \quad (3.25)$$

Obviously, the computation of  $\mathbf{w}_{LS}$  requires the solution of an  $m \times m$  system of equations, which is prohibited when operating in an online fashion, since this has to be repeated every time instant. The RLS algorithm overcomes this obstacle by solving the LS task in a time

iterative manner. Observe that:

$$\sum_{n=0}^{K+1} \zeta^{K+1-n} \mathbf{u}_n \mathbf{u}_n^T = \mathbf{u}_{K+1} \mathbf{u}_{K+1}^T + \zeta \sum_{n=0}^K \zeta^{K-n} \mathbf{u}_n \mathbf{u}_n^T = \mathbf{u}_{K+1} \mathbf{u}_{K+1}^T + \zeta \Phi_K, \quad (3.26)$$

where  $\Phi_K = \sum_{n=0}^K \zeta^{K-n} \mathbf{u}_n \mathbf{u}_n^T$ . Similarly, we have that  $\sum_{n=0}^{K+1} \zeta^{K+1-n} \mathbf{u}_n d_n = \mathbf{u}_{K+1} d_{K+1} + \zeta \mathbf{p}_K$ , with  $\mathbf{p}_K = \sum_{n=0}^K \zeta^{K-n} \mathbf{u}_n d_n$ . According to these relations, the optimum vector  $\mathbf{w}_{LS}$  is given by

$$\mathbf{w}_{LS} = (\mathbf{u}_{K+1} \mathbf{u}_{K+1}^T + \zeta \Phi_K)^{-1} (\mathbf{u}_{K+1} d_{K+1} + \zeta \mathbf{p}_K). \quad (3.27)$$

Next, the celebrated Matrix Inversion Lemma will be employed to facilitate the computation of the inverse in the right hand side in (3.27).

**Lemma 1.** *Consider the  $m \times m$  positive definite matrices  $\mathbf{A}, \mathbf{B}$ . Moreover, assume that  $\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C} \mathbf{D}^{-1} \mathbf{C}^T$ , where the  $\mathbf{D}$   $m \times r$  is positive definite and  $\mathbf{C}$  is an  $m \times r$  matrix. Then it holds that*

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B} \mathbf{C} (\mathbf{D} + \mathbf{C}^T \mathbf{B} \mathbf{C})^{-1} \mathbf{C}^T \mathbf{B}. \quad (3.28)$$

If we apply the matrix inversion Lemma with  $\mathbf{A} = \Phi_{n+1}$ ,  $\mathbf{B}^{-1} = \zeta \mathbf{p}_n$ ,  $\mathbf{C} = \mathbf{u}_n$  and  $\mathbf{D} = 1$ , we get that

$$\begin{aligned} \mathcal{P}_{n+1} &:= \Phi_{n+1}^{-1} \\ &= \zeta^{-1} \mathcal{P}_n - \frac{\zeta^{-2} \mathcal{P}_n \mathbf{u}_n \mathbf{u}_n^T \mathcal{P}_n}{1 + \zeta^{-1} \mathbf{u}_n^T \mathcal{P}_n \mathbf{u}_n}. \end{aligned} \quad (3.29)$$

Summarizing the previous relations we define:

$$\mathbf{k}_{n+1} = \frac{\zeta^{-1} \mathcal{P}_n \mathbf{u}_n}{1 + \zeta^{-1} \mathbf{u}_n^T \mathcal{P}_n \mathbf{u}_n}, \quad (3.30)$$

and (3.29) can be rewritten as:

$$\mathcal{P}_{n+1} = \zeta^{-1} \mathcal{P}_n - \zeta^{-1} \mathbf{k}_{n+1} \mathbf{u}_n^T \mathcal{P}_n. \quad (3.31)$$

The steps of the RLS algorithm are summarized as follows, [120]:

1. Initializations:  $\mathbf{w}_0 = \mathbf{0}_m$ ,  $\mathcal{P}_0 = \delta \mathbf{I}_m$ , for a small  $\delta > 0$ .

2. Computation of  $\mathbf{k}_{n+1}$  via (3.30).
3. Computation of the error residual  $e_n := d_n - \mathbf{w}_n^T \mathbf{u}_n$ .
4. Estimate update  $\mathbf{w}_{n+1} = \mathbf{w}_n + \mathbf{k}_{n+1} e_n$ .
5. Update  $\mathcal{P}_{n+1}$  via (3.31).

**Remark 10.** *The complexity of the RLS algorithm is higher compared to the complexity of the LMS. More specifically, since matrix operations are required, the complexity of the RLS is of order  $O(m^2)$ . In the literature, variations of the RLS have been proposed, which reduce the number of operations, e.g., [47, 60, 101, 120].*

**Remark 11.** *As we have already mentioned before, the forgetting factor  $\zeta$  helps to “forget” past data. Hence, if  $\zeta$  equals to 1 and  $\mathbf{w}_*$  is time varying, then the performance of the algorithm is degraded, since values corresponding to previous instances of  $\mathbf{w}_*$  will be taken into consideration. On the contrary, if  $\zeta < 1$ , then values from the remote past will be weighted with weights close to zero and they will not contribute to the cost function. In practice, e.g., [120], if one chooses a  $\zeta < 1$  then the algorithm obtains a tracking potential.*

### 3.2.1 Diffusion Recursive Least Squares

In this section, the RLS algorithm will be brought in a form that complies with the diffusion rationale. The diffusion RLS scheme, presented in [27], consists of two steps: a) an incremental one, in which the nodes exchange their measurements and perform successive least squares operations, and b) a diffusion step, where the nodes exchange their obtained estimates, which were previously computed, and then combine them via an adopted combination scheme. The steps of the diffusion based RLS are given in the sequel:

1. Initializations:  $\mathbf{w}_{k,0} = \mathbf{0}_m$ ,  $\mathcal{P}_{k,0} = \delta \mathbf{I}_m$ , for a small  $\delta > 0$ .
2. Incremental Steps:
  - $\psi_{k,n} = \mathbf{w}_{k,n}$
  - $\mathcal{P}_{k,n} = \zeta^{-1} \mathcal{P}_{k,n-1}$

- For every node of the neighborhood  $l \in \mathcal{N}_k$  repeat:

$$\boldsymbol{\psi}_{k,n+1} = \boldsymbol{\psi}_{k,n} + \frac{c_{k,l} \mathcal{P}_{k,n} \mathbf{u}_{l,n}}{1 + c_{k,l} \mathbf{u}_{l,n}^T \mathcal{P}_{k,n} \mathbf{u}_{l,n}} (d_{l,n} - \mathbf{u}_{l,n}^T \boldsymbol{\psi}_{k,n}) \quad (3.32)$$

$$\mathcal{P}_{k,n+1} = \mathcal{P}_{k,n} - \frac{c_{k,l} \mathcal{P}_{k,n} \mathbf{u}_{l,n} \mathbf{u}_{l,n}^T \mathcal{P}_{k,n}}{1 + c_{k,l} \mathbf{u}_{l,n}^T \mathcal{P}_{k,n} \mathbf{u}_{l,n}} \quad (3.33)$$

- end

### 3. Diffusion Step:

- $\mathbf{w}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{k,l} \boldsymbol{\psi}_{k,n+1}$ .

Notice that in the diffusion RLS, the nodes do not exchange the matrices  $\mathcal{P}_{l,n}$ , since this would require transmission of  $m^2$  coefficients and would be a burden in terms of communication resources. Instead, the nodes exchange  $d_{l,n}$ ,  $\mathbf{u}_{l,n}$  and the vector  $\boldsymbol{\psi}_{l,n}$ .

It can be verified that at the end of the incremental step, the following equations are satisfied:

$$\mathcal{P}_{k,n+1}^{-1} = \zeta \mathcal{P}_{k,n}^{-1} + \sum_{l \in \mathcal{N}_k} c_{k,l} \mathbf{u}_{l,n} \mathbf{u}_{l,n}^T \quad (3.34)$$

$$\mathbf{q}_{k,n+1} := \mathcal{P}_{k,n+1}^{-1} \boldsymbol{\psi}_{k,n+1} = \zeta \mathcal{P}_{k,n}^{-1} \mathbf{w}_{k,n} + \sum_{l \in \mathcal{N}_k} c_{k,l} \mathbf{u}_{l,n} d_{l,n}. \quad (3.35)$$

Exploiting these two relations, the diffusion RLS can be rewritten:

$$\mathbf{w}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \boldsymbol{\psi}_{l,n} \quad (3.36)$$

$$\mathcal{P}_{k,n+1}^{-1} = \zeta \mathcal{P}_{k,n}^{-1} + \sum_{l \in \mathcal{N}_k} c_{k,l} \mathbf{u}_{l,n} \mathbf{u}_{l,n}^T \quad (3.37)$$

$$\mathbf{q}_{k,n+1} = \zeta \mathcal{P}_{k,n}^{-1} \mathbf{w}_{k,n} + \sum_{l \in \mathcal{N}_k} c_{k,l} \mathbf{u}_{l,n} d_{l,n}. \quad (3.38)$$

$$\boldsymbol{\psi}_{k,n+1} = \mathcal{P}_{k,n+1} \mathbf{q}_{k,n+1}. \quad (3.39)$$

The theoretical properties of the diffusion based RLS are discussed in [27].

**Remark 12.** *RLS based algorithms following the consensus approach have been presented, together with their theoretical analyses, in [97, 98]. It is worth pointing out that the diffusion*

*RLS enjoys a faster convergence speed compared to the consensus based one, as it has been experimentally verified in [27].*

### 3.3 The Adaptive Projected Subgradient Method

In this section, we will discuss the “algorithmic tool”, which comprises the method on which the novel results in this dissertation are built upon, *i.e.*, the Adaptive Projected Subgradient Method.

The main idea of the previously described adaptive algorithms is to employ a proper cost function and minimize it so as to compute estimates of the unknown parameter vector. These cost functions quantify the deviation/dissimilarity between the input and the output, so the point that minimizes them, minimizes this dissimilarity as well. Nevertheless, in many applications, adopting a cost function is not a trivial task. For example, it is by now well established, that the performance of the LMS algorithm is dictated by the statistics of the input, in the sense that if the autocorrelation matrix has a large eigenvalue spread, then the performance of the LMS is degraded, *e.g.*, [120]. Another typical example (see [134]) is that the Least Squares cost function, which is employed in the RLS algorithm, is very sensitive to outliers, *i.e.*, noise values with extremely large amplitudes. Finally, it is worth pointing out that frequently the choice of the cost function is mainly dictated by the mathematical properties that underlie it and not by the specific nature of the problem, which is dealt. More specifically, since the problem is solved via the minimization of the respective functions, issues such as differentiability often dictate the choice of the loss function.

The APSM algorithm, which was introduced in [148] and generalized in [125, 128], follows a different route. Instead of a single loss function, that is minimized over the whole set of measurements, each time instant is treated separately. Given the measurement pair  $(d_n, \mathbf{u}_n) \in \mathbb{R} \times \mathbb{R}^m$ , at time  $n$ , the designer quantifies his/her perception of loss, with respect to the received measurement pair, by a “local” loss function, which can also be considered to be time varying. This “local” loss defines a region (set of points), which is also known as a *property set*, where the estimate of the unknown vector would be desirable to lie, in order to be considered in *agreement* with the current measurements (*i.e.*, low error, smaller than a predefined threshold). Furthermore, in several applications additional knowledge concerning

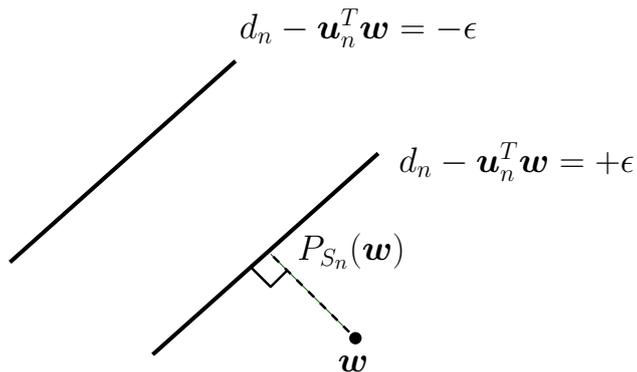


Figure 3.1: Illustration of a hyperslab and the projection of a point onto it.

the unknown parameter vector may be available *a-priori*. This a-priori knowledge can be embedded in the algorithm in the form of constraints. Each constraint defines a corresponding set of points in the solutions space. The goal is to find a point in the intersection of all these property sets as well as the respective constraint sets. In order to achieve this goal, a sequence of projections onto all the previously mentioned sets will be mobilized; this is in line with the classical concept of the method of Projections onto Convex Sets (POCS), [142]. It is worth pointing out that the only requirements for the property and the constraint sets, is convexity and not differentiability of any loss functions, which was the case in the LMS and the RLS algorithms.

To learn by example, before we describe the algorithm, let us provide a typical example of a set, which can be employed for the estimation of  $\mathbf{w}_*$ . Recall the linear system (3.1) and assume that the noise process is bounded by a constant  $\rho$ , *i.e.*,  $|v_n| \leq \rho$ ,  $n \in \mathbb{Z}$ . At each time instance, the following set of points is defined

$$S_n := \{\mathbf{w} \in \mathbb{R}^m : |d_n - \mathbf{u}_n^T \mathbf{w}| \leq \epsilon\}, \quad (3.40)$$

where  $\epsilon$  is a user-defined threshold that satisfies  $\epsilon \geq \rho$ . This set is known as a *hyperslab* and it is illustrated in Fig 3.1. Since the noise is bounded and  $\epsilon \geq \rho$  it can be readily verified that the unknown vector  $\mathbf{w}_* \in S_n$ ,  $\forall n \in \mathbb{Z}$ . This justifies the notion behind the APSM algorithm, where one seeks for a point, that lies in the intersection of the property sets, since all of them contain the unknown solution. If the noise is not bounded, then an appropriate choice of  $\epsilon$ , can guarantee that with high probability [150].

### 3.3.1 Basic Concepts of Convex Analysis and the POCS Algorithm.

At the heart of the APSM algorithm lies the idea of projecting onto the property sets in order to find a point, which belongs in their intersection. From this point of view, the algorithm can be seen as a generalization of the classical Projections Onto Convex Sets (POCS) algorithm. The main difference is that in the POCS algorithm, the number of involved sets is *finite*, whereas the APSM deals with an *infinite* number of sets, one per time instance. In this subsection, we will describe the basic “tools”, which will be used for the derivation of the main algorithmic scheme. For the sake of completeness and in order to help the reader to grasp the main idea behind the APSM algorithm, the POCS algorithm will be discussed.

#### Basic Concepts Of Convex Analysis

A set  $\mathcal{C} \subseteq \mathbb{R}^m$ , for which it holds that  $\forall \mathbf{w}_1, \mathbf{w}_2 \in \mathcal{C}$  and  $\forall \hat{\alpha} \in [0, 1]$ ,  $\hat{\alpha}\mathbf{w}_1 + (1 - \hat{\alpha})\mathbf{w}_2 \in \mathcal{C}$ , is called convex. From a geometric point of view, this means that every line segment having as endpoints any  $\mathbf{w}_1, \mathbf{w}_2$  will lie in  $\mathcal{C}$ . Moreover, a function  $\Theta : \mathbb{R}^m \rightarrow \mathbb{R}$  is called convex if  $\forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^m$  and  $\forall \hat{\alpha} \in [0, 1]$  the inequality  $\Theta(\hat{\alpha}\mathbf{w}_1 + (1 - \hat{\alpha})\mathbf{w}_2) \leq \hat{\alpha}\Theta(\mathbf{w}_1) + (1 - \hat{\alpha})\Theta(\mathbf{w}_2)$  is satisfied. The 0-th level set of the convex function  $\Theta$  is defined as

$$\text{lev}_{\leq 0}\Theta = \{\mathbf{w} \in \mathbb{R}^m : \Theta(\mathbf{w}) \leq 0\}. \quad (3.41)$$

Finally, the subdifferential of  $\Theta$  at an arbitrary point, say  $\mathbf{w}_1$ , is defined as the set of all subgradients,  $\partial\Theta$  of  $\Theta$  at  $\mathbf{w}_1$  ([11, 65]), *i.e.*,

$$\partial\Theta(\mathbf{w}_1) := \{\mathbf{s} \in \mathbb{R}^m : \Theta(\mathbf{w}_1) + (\mathbf{w} - \mathbf{w}_1)^T \mathbf{s} \leq \Theta(\mathbf{w}), \forall \mathbf{w} \in \mathbb{R}^m\}. \quad (3.42)$$

The subgradient of a convex function is a generalization of the gradient, which is only defined if the function is differentiable. As a matter of fact, if a convex function is differentiable, its subdifferential at point  $\mathbf{w}_1$  is a singleton, with a single element, that is the gradient of the function at this point. It is well known that the gradient at a point  $\mathbf{w}_1$  has an elegant geometric interpretation. It defines the unique hyperplane, which is tangent at  $\mathbf{w}_1$  to the graph of  $\Theta(\mathbf{w})$ . Moreover, if  $\Theta(\mathbf{w})$  is convex, the graph of  $\Theta(\mathbf{w})$  lies in one of the sides of this

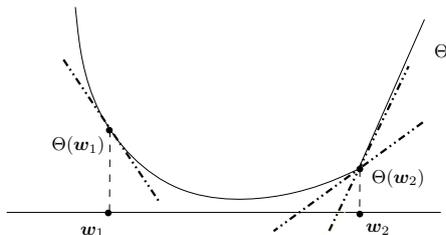


Figure 3.2: Illustration of a gradient and two subgradients of a convex function. Note that in  $\mathbf{w}_1$  the function is differentiable, so there exists a unique supporting hyperplane, which is tangent to the graph of the function, whereas in  $\mathbf{w}_2$  the function is not differentiable and there exist more than one hyperplanes, that support the graph of the function.

hyperplane. Similarly, each subgradient at a point of a convex function is associated with a hyperplane that leaves the graph of  $\Theta(\mathbf{w})$  on one of its side (supporting hyperplane). The only difference, now, is that there are more than one, possibly infinite, such hyperplanes. This is basically guaranteed by (3.42) and it is illustrated in Fig. 3.2.

The distance of an arbitrary point,  $\mathbf{w}$ , from a *closed* non-empty convex set,  $\mathcal{C}$ , is given by the distance function

$$d(\cdot, \mathcal{C}) : \mathbb{R}^m \rightarrow [0, +\infty)$$

$$: \mathbf{w} \mapsto \inf \{ \|\mathbf{w} - \hat{\mathbf{w}}\| : \hat{\mathbf{w}} \in \mathcal{C} \}.$$

This function is continuous, convex, nonnegative and is equal to zero for every point that lies in  $\mathcal{C}$  [65]. Moreover, the projection mapping,  $P_{\mathcal{C}}$  onto  $\mathcal{C}$ , is the mapping which takes a point  $\mathbf{w}$  to the uniquely existing point,  $P_{\mathcal{C}}(\mathbf{w}) \in \mathcal{C}$ , such that

$$\|\mathbf{w} - P_{\mathcal{C}}(\mathbf{w})\| = d(\mathbf{w}, \mathcal{C}).$$

It also holds that,  $P_{\mathcal{C}}(\mathbf{w}) = \mathbf{w}$ ,  $\forall \mathbf{w} \in \mathcal{C}$ . Making as our kick-off point the projection mapping, we define the relaxed projection mapping as follows:

$$T_{\mathcal{C}} = I + \mu(P_{\mathcal{C}} - I), \tag{3.43}$$

where  $\mu \in (0, 2)$  and  $I$  is the identity mapping. Obviously, if  $\mu = 1$  then  $T_{\mathcal{C}} = P_{\mathcal{C}}$ . The projection and the relaxed projection operators are illustrated in Fig. 3.3. In the sequel, we

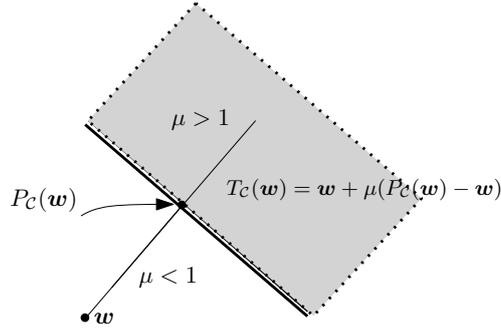


Figure 3.3: The projection and the relaxed projection operators.

present some convex sets alongside their projection operators, which will be used throughout this dissertation.

Given a scalar,  $d$ , and a nonzero vector  $\mathbf{u} \in \mathbb{R}^m$ , the definition of a closed halfspace (Fig. 3.4a) is given by

$$H^- := \{\mathbf{w} \in \mathbb{R}^m : \mathbf{w}^T \mathbf{u} \leq d\}, \quad (3.44)$$

and the projection operator associated with it is given by

$$P_{H^-}(\mathbf{w}) = \mathbf{w} - \frac{\min\{0, \mathbf{w}^T \mathbf{u} - d\}}{\|\mathbf{u}\|^2} \mathbf{u}, \quad \forall \mathbf{w} \in \mathbb{R}^m. \quad (3.45)$$

In a similar notion, we define the hyperplane

$$H := \{\mathbf{w} \in \mathbb{R}^m : \mathbf{w}^T \mathbf{u} = d\}, \quad (3.46)$$

and the resulting projection operator is

$$P_H(\mathbf{w}) := \mathbf{w} - \frac{\mathbf{w}^T \mathbf{u} - d}{\|\mathbf{u}\|^2} \mathbf{u}, \quad \forall \mathbf{w} \in \mathbb{R}^m. \quad (3.47)$$

Two more convex sets, which will be used in the theoretical analysis of the algorithms, are the closed and open balls with center  $\mathbf{c}$  and radius  $\delta$  (Fig. 3.4b). The definition of the closed ball set is:

$$B_{[\mathbf{c}, \delta]} := \{\mathbf{w} \in \mathbb{R}^m : \|\mathbf{w} - \mathbf{c}\| \leq \delta\}. \quad (3.48)$$

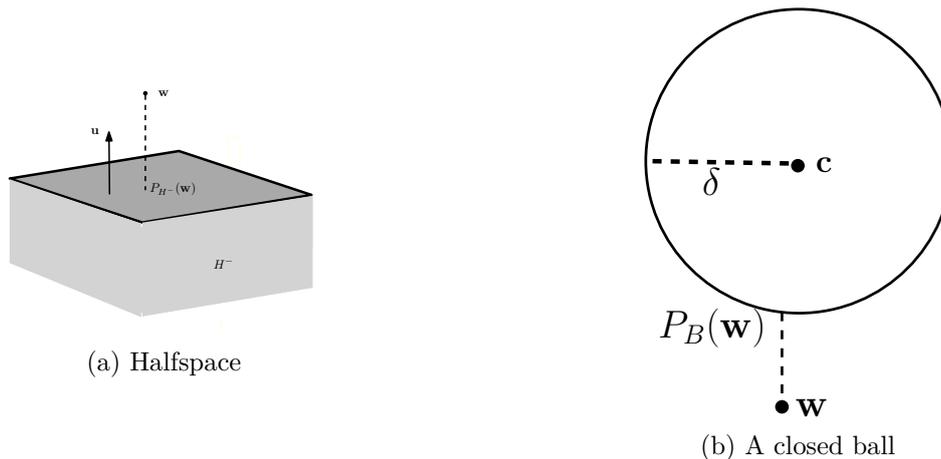


Figure 3.4: (a) The geometry of a halfspace. Its boundary is a hyperplane. (b) A closed ball  $B_{[c,\delta]}$ .

In a similar notion, the open ball is defined as

$$B_{(c,\delta)} := \{\mathbf{w} \in \mathbb{R}^m : \|\mathbf{w} - \mathbf{c}\| < \delta\}. \quad (3.49)$$

Finally, the relative interior of a nonempty set,  $\mathcal{C}$ , with respect to another one,  $\mathcal{S}$ , is defined as

$$\text{ri}_{\mathcal{S}}(\mathcal{C}) = \{\mathbf{w} \in \mathcal{C} : \exists \varepsilon_0 > 0 \text{ with } \emptyset \neq (B_{(\mathbf{w},\varepsilon_0)} \cap \mathcal{S}) \subset \mathcal{C}\}.$$

In a similar way, we define the interior of a set

$$\text{int}(\mathcal{C}) := \{\mathbf{w} \in \mathbb{R}^m : \exists \delta := \delta(\mathbf{w}) > 0 \text{ such that } B(\mathbf{w}, \delta) \subset \mathcal{C}\}. \quad (3.50)$$

The (relative) interior will be used in the convergence proof of the APSM-based algorithms.

### The POCS Algorithm

As it was previously discussed, the goal of the POCS algorithm is to find a point that lies in the intersection of a finite number of sets. Assume that we are given  $K$  closed convex sets, namely  $\mathcal{C}_i \subset \mathbb{R}^m$ ,  $i = 1, \dots, K$ . Moreover, assume that these sets share a non-empty intersection, *i.e.*,  $\check{\Omega} := \bigcap_{i=1}^K \mathcal{C}_i \neq \emptyset$ . We denote with  $T_{\mathcal{C}_i} = I + \mu_i(P_{\mathcal{C}_i} - I)$  the relaxed

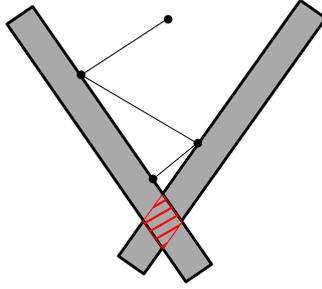


Figure 3.5: Sequential formulation of the POCS algorithm. Each iteration takes us closer to the intersection of the convex sets.

projection operator onto the set  $\mathcal{C}_i$  and  $\mu_i \in (0, 2)$ . Furthermore, we define the mapping:

$$T = T_{\mathcal{C}_K} T_{\mathcal{C}_{K-1}} \dots T_{\mathcal{C}_1}. \quad (3.51)$$

This mapping comprises  $K$  consecutive relaxed projections. The first takes place onto  $\mathcal{C}_1$ , the obtained vector is projected onto  $\mathcal{C}_2$  and so on.

**Theorem 4.** *Assume that the intersection of the  $\mathcal{C}_i$ -s is nonempty, i.e.,  $\check{\Omega} \neq \emptyset$ . Then for any initial point  $\mathbf{w}_0 \in \mathbb{R}^m$ , the sequence  $T^n(\mathbf{w}_0)$  converges to a point in  $\check{\Omega}$ , i.e.,*

$$\lim_{n \rightarrow \infty} T^n(\mathbf{w}_0) = \hat{\mathbf{w}}, \quad \hat{\mathbf{w}} \in \check{\Omega}. \quad (3.52)$$

*Proof.* The proof of the theorem can be found in [18]. From a geometrical point of view, the theorem is illustrated in Fig. 3.5 □

In the previous derivation of the POCS algorithm the projections took place sequentially. A parallel version of the POCS algorithm is given by the following iterative scheme

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu_n \left( \sum_{j=1}^K \omega_j P_{\mathcal{C}_j}(\mathbf{w}_n) - \mathbf{w}_n \right), \quad (3.53)$$

where  $\sum_{j=1}^K \omega_j = 1$  and  $\mu_n \in (0, 2\mathfrak{M}_n)$ , with

$$\mathfrak{M}_n = \frac{\sum_{j=1}^K \omega_j \|P_{\mathcal{C}_j}(\mathbf{w}_n) - \mathbf{w}_n\|^2}{\|\sum_{j=1}^K \omega_j P_{\mathcal{C}_j}(\mathbf{w}_n) - \mathbf{w}_n\|^2}. \quad (3.54)$$

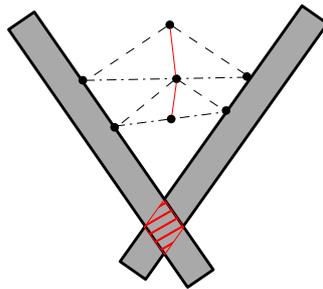


Figure 3.6: Parallel formulation of the POCS algorithm. Each iteration takes us closer to the intersection of the convex sets.

Notice that at each iteration step, the most recent estimate is projected onto all the sets and a convex combination of these projections is computed. The new estimate is obtained via (3.53) and in order to guarantee convergence the step-size  $\mu_n$  lies in the interval  $(0, 2\mathfrak{M}_n)$ . The geometric interpretation is shown in Fig. 3.6. Finally, the theoretical properties of the algorithm in (3.53) are studied in [109]. There, it was shown that the algorithm converges to a point that lies in  $\tilde{\Omega}$ .

#### Adaptive Projected Subgradient Method

So far, we have considered the case where the number of sets, onto which one seeks for a solution, is finite; the POCS algorithm was employed in this problem. In adaptive learning, an infinite number of measurements is (theoretically) available and through these, an infinite number of convex sets are constructed. The APSM algorithm, *e.g.*, [125, 128, 148] is a generalization of the POCS algorithm, in the sense that it deals with infinite convex sets. The goal remains the same; find a point that lies in the intersection of these infinite convex sets, with the possible exception of a finite number of outliers. Put in mathematical terms, we denote the received sets  $\mathcal{C}_n$ ,  $n \in \mathbb{Z}$  and the goal is to find a  $\hat{\mathbf{w}} \in \bigcap_{n \geq n_0} \mathcal{C}_n$  for a  $n_0 \geq 0$ .

For the derivation of the APSM, our starting point will be the parallel formulation of the POCS algorithm given by (3.53). The difference is that instead of projecting onto the  $K$  available convex sets, here we project onto the  $q$  most recent ones. For an arbitrarily initialized  $\mathbf{w}_0$ , the resulting recursion is the following

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu_n \left( \sum_{j \in \mathcal{J}_n} \omega_j P_{\mathcal{C}_j}(\mathbf{w}_n) - \mathbf{w}_n \right), \quad (3.55)$$

where  $\sum_{j \in \mathcal{J}_n} \omega_j = 1$ ,  $\mathcal{J}_n := \overline{\max\{0, n - q + 1\}, n}$  and  $\mu_n \in (0, 2\mathcal{M}_n)$ , with

$$\mathcal{M}_n = \frac{\sum_{j \in \mathcal{J}_n} \omega_j \|P_{\mathcal{C}_j}(\mathbf{w}_n) - \mathbf{w}_n\|^2}{\|\sum_{j \in \mathcal{J}_n} \omega_j P_{\mathcal{C}_j}(\mathbf{w}_n) - \mathbf{w}_n\|^2}. \quad (3.56)$$

Under certain assumptions, it can be shown that the APSM converges to a point that lies arbitrarily close to the intersection of the infinite convex sets, with the possible exception of a finite number of sets. The theoretical properties of this algorithm will be discussed later on.

An important property of the APSM algorithm is that convex constraints can be readily incorporated in the optimization task, in a rather trivial way, *e.g.*, [82,124]. More specifically, assume that we are given an a-priori knowledge regarding the desired solution. Typical examples of such knowledge is the sparsity of the unknown vector, *e.g.*, [82], or sparsification constraints, *e.g.*, [124], robustness constraints [127], etc. This information can be embedded in an elegant way in the algorithm, as a set of convex constraints. Let us define the constraint sets  $\bar{\mathcal{C}}_n$ ,  $n \in \mathbb{Z}$ . The goal now becomes to find a point that lies in the intersection of the property sets with the constraint sets. The occurring iterative scheme is given by:

$$\mathbf{w}_{n+1} = P_{\bar{\mathcal{C}}_n} \left( \mathbf{w}_n + \mu_n \left( \sum_{j \in \mathcal{J}_n} \omega_j P_{\mathcal{C}_j}(\mathbf{w}_n) - \mathbf{w}_n \right) \right). \quad (3.57)$$

It has been verified that if properly chosen constraints are used in the learning procedure, then the performance of the algorithm is significantly enhanced. This comes at the expense of a single extra projection at each step. From a geometrical point of view, the APSM recursion is illustrated in Fig. 3.7.

Until now, we have discussed the case where the projection operators onto the sets, into which one seeks for the unknown solution, are known. For example, if the property sets take the form of hyperslabs, then, due to the fact that the projection onto that sets is known, one can employ (3.55) or (3.57) in the constrained scenario, for the unknown vector estimation. Nevertheless, a closed form expression for the projection mappings is not always the case. In the sequel, we will present an elegant way to overcome this problem.

Consider a convex loss function  $\Theta_n : \mathbb{R}^m \rightarrow \mathbb{R} : \mathbf{w} \mapsto \Theta_n(\mathbf{w})$ , which is constructed via the measurements  $d_n, \mathbf{u}_n$ . The previously described property sets, where the estimate

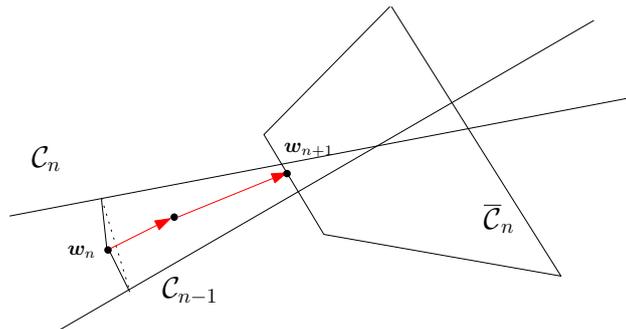


Figure 3.7: Geometrical illustration of the algorithm. The point  $w_n$  is projected onto  $C_n$ ,  $C_{n-1}$ , a convex combination of these projection is computed. The occurring vector from this step is projected onto the constraint set.

of the unknown vector would be desirable to lie, can be thought as the 0-th level sets of an appropriately chosen convex loss function. As an example, consider the quadratic  $\epsilon$ -insensitive loss function given by:

$$\Theta_n(\mathbf{w}) = \max\{0, (d_n - \mathbf{w}^T \mathbf{u}_n)^2 - \epsilon\}. \quad (3.58)$$

If the square distance between the desired response,  $d_n$ , and the response to the input, *i.e.*,  $\mathbf{w}^T \mathbf{u}_n$ , is smaller or equal than  $\epsilon$ , then a vector is considered to be in agreement with the measurements. On the contrary, if the square distance between  $d_n$  and  $\mathbf{w}^T \mathbf{u}_n$  is larger than  $\epsilon$ , then the loss function scores a quadratic penalty. Note that the corresponding property set coincides with the zero level set of the function  $\Theta_n$ . The goal is to find points which lie in this property set. However, this level set is defined by a hyperquadtratic surface (see Fig. 3.8.b) and the projection operator onto such a set is not known in analytical form. To bypass this difficulty, the following methodology is used. Being at a point  $\mathbf{w}$ , a support hyperplane, defined by a subgradient (the gradient), divides  $\mathbb{R}^m$  in two halfspaces. Projecting  $\mathbf{w}$  onto the halfspace, where the level set lies, guarantees that we get closer to the level set, where a solution lies. This is illustrated in Fig. 3.8.b.

It is interesting to notice a difference between Figs. 3.8.a and 3.8.b. In Fig. 3.8.a, the 0-th level set is a hyperslab, and one can reach it in a single step, via a single projection of  $\mathbf{w}$  onto the hyperslab. In other words, in this case, the APSM algorithm breaks down to a simple projection onto the hyperslab. The case of Fig. 3.8.b is different, where in order to approach the level set, APSM results in a succession of projections onto a sequence of

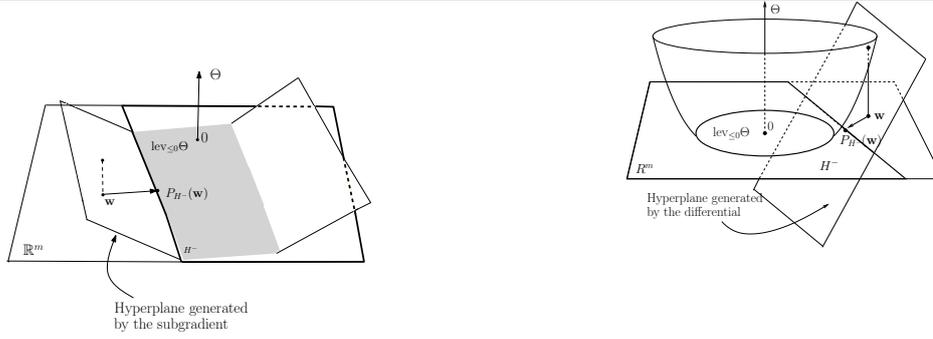


Figure 3.8: (a) A cost function whose 0-th level set is a hyperslab. (b) A cost function and the supporting hyperplane generated by the differential.

halfspaces. The algorithmic recursion, when projecting onto halfspaces generated by the subgradient, becomes

$$\mathbf{w}_{n+1} = \begin{cases} P_{\bar{C}_n} \left( \mathbf{w}_n - \lambda_n \frac{\Theta_n(\mathbf{w}_n)}{\|\Theta'_n(\mathbf{w}_n)\|^2} \Theta'_n(\mathbf{w}_n) \right), & \Theta'_n(\mathbf{w}_n) \neq \mathbf{0}_m, \\ P_{\bar{C}_n}(\mathbf{w}_n), & \Theta'_n(\mathbf{w}_n) = \mathbf{0}_m, \end{cases} \quad (3.59)$$

where  $\Theta'_n(\mathbf{w}_n)$  is the subgradient of  $\Theta_n$  at the current estimate  $\mathbf{w}_n$  and  $\lambda_n \in (0, 2)$ .

**Remark 13.** *The complexity of the APSM algorithm is linear with respect to the dimension of  $\mathbf{w}_*$ . More specifically, at each time instance the complexity accounts to  $O(qm)$ , i.e., the complexity increases w.r.t the number of parallel projections. However, as it has been experimentally verified in [82], a larger  $q$  results to a faster convergence speed.*

**Remark 14.** *Note that the APSM stems for Polyak's algorithm [110]. Nevertheless, in contrast to Polyak's algorithm, where the cost function to be minimized is fixed, here it may be time varying, a fact that allows the adopted algorithmic scheme to be applicable in dynamic and time-adaptive scenarios.*

### Theoretical Properties of the APSM

The majority of studies, dealing with the theoretical properties of the APSM algorithm, follow a deterministic route. More specifically, it has been shown that under certain assumptions, e.g., [128], the convergence of the APSM algorithm enjoys properties such as: monotonicity, optimality and strong convergence to a point that lies arbitrarily close to the property sets. The theoretical properties of the APSM under the deterministic perspective were first examined in [149] and generalized in [125, 128].

On the contrary, the efforts in [46, 131] study the convergence properties of the APSM by employing stochastic arguments. More analytically, the MSE performance of a special case of the algorithm, which uses projections onto hyperslabs, assuming Gaussian noise, is given in [131] by employing energy conservation arguments [120]. There, a single hyperslab is considered at each time instance and the step-size equals to one. The study in [46] considers the hyperslab-inspired version of the APSM, for an arbitrary number of hyperslabs processed at each time instance and an arbitrary stepsize. There it was shown that the algorithm generates a sequence of estimates which converge to a point located, with probability one, arbitrarily close to the unknown vector  $\mathbf{w}_*$ , under the bounded noise assumption.

#### Convergence properties of the APSM: the deterministic viewpoint

In this subsection we will describe, in brief, the convergence properties of the APSM employing deterministic arguments.

First of all, it is worth pointing out that (3.57) is a special case of the algorithm (3.59) (see [82]). Nevertheless, due to the fact that the majority of algorithms, which are developed in this dissertation, are variations of the scheme given in (3.57), we will discuss the theoretical properties of the latter algorithm.

The assumptions under which the algorithm converges are the following:

#### Assumptions 2.

1. Define  $\forall n \in \mathbb{Z}$ ,  $\Omega_n = \bar{\mathcal{C}}_n \cap \left( \bigcap_{j \in \mathcal{J}_n} \mathcal{C}_n \right)$ . The set  $\Omega_n$  is the intersection of the constraint sets and the sets considered at time instance  $n$ . Assume that there exists  $n_0 \in \mathbb{Z}$  such that  $\Omega := \bigcap_{n \geq n_0} \Omega_n \neq \emptyset$ . In words, with the exception of a finite number of sets  $\Omega_n$ , the rest of them have a nonempty intersection.
2. Assume a sufficiently small  $\varepsilon$  such that  $\forall n \in \mathbb{Z}$ ,  $\frac{\mu_n}{\mathcal{M}_n} \in [\varepsilon, 2 - \varepsilon]$ .
3. The interior of  $\Omega$  is assumed to be nonempty, i.e.,  $\text{int}(\Omega) \neq \emptyset$ .
4. Assume that  $\tilde{\omega} := \inf\{\omega_j : j \in \mathcal{J}_n, n \in \mathbb{Z}\} > 0$ , that is the weights, which are used for combining the projections, will not fade away as the time index  $n$  advances.

**Theorem 5.** *Under the previously mentioned assumptions, the following properties hold:*

1. **Monotonicity:** *Every step leads us closer to the intersection  $\Omega$ , i.e.,  $\forall n \geq n_0$ ,  $d(\mathbf{w}_{n+1}, \Omega) \leq d(\mathbf{w}_n, \Omega)$ .*
2. **Asymptotic Optimality.** *Asymptotically the distance of the obtained estimates from the sets  $\mathcal{C}_n$  as well as from the constraints sets  $\bar{\mathcal{C}}_n$  tends to zero, that is  $\lim_{n \rightarrow \infty} \max\{d(\mathbf{w}_n, \mathcal{C}_j) : j \in \mathcal{J}_n\} = 0$  and  $\lim_{n \rightarrow \infty} d(\mathbf{w}_n, \bar{\mathcal{C}}_n) = 0$ .*
3. *The sequence of estimates converges to a point  $\hat{\mathbf{w}}_*$ , i.e.,  $\lim_{n \rightarrow \infty} \mathbf{w}_n = \hat{\mathbf{w}}_*$ . Furthermore, it holds that:*

$$\hat{\mathbf{w}}_* \in \left( \overline{\liminf_{n \rightarrow \infty} \bar{\mathcal{C}}_n} \right) \cap \left( \overline{\liminf_{n \rightarrow \infty} \mathcal{C}_n} \right), \quad (3.60)$$

where  $\liminf_{n \rightarrow \infty} \mathcal{C}_n := \bigcup_{n \geq 0} \bigcap_{r \geq n} \mathcal{C}_r$  and the overline symbol denotes the closure of a set. In words, the algorithm converges to a point that lies arbitrarily close to the intersection of all the involved sets.

*Proof.* The proof of this theorem can be found in [82] and the theoretical properties of the most general case of the APSM are studied in [128]. □

## Stochastic Analysis of the APSM: Convergence in Probability<sup>2</sup>

In the sequel, the main theorem, which was presented in [46], is given. In a nutshell, the theorem states that the user can choose an arbitrarily small  $\alpha > 0$ , and with probability equal to 1, the distance of the estimate, to which the algorithm converges, from  $\mathbf{w}_*$  will be smaller than or equal to  $\alpha$ . In other words, *any* user-defined accuracy can be achieved.

In the theorem, which will be discussed here, stochastic arguments are employed. To this end, a slight modification in the notation will be introduced, as the stochastic processes have to be explicitly stated. Moreover, we define the probability space  $(\Xi, \mathbb{F}, \Pr)$ , where  $\Xi$  is the sample space,  $\mathbb{F}$  is the  $\sigma$ -field of events, and  $\Pr$  is the probability measure.

The classical linear model is considered, i.e.,:

$$d_n(\xi) = \mathbf{w}_*^T \mathbf{u}_n(\xi) + v_n(\xi), \quad \forall n \in \mathbb{Z}, \xi \in \Xi, \quad (3.61)$$

---

<sup>2</sup>This section involves results which are novel in this thesis

where  $(v_n(\xi))_{n \in \mathbb{Z}_{\geq 0}} \subset \mathbb{R}$  is the noise stochastic process,  $(\mathbf{u}_n(\xi))_{n \in \mathbb{Z}_{\geq 0}} \subset \mathbb{R}^m$  is the input signal process, and  $(d_n(\xi))_{n \in \mathbb{Z}_{\geq 0}} \subset \mathbb{R}$  is the process of output measurements<sup>3</sup>.

The sets into which one seeks for a candidate solution, by employing the APSM, are a slight modification of the previously described hyperslabs. Here, the definition of a hyperslab is given by:

$$S_n(\xi) := \begin{cases} \{\mathbf{w} \in \mathbb{R}^m : |d_n(\xi) - \mathbf{w}^T \mathbf{u}_n(\xi)| \leq \epsilon\}, & \text{if } \|\mathbf{u}_n(\xi)\| \in (0, \Delta), \\ \mathbb{R}^m, & \text{otherwise,} \end{cases} \quad (3.62)$$

where  $\Delta > 0$  is a user defined parameter, which is used in order to prevent the input signals from taking excessively large values. The projection onto the hyperslab in (3.62) equals to:  $\forall \mathbf{w} \in \mathbb{R}^m$ ,

$$P_{S_n(\xi)}(\mathbf{w}) = \begin{cases} \mathbf{w} + \beta_n(\xi) \mathbf{u}_n(\xi), & \text{if } \|\mathbf{u}_n(\xi)\| \in (0, \Delta), \\ \mathbf{w}, & \text{otherwise,} \end{cases} \quad (3.63)$$

$$\beta_n(\xi) := \begin{cases} \frac{d_n(\xi) - \mathbf{w}^T \mathbf{u}_n(\xi) + \epsilon}{\|\mathbf{u}_n(\xi)\|^2}, & \text{if } d_n(\xi) - \mathbf{w}^T \mathbf{u}_n(\xi) < -\epsilon, \\ 0, & \text{if } |d_n(\xi) - \mathbf{w}^T \mathbf{u}_n(\xi)| \leq \epsilon, \\ \frac{d_n(\xi) - \mathbf{w}^T \mathbf{u}_n(\xi) - \epsilon}{\|\mathbf{u}_n(\xi)\|^2}, & \text{if } d_n(\xi) - \mathbf{w}^T \mathbf{u}_n(\xi) > \epsilon. \end{cases}$$

The recursion that describes this special case of the APSM, which is considered here, occurs if in (3.55) we employ the projection operators onto the hyperslabs. More specifically, given any starting point  $\mathbf{w}_0(\xi)$ , define  $\forall n \in \mathbb{Z}_{\geq 0}$ ,

$$\mathbf{w}_{n+1}(\xi) := \mathbf{w}_n(\xi) + \mu_n(\xi) \left( \sum_{j \in \mathcal{J}_n} \omega_j^{(n)} P_{S_j(\xi)}(\mathbf{w}_n(\xi)) - \mathbf{w}_n(\xi) \right), \quad (3.64)$$

where the extrapolation parameter  $\mu_n(\xi)$  can take any value that lies within the interval

---

<sup>3</sup>In the stochastic quantities, the term  $(\xi)$  will be embedded.

$\mu_n(\xi) \in [\varepsilon \mathcal{M}_n(\xi), (2 - \varepsilon) \mathcal{M}_n(\xi)]$ , where

$$\mathcal{M}_n(\xi) := \begin{cases} \frac{\sum_{j \in \mathcal{J}_n} \omega_j^{(n)} \left\| P_{S_j(\xi)}(\mathbf{w}_n(\xi)) - \mathbf{w}_n(\xi) \right\|^2}{\left\| \sum_{j \in \mathcal{J}_n} \omega_j^{(n)} P_{S_j(\xi)}(\mathbf{w}_n(\xi)) - \mathbf{w}_n(\xi) \right\|^2}, & \text{if } \left\| \sum_{j \in \mathcal{J}_n} \omega_j^{(n)} P_{S_j(\xi)}(\mathbf{w}_n(\xi)) - \mathbf{w}_n(\xi) \right\| \neq 0, \\ 1, & \text{otherwise,} \end{cases} \quad (3.65)$$

and  $\varepsilon$  is a user-defined parameter s.t.  $\varepsilon \in (0, 1]$ , and obviously  $\mathcal{M}_n(\xi) \geq 1$ ,  $\forall n \in \mathbb{Z}_{\geq 0}$ ,  $\forall \xi \in \Xi$ . The terms  $\mathcal{J}_n$ ,  $\omega_j$  are defined in a similar way as in the algorithm given in (3.55).

Let us redefine, here,  $\forall n \in \mathbb{Z}_{\geq 0}$ ,  $\forall \xi \in \Xi$ , the set  $\Omega_n(\xi) := \bigcap_{j \in \mathcal{J}_n} S_j(\xi)$ . Then, let  $\Gamma := \left\{ \xi \in \Xi : \exists n_0 := n_0(\xi) \in \mathbb{Z}_{\geq 0} \text{ s.t. } \text{int}\left(\bigcap_{n \geq n_0} \Omega_n(\xi)\right) \neq \emptyset \right\}$ .

**Fact 1.** *Assume that  $\Gamma \neq \emptyset$ . Then,  $\forall \xi \in \Gamma$ , the APSM, illustrated in Algorithm 3.64, produces a sequence of estimates  $(\mathbf{w}_n(\xi))_{n \in \mathbb{Z}_{\geq 0}}$  which converges to a point  $\hat{\mathbf{w}}(\xi)$ , i.e.,  $\lim_{n \rightarrow \infty} \mathbf{w}_n(\xi) = \hat{\mathbf{w}}(\xi)$ . Moreover,  $\forall \xi \in \Gamma$ , the following holds true:  $\lim_{n \rightarrow \infty} d(\mathbf{w}_n(\xi), S_n(\xi)) = 0$  [134].*

The assumptions, under which the algorithms converges are the following:

### Assumptions 3.

1. *The noise process is bounded, i.e.,  $\exists \rho > 0$ , s.t.  $|v_n(\xi)| \leq \rho$ ,  $\forall n \in \mathbb{Z}_{\geq 0}$ , almost surely (a.s.).*
2. *The input vector is defined as  $\mathbf{u}_n(\xi) := [u_n(\xi), u_{n-1}(\xi), \dots, u_{n-m+1}(\xi)]^T$ , where  $(u_n(\xi))_{n \in \mathbb{Z}}$  is a realization of an i.i.d. process, with a probability density function  $f$  whose support contains at least one of the intervals  $(0, \infty)$ ,  $(-\infty, 0)$ , i.e.,  $(0, \infty)$  or/and  $(-\infty, 0) \subset \text{supp}(f) := \{u \in \mathbb{R} : f(u) > 0\}$ , and every interval of non-zero length, which belongs to  $\text{supp}(f)$ , owns a non-zero probability.*

Note that the assumption regarding the input covers a variety of distributions, e.g., Gaussian, generalized Gaussian, Cauchy, Laplacian. Moreover, it should be pointed out that the sliding window construction of the input vectors is not restrictive. The theorem to be presented in the sequel can be generalized in cases where the input vectors do not obey to this sliding window formulation. In such a case, successive input vectors do not share common components, and assuming that they are independent, the proof of the theorem

follows similar steps. Finally, it should be stressed that the adopted assumption regarding the input is realistic in contrast to an independence assumption (see for further details [120]), often adopted in adaptive filtering. As we have already discussed previously, this assumption states that the input vectors are independent  $\forall n \in \mathbb{Z}$ , which does not hold true if the input has the sliding-window form. Albeit unrealistic, such an independence assumption is widely used in adaptive filtering, since it simplifies the analysis. In Assumption 3.2, the vector entries are assumed to be independent, instead of the whole vectors, which holds true only for properly chosen inputs.

Before we derive the main theorem, we need to prove the following lemma.

**Lemma 2.** *Let Assumption 3.1 hold true, and choose the user-defined hyper slab parameter  $\epsilon$  to be larger than the bound of the noise, i.e.,  $\epsilon > \rho$ . Moreover, fix arbitrarily the value of the user-defined parameter  $\Delta > 0$ . Then,  $\Pr(\Gamma) = 1$ .*

*Proof.* First of all, notice that  $\forall n \in \mathbb{Z}_{\geq 0}$  s.t.  $\|\mathbf{u}_n(\xi)\| \in (0, \Delta)$ ,  $|d_n(\xi) - \mathbf{w}_*^T \mathbf{u}_n(\xi)| = |v_n(\xi)| \leq \rho < \epsilon$ , a.s. Hence, for such  $n$ ,  $\mathbf{w}_* \in S_n(\xi)$ . For all those  $n$  s.t.  $\|\mathbf{u}_n(\xi)\| \notin (0, \Delta)$ , by (3.62),  $\mathbf{w}_* \in S_n(\xi) = \mathbb{R}^m$ . To summarize,  $\mathbf{w}_* \in S_n(\xi)$ ,  $\forall n \in \mathbb{Z}_{\geq 0}$ . Moreover,  $\mathbf{w}_* \in \Omega_n(\xi)$ ,  $\forall n \in \mathbb{Z}_{\geq 0}$ , which implies in turn that  $\mathbf{w}_* \in \bigcap_{n \in \mathbb{Z}_{\geq 0}} \Omega_n(\xi)$ . Fix, now, any  $\mathbf{w} \in B(\mathbf{w}_*, (\epsilon - \rho)/\Delta)$ . Obviously,  $\|\mathbf{w} - \mathbf{w}_*\| < (\epsilon - \rho)/\Delta$ . It can be easily verified that  $\forall n \in \mathbb{Z}_{\geq 0}$ ,

$$\begin{aligned} |d_n(\xi) - \mathbf{u}_n^T(\xi) \mathbf{w}| &= |\mathbf{u}_n^T(\xi) (\mathbf{w}_* - \mathbf{w}) + v_n(\xi)| \\ &\leq \|\mathbf{u}_n(\xi)\| \|\mathbf{w}_* - \mathbf{w}\| + |v_n(\xi)| < \Delta \frac{\epsilon - \rho}{\Delta} + \rho = \epsilon, \text{ a.s.} \end{aligned}$$

Hence,  $B(\mathbf{w}_*, (\epsilon - \rho)/\Delta) \subset \bigcap_{n \in \mathbb{Z}_{\geq 0}} \Omega_n(\xi)$ . This suggests that  $\text{int}(\bigcap_{n \in \mathbb{Z}_{\geq 0}} \Omega_n(\xi)) \neq \emptyset$ , which establishes the claim of Lemma 2.  $\square$

**Theorem 6.** *Let Assumptions 3 hold true. Choose an arbitrary accuracy  $\alpha > 0$ . Moreover, choose a  $\Delta \in (\frac{4\epsilon\sqrt{m}}{\alpha}, \infty)$ , where the parameter  $\Delta$  refers to (3.62), with  $\epsilon > \rho$ . Then,  $(\mathbf{w}_n(\cdot))_{n \in \mathbb{Z}}$  of Algorithm 3.64 converges to a point  $\hat{\mathbf{w}}(\cdot)$ . If  $\hat{\mathbf{w}}(\cdot)$  is bounded a.s., i.e.,  $\exists B > 0$  s.t.  $\Pr(\{\xi \in \Xi : \|\hat{\mathbf{w}}(\xi)\| \leq B\}) = 1$ , then  $\hat{\mathbf{w}}(\cdot)$  achieves the user-defined accuracy  $\alpha > 0$ , i.e.,  $\Pr(\{\xi \in \Xi : \|\mathbf{w}_* - \hat{\mathbf{w}}(\xi)\| \leq \alpha\}) = 1$ .*

*Proof.* The assumptions of the theorem, as well as Lemma 2 and Fact 1, imply the existence of  $\hat{\mathbf{w}}(\cdot)$ . Hence, there exists a subset  $\Xi'$  of  $\Xi$ , with  $\Pr(\Xi') = 1$ , such that  $\hat{\mathbf{w}}(\cdot)$  exists  $\forall \xi \in \Xi'$ ,

and a sufficiently large  $B' > 0$  can be chosen s.t.  $\|\mathbf{w}_* - \hat{\mathbf{w}}(\xi)\| \leq B', \forall \xi \in \Xi'$ .

Let  $\mathfrak{G} := \{\xi \in \Xi' : \|\mathbf{w}_* - \hat{\mathbf{w}}(\xi)\| > \alpha\}$ . In order to establish the second claim of our theorem, it is sufficient to prove that  $\Pr(\mathfrak{G}) = 0$ . To this end, given any  $i \in \overline{1, m}$ , define  $\mathfrak{P}_i := \{\xi \in \Xi' : |[w_* - \hat{w}(\xi)]_i| > \alpha/\sqrt{m}\}$ . Notice here that

$$\mathfrak{G} \subset \bigcup_{i=1}^m \mathfrak{P}_i. \quad (3.66)$$

Indeed, given any  $\xi \in \mathfrak{G}$ , there exists an  $i_0 \in \overline{1, m}$  s.t.  $|[w_* - \hat{w}(\xi)]_{i_0}| > \alpha/\sqrt{m}$ , i.e.,  $\xi \in \mathfrak{P}_{i_0}$ .

Fix now arbitrarily any  $i \in \overline{1, m}$ , and define the following sequence of events;  $\forall n \in \mathbb{Z}_{\geq 0}$ ,

$$E_{i,n}^\pm := \left\{ \begin{array}{l} \xi \in \Xi' : \text{sgn}([u_n(\xi)]_i) = \pm 1, \\ \frac{4\epsilon\sqrt{m}}{\alpha} < |[u_n(\xi)]_i| < \Delta, \\ |[u_n(\xi)]_j| < \min\left\{\frac{\epsilon}{(m-1)B'}, \Delta\right\}, \forall j \neq i \end{array} \right\}. \quad (3.67)$$

Without any loss of generality, let us assume that  $(0, \infty) \subset \text{supp}(f)$  holds true. Then, our assumptions guarantee that any event  $E_{i,n}^+$  is non-empty and possesses a positive probability,  $\forall n$ . Fix, now, arbitrarily any  $\bar{\xi} \in \mathfrak{P}_i$ , and assume that

$$\bar{\xi} \in \bigcap_{n \geq 0} \bigcup_{k \geq n} E_{i,km}^+. \quad (3.68)$$

The reason for choosing the subsequence  $(E_{i,km}^+)_{k \in \mathbb{Z}}$  will be given shortly, after (3.72).

Our assumption (3.68) can be rephrased equivalently as follows; there exists a subsequence  $(k_l)_{l \in \mathbb{Z}}$  s.t.  $\bar{\xi} \in E_{i,k_l m}^+, \forall l \in \mathbb{Z}$ . Moreover, given  $\bar{\xi} \in \mathfrak{P}_i$ , either  $\text{sgn}([w_* - \hat{w}(\bar{\xi})]_i) = +1$  or  $\text{sgn}([w_* - \hat{w}(\bar{\xi})]_i) = -1$ . If  $\text{sgn}([w_* - \hat{w}(\bar{\xi})]_i) = +1$ , recalling Assumption 3.1, (3.67), and the fact that  $\epsilon \geq \rho, \forall l \in \mathbb{Z}$ ,

$$\begin{aligned} d_{k_l m}(\bar{\xi}) - \mathbf{u}_{k_l m}^T(\bar{\xi})\hat{\mathbf{w}}(\bar{\xi}) &= v_{k_l m}(\bar{\xi}) + \mathbf{u}_{k_l m}^T(\bar{\xi})(\mathbf{w}_* - \hat{\mathbf{w}}(\bar{\xi})) \\ &\geq -\rho + \sum_{j \neq i} [u_{k_l m}(\bar{\xi})]_j [w_* - \hat{w}(\bar{\xi})]_j + [u_{k_l m}(\bar{\xi})]_i [w_* - \hat{w}(\bar{\xi})]_i \\ &\geq -\epsilon - \sum_{j \neq i} |[u_{k_l m}(\bar{\xi})]_j| |[w_* - \hat{w}(\bar{\xi})]_j| + |[u_{k_l m}(\bar{\xi})]_i| |[w_* - \hat{w}(\bar{\xi})]_i| \\ &> -\epsilon - \sum_{j \neq i} \frac{\epsilon}{(m-1)B'} B' + \frac{4\epsilon\sqrt{m}}{\alpha} \frac{\alpha}{\sqrt{m}} = -\epsilon - \epsilon + 4\epsilon = 2\epsilon. \end{aligned} \quad (3.69)$$

In the case where  $\text{sgn}([w_* - \hat{w}(\bar{\xi})]_i) = -1$ , similar computations result into  $d_{k_l m}(\bar{\xi}) -$

$\mathbf{u}_{k_l m}^T(\bar{\xi})\hat{\mathbf{w}}(\bar{\xi}) < -2\epsilon, \forall l \in \mathbb{Z}$ . For both of these cases, (3.63) and (3.69) suggest that  $\forall l \in \mathbb{Z}$ ,

$$d(\hat{\mathbf{w}}(\bar{\xi}), S_{k_l m}(\bar{\xi})) = \left\| \hat{\mathbf{w}}(\bar{\xi}) - P_{S_{k_l m}(\bar{\xi})}(\hat{\mathbf{w}}(\bar{\xi})) \right\| > \frac{\epsilon}{\Delta}. \quad (3.70)$$

Since  $\forall l \in \mathbb{Z}$ ,  $d(\hat{\mathbf{w}}(\bar{\xi}), S_{k_l m}(\bar{\xi})) \leq d(\mathbf{w}_{k_l m}(\bar{\xi}), S_{k_l m}(\bar{\xi})) + \|\hat{\mathbf{w}}(\bar{\xi}) - \mathbf{w}_{k_l m}(\bar{\xi})\|$ , then Fact 1 implies that  $\lim_{l \rightarrow \infty} d(\hat{\mathbf{w}}(\bar{\xi}), S_{k_l m}(\bar{\xi})) = 0$ . However, this contradicts (3.70), and, thus, our initial claim in (3.68) is false.

For compact notations, let us define also  $\mathfrak{A}_{i,n}^+ := \bigcup_{k \geq n} E_{i,km}^+, \forall n \geq 0$ . Since  $\bar{\xi}$  was chosen arbitrarily from  $\mathfrak{P}_i$  in the previous paragraphs, the contra position of (3.68) can be formulated as follows:

$$\mathfrak{P}_i \cap \bigcap_{n \geq 0} \mathfrak{A}_{i,n}^+ = \emptyset. \quad (3.71)$$

Notice, now, that  $\forall n \in \mathbb{Z}, \forall K \in \mathbb{Z} \setminus \{0\}$ ,

$$\begin{aligned} 1 &\geq \Pr(\mathfrak{A}_{i,n}^+) \geq \Pr\left(\bigcup_{k=n}^{n+K-1} E_{i,km}^+\right) \\ &= 1 - \Pr\left(\bigcap_{k=n}^{n+K-1} \Xi' \setminus E_{i,km}^+\right). \end{aligned} \quad (3.72)$$

Since the members of the process  $(u_n)_{n \in \mathbb{Z}_{\geq 0}}$  are independent, the members of the vector-valued process  $(\mathbf{u}_{km})_{k \in \mathbb{Z}}$  are also mutually independent<sup>4</sup>. This is the reason behind the choice of the subsequence  $(E_{i,km}^+)_{k \in \mathbb{Z}}$ . Since, also,  $(u_n)_{n \in \mathbb{Z}_{\geq 0}}$  is i.i.d., then  $\Pr(E_{i,km}^+)$  attains a positive value independent of the index  $k$ , *i.e.*,  $\Pr(E_{i,km}^+) =: \lambda_+ \in (0, 1), \forall k \in \mathbb{N}$ . Hence, it can be verified by the independency of the events  $(\Xi' \setminus E_{i,km}^+)_{k \in \mathbb{Z}}$  that  $\Pr\left(\bigcap_{k=n}^{n+K-1} \Xi' \setminus E_{i,km}^+\right) = \prod_{k=n}^{n+K-1} \Pr(\Xi' \setminus E_{i,km}^+) = (1 - \lambda_+)^K$ , and (3.72) becomes  $1 \geq \Pr(\mathfrak{A}_{i,n}^+) \geq 1 - (1 - \lambda_+)^K$ . If  $\lim_{K \rightarrow \infty}$  is applied to the previous inequality, then it can be verified by  $(1 - \lambda_+) \in (0, 1)$  that  $\Pr(\mathfrak{A}_{i,n}^+) = 1, \forall n \in \mathbb{Z}$ .

To summarize:  $\mathfrak{A}_{i,n+1}^+ \subset \mathfrak{A}_{i,n}^+$ , and  $\Pr(\mathfrak{A}_{i,n}^+) = 1, \forall n \in \mathbb{Z}$ . Based on these outcomes, a classical result of probability theory, *e.g.*, [91, p. 152], suggests that  $\Pr\left(\bigcap_{n \geq 0} \mathfrak{A}_{i,n}^+\right) = \lim_{n \rightarrow \infty} \Pr(\mathfrak{A}_{i,n}^+) = 1$ . Therefore, by (3.71),  $0 \leq \Pr(\mathfrak{P}_i) \leq 1 - \Pr\left(\bigcap_{n \geq 0} \mathfrak{A}_{i,n}^+\right) = 0$ .

Since  $i$  was chosen arbitrarily from  $\overline{1, m}$ , the previous discussion leads to  $\Pr(\mathfrak{P}_i) = 0$ ,

---

<sup>4</sup>In the case where the input has not a sliding window form, the proof of the theorem follows a similar path.

$\forall i \in \overline{1, m}$ . Therefore, (3.66) suggests that  $0 \leq \Pr(\mathfrak{G}) \leq \Pr(\bigcup_{i=1}^m \mathfrak{P}_i) \leq \sum_{i=1}^m \Pr(\mathfrak{P}_i) = 0$ , which establishes the second claim of the theorem.  $\square$

### 3.4 Numerical Examples

In this section, numerical examples in the context of the system identification task are provided. The goal is twofold: a) to demonstrate the performance of the algorithms described in this chapter, in a typical setup of Gaussian unbounded noise and b) to validate numerically Theorem 4.

The task is the estimation of an unknown vector  $\mathbf{w}_*$  of dimension  $m = 40$ , through measurements  $d_n, \mathbf{u}_n$ , related via the linear model. The APSM algorithm is compared to the LMS, the RLS and the Normalized LMS (NLMS). It is worth pointing out that the NLMS is a special case of the hyperslab-based APSM and occurs if one lets  $q = 1$  and  $\epsilon = 0$ . In the first experiment, we assume that the variance of the noise equals to  $10^{-2}$  and the input coefficients are drawn from the Gaussian distribution, with variance equal to 1. The step-size for the LMS equals to 0.1 whereas for the NLMS 1. Moreover, the forgetting factor  $\zeta$  in the RLS equals to 1. Finally, the parameters in the APSM algorithm are:  $q = 15$ ,  $\mu_n = (1/2) \times \mathcal{M}_n$  and  $\epsilon = \sqrt{2}\sigma$ . The Mean Square Deviation (MSD), *i.e.*,  $\text{MSD}_n := \|\mathbf{w}_* - \mathbf{w}_n\|^2$ , is presented and the curves occur from 100 Monte Carlo runs.

Fig. 3.9 indicates that the RLS and the APSM have similar convergence speeds. However, the former algorithm converges to a lower steady state error floor, at the expense of a higher complexity. Furthermore, the NLMS algorithm converges faster compared to the LMS, to the same steady state error. Both algorithms, converge slower compared to the RLS and the APSM.

In the second experiment, the parameters are the same as in the first one, albeit the coefficients of the input are assumed to be strongly correlated. More specifically, we assume that the input coefficients are related via  $u_n = 0.8u_{n-1} + \chi_n$ , where  $\chi_n$  follows the Gaussian distribution with zero mean and variance 1. The step-sizes for the LMS and the NLMS take the maximum value for which the algorithms converge. As it can be readily seen in Fig. 3.10, the performance of the LMS and the NLMS algorithms is degraded compared to the previous experiment. This is expected, since the LMS-based algorithms are sensitive to colored inputs.

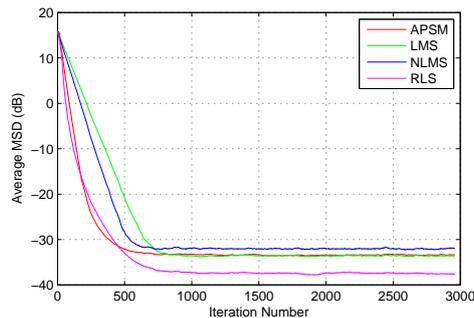


Figure 3.9: MSD performance for the first experiment.

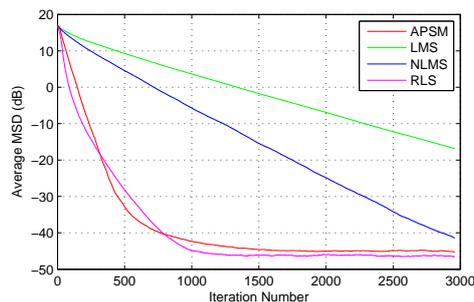


Figure 3.10: MSD performance for the second experiment.

This fact is also theoretically validated, *e.g.*, [120]. On the contrary, we observe that the performance of the RLS and the APSM algorithm is not affected significantly.

In the next experiments, the numerical validation of Theorem 4 will be studied. The length of  $\mathbf{w}_*$  is set to be equal to  $m = 64$ , and its coefficients are drawn from the Gaussian distribution with standard deviation set to 1. The input samples are generated by the Gaussian distribution with standard deviation equal to 1. The noise samples are generated by the Gaussian distribution, with standard deviation  $\sigma = \sqrt{0.5}$ , and samples with amplitude equal to or greater than  $\rho = 2 * \sigma$  are set equal to  $\rho$ . Moreover, for the APSM,  $q = 32$ ,  $\omega_j = 1/q, \forall j \in \mathcal{J}_n$ ,  $\mu_n = \mathcal{M}_n/2$ , and  $\epsilon = \rho + 10^{-3}$ . In the third experiment, three different values for  $\alpha$  are chosen, *i.e.*,  $\alpha \in \{10^{-2}, 10^{-4}, 10^{-6}\}$ , and 1000 independent experiments are performed for every  $\alpha$ . The number of times that the desired accuracy is achieved is counted. It was observed that all of the accuracies are achieved, since the Euclidean distance of the vector, to which the algorithm converges, from the estimand stays smaller than  $\alpha$ , for *every* experiment. This fact corroborates the theoretical findings of the study.

In the fourth experiment (Fig. 3.11), the APSM, with  $q = 32$  (APSM <sub>$q=32$</sub> ), the RLS, the LMS, and the NLMS are validated. To study the robustness of the APSM, whenever

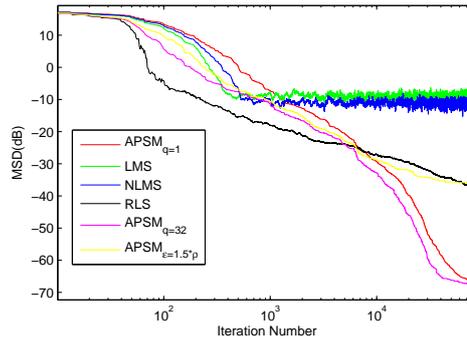


Figure 3.11: MSD curves for the 2nd experiment. Only a single iteration is plotted, since Theorem 4 does not involve the expectation operator, where the averaging of independent iterations is necessary.

the exact value of  $\sigma$  is not known, we validate it in the cases where  $q = 1$  ( $\text{APSM}_{q=1}$ ), and  $\epsilon = 1.5\rho$  ( $\text{APSM}_{\epsilon=1.5*\rho}$ ). The step-sizes for the LMS and the NLMS are equal to  $10^{-2}$  and 0.5, respectively, and they are chosen so that both algorithms exhibit similar convergence speed. It can be readily seen that the  $\text{APSM}_{q=32}$ , the  $\text{APSM}_{\epsilon=1.5*\rho}$  and the RLS outperforms the LMS and the NLMS, at the expense of larger complexity. It is interesting to notice that  $\text{APSM}_{q=1}$  outperforms also the LMS-based algorithms, although their complexities are similar. This behavior is a direct consequence of the fact that by employing the hyperslabs, in the case of bounded noise, the estimand belongs surely to these sets. In NLMS, since  $\epsilon = 0$ , the  $\mathbf{w}_*$  may not belong to a property set. Moreover, it can be seen that APSM-based algorithms converge to lower steady state error floors, compared to the RLS, except from  $\text{APSM}_{\epsilon=1.5*\rho}$ .

It is worth noticing that similar behavior was observed also in cases of larger  $m$ . In such scenarios, the designer possesses the additional freedom of choosing the values of  $q$  from larger intervals. In general, and as Fig. 3.11 attests, the larger the  $q$ , the faster the convergence speed of APSM [134]. However, large  $q$  values come at the expense of increased computational complexity.

# Chapter 4

## Adaptive Robust Algorithms for Distributed Learning

*In this chapter, distributed algorithms, which follow the diffusion rationale and belong to the family of the Adaptive Projected Subgradient Method, will be developed. The proposed algorithms adopt the novel combine–project–adapt cooperation protocol. The intermediate extra projection step of this protocol “harmonizes” the local information, which comprises the input/output measurements, with the information coming from the neighborhood, i.e., the estimates obtained from the neighboring nodes. It turns out that this enhances significantly the performance of the respective schemes. Moreover, the possibility that some of the nodes may fail is also considered and it is addressed by building the required property sets via the use of loss functions, which have been used in the context of robust statistics. Under some mild assumptions, the developed algorithms enjoy monotonicity, asymptotic optimality, asymptotic consensus, strong convergence to a point that lies in the consensus subspace and linear complexity with respect to the number of unknown parameters. Finally, system–identification experiments verify the validity of the proposed algorithmic schemes, which are compared to other state of the art algorithms which have been developed in the context of adaptive distributed learning.*

## 4.1 Diffusion Adaptive Algorithm Using Projections onto Hyperslabs

Recall the problem described in Chapter 2. A network of  $N$  nodes is considered and each node,  $k$ , at time  $n$ , has access to the measurements  $d_{k,n} \in \mathbb{R}$ ,  $\mathbf{u}_{k,n} \in \mathbb{R}^m$  generated by the linear system:

$$d_{k,n} = \mathbf{w}_*^T \mathbf{u}_{k,n} + v_{k,n}, \quad (4.1)$$

where  $v_{k,n}$  is an additive noise process of zero mean and variance  $\sigma_k^2$ . The goal is the estimation of the  $m \times 1$  vector  $\mathbf{w}_*$ .

In this chapter, an APSM-based scheme, which employs projections onto hyperslabs, is developed. The scheme is brought in a distributed fashion by following a similar rationale as the one of the diffusion LMS/RLS, which were presented in the previous chapter. Although this is a possibility, here an extra step is added, that follows the combination stage and precedes the adaptation one. As it will become apparent in the simulations section, such a step turns out to be beneficial to the convergence performance, at the expense of the minimal cost of an extra projection onto a hyperslab  $S'_{k,n}$ , which is defined as

$$S'_{k,n} = \{\mathbf{w} \in \mathbb{R}^m : |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \epsilon'_k\},$$

where  $\epsilon'_k > \epsilon_k$  and  $\epsilon_k$  is the user defined parameter associated with the hyperslabs, that will be used in the adaptation step at node  $k$ , *i.e.*,

$$S_{k,n} = \{\mathbf{w} \in \mathbb{R}^m : |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \epsilon_k\}.$$

The algorithm comprises the following steps:

1. **Combination Step:** The estimates from the nodes that belong to  $\mathcal{N}_k$  are received and convexly combined with respect to the combination weights  $a_{k,l}$ .
2. **Projection Step:** The resulting aggregate is first projected onto the hyperslab  $S'_{k,n}$ <sup>1</sup>.
3. **Adaptation Step:** The adaptation step is performed.

---

<sup>1</sup>The projection of a point onto a hyperslab is provided in Chapter 3.

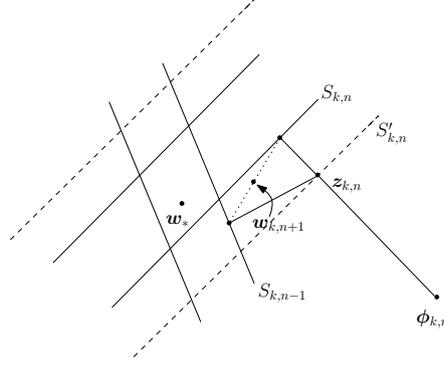


Figure 4.1: Illustration of an iteration for the case of  $q = 2$ . The aggregate is projected onto an external hyperslab and the result,  $\mathbf{z}_{k,n}$ , is used in the adaptation step. Note that  $\mathbf{z}_{k,n}$  is projected onto  $S_{k,n}$  and  $S_{k,n-1}$ , and the projections are combined together. The update estimate,  $\mathbf{w}_{k,n+1}$  lies closer to the intersection of the hyperslabs, compared to  $\phi_{k,n}$ . Note, also, that the hyperslab  $S'_{k,n}$  contains  $S_{k,n}$ .

**Algorithm 1:**

$$\phi_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{w}_{l,n}, \quad (4.2a)$$

$$\mathbf{z}_{k,n} = P_{S'_{k,n}}(\phi_{k,n}), \quad (4.2b)$$

$$\mathbf{w}_{k,n+1} = \mathbf{z}_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_{k,n}) - \mathbf{z}_{k,n} \right), \quad (4.2c)$$

where  $P_{S'_{k,n}}$  and  $P_{S_{k,n}}$  are the projection operators onto the respective hyperslabs,  $\sum_{j \in \mathcal{J}_n} \omega_{k,j} = 1$  and  $\mathcal{J}_n := \overline{\max\{0, n - q + 1\}, n}$  (see also chapter 3).

The geometry for the case of  $q = 2$  is shown in Fig. 4.1. The aggregate  $\phi_{k,n}$  is first projected onto  $S'_{k,n}$  to provide  $\mathbf{z}_{k,n}$ . The latter is the point that gets involved in the adaptation step, which consists of the projections onto  $S_{k,n}$  and  $S_{k,n-1}$ , and in their subsequent convex combination in accordance to the APSM rationale. As it will be shown in the Appendix B, convergence is guaranteed if  $\mu_{k,n} \in (0, 2\mathcal{M}_{k,n})$  where

$$\mathcal{M}_{k,n} = \begin{cases} \frac{\sum_{j \in \mathcal{J}_n} \omega_{k,j} \|P_{S_{k,j}}(\mathbf{z}_{k,n}) - \mathbf{z}_{k,n}\|^2}{\|\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_{k,n}) - \mathbf{z}_{k,n}\|^2}, & \text{if } \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_{k,n}) \neq \mathbf{z}_{k,n}, \\ 1, & \text{otherwise.} \end{cases} \quad (4.3)$$

Also,  $\mathcal{M}_{k,n} \geq 1$  [124], hence  $\mu_{k,n} = 1$  is an acceptable step-size  $\forall n \in \mathbb{Z}$ .

Before we proceed further, let us “translate” equations (4.2a)-(4.2c) from the local to a

global form to include all nodes. It is a matter of simple algebra to see that

$$\underline{\mathbf{z}}_n := \begin{bmatrix} \mathbf{z}_{1,n} \\ \vdots \\ \mathbf{z}_{N,n} \end{bmatrix} = \begin{bmatrix} P_{S'_{1,n}} \left( \sum_{l \in \mathcal{N}_1} a_{1,l} \mathbf{w}_{l,n-1} \right) \\ P_{S'_{2,n}} \left( \sum_{l \in \mathcal{N}_2} a_{2,l} \mathbf{w}_{l,n-1} \right) \\ \vdots \\ P_{S'_{N,n}} \left( \sum_{l \in \mathcal{N}_N} a_{N,l} \mathbf{w}_{l,n-1} \right) \end{bmatrix}, \quad (4.4)$$

$$\underline{\mathbf{w}}_{n+1} := \begin{bmatrix} \mathbf{w}_{1,n+1} \\ \vdots \\ \mathbf{w}_{N,n+1} \end{bmatrix} = \underline{\mathbf{z}}_n + \mathbf{M}_n \begin{bmatrix} \sum_{j \in \mathcal{J}_n} \omega_{1,j} P_{S_{1,j}}(\mathbf{z}_{1,n}) - \mathbf{z}_{1,n} \\ \sum_{j \in \mathcal{J}_n} \omega_{2,j} P_{S_{2,j}}(\mathbf{z}_{2,n}) - \mathbf{z}_{2,n} \\ \vdots \\ \sum_{j \in \mathcal{J}_n} \omega_{N,j} P_{S_{N,j}}(\mathbf{z}_{N,n}) - \mathbf{z}_{N,n} \end{bmatrix}, \quad (4.5)$$

where  $\mathbf{M}_n = \text{diag}\{\mu_{1,n} \mathbf{I}_m, \mu_{2,n} \mathbf{I}_m, \dots, \mu_{N,n} \mathbf{I}_m\}$ .

Observe that the resulting scheme is structurally simple. It consists of two vector equations, one for the combination/fusion and one for the update. The main operations which are involved are projections. The complexity per time update amounts to  $O(qm)$ , where  $q$  is the number of hyperslabs onto which one projects during the adaptation step, in every node. Moreover, note that in a parallel processing environment, the  $q$  projections can take place concurrently.

**Remark 15.** *A projection based algorithm of the adapt–combine type has been developed in [31]. In contrast, the new algorithm follows the combine–project–adapt philosophy. This scenario gives us the advantage of being able to accommodate the extra projection. The notion behind this extra step is to “harmonize”, at each node, the information that is sensed locally with the information transmitted by the neighboring nodes. Although all nodes search for the same unknown vector (i.e.,  $\mathbf{w}_*$ ), the statistics of the regressors are, in general, different. For this reason, by projecting the aggregate (which is the information collected from the neighborhood) onto a hyperslab, i.e.,  $S'_{k,n}$ , which is constructed using information that is sensed locally, we push the aggregate closer to the feasible region and the convergence is accelerated, which is verified by the experiments. Note that if we let  $P_{S'_{k,n}}$  be the identity mapping, then the algorithm conforms to the simple combine–adapt cooperation protocol. Another notable difference with [31] is that the theoretical analysis is different and strong convergence has been proved for the scheme developed here, which was not the case with [31].*

## 4.2 Theoretical Analysis

As it will be established in Appendix B, the algorithm (4.4)-(4.5) enjoys a number of nice convergence properties such as monotonicity, strong convergence to a point and consensus. To prove these properties the following assumptions must hold.

**Assumptions:**

- (a) There exists a non-negative integer, say  $n_0$ , for which  $\Omega = \bigcap_{n \geq n_0} \Omega_n \neq \emptyset$ , where  $\Omega_n = \bigcap_{k \in \mathcal{N}} \bigcap_{j \in \mathcal{J}_n} S_{k,j}$ . This means that the hyperslabs share a non empty intersection. However, it is possible for a finite number of them,  $n_0$ , not to share a common intersection. This is important, since the presence of a finite number of outliers does not affect convergence. A case where such an assumption holds true is whenever the noise is bounded, and the width of the hyperslabs, determined by the parameter  $\epsilon_k$ , is chosen appropriately so as to contain  $\mathbf{w}_*$  (see also Chapter 3).
- (b) The local stepsize  $\mu_{k,n} \in (0, 2\mathcal{M}_{k,n})$ ,  $k = 1, \dots, N$ . As it is always the case with adaptive algorithms, the adaptation step must lie within an interval, in order to guarantee convergence. Here, this interval is computed by the algorithm itself.
- (c) Let  $\varepsilon_1 > 0$  be a sufficiently small constant such that  $\mu_{k,n} \in [\varepsilon_1 \mathcal{M}_{k,n}, \mathcal{M}_{k,n}(2 - \varepsilon_1)]$ ,  $k = 1, \dots, N$ .
- (d) In order to guarantee consensus the following statement must hold:  $\epsilon'_k > \epsilon_k$ ,  $\forall k \in \mathcal{N}$ .
- (e) Let us define  $\mathfrak{C} := \mathcal{O} \cap \Omega$ , where the Cartesian product space,  $\Omega$ , is defined as  $\Omega = \underbrace{\Omega \times \dots \times \Omega}_N$  and  $\mathcal{O}$  is the consensus subspace, which is defined in Appendix A. We assume that  $\text{ri}_{\mathcal{O}}(\mathfrak{C}) \neq \emptyset$ , where  $\text{ri}_{\mathcal{O}}(\mathfrak{C})$  stands for the relative interior of  $\mathfrak{C}$  with respect to  $\mathcal{O}$ , and its definition can be found in chapter 3.

**Theorem 1:** For any  $\underline{\mathbf{w}}_* \in \mathfrak{C}$ , which is of the form  $\underline{\mathbf{w}}_* = [\hat{\mathbf{w}}_*^T, \dots, \hat{\mathbf{w}}_*^T]^T \in \mathbb{R}^{Nm}$ , with  $\hat{\mathbf{w}}_* \in \Omega$ , the following hold true:

1. **Monotone Approximation.** Under assumptions (a), (b),

$$\|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\| \leq \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|, \forall n \geq n_0. \quad (4.6)$$

The previous inequality yields that, the distance of  $\underline{\mathbf{w}}_n$  from any point  $\underline{\mathbf{w}}_* \in \mathfrak{C}$  (that comprises our solution space) is a nonincreasing function of the time adaptation step,  $n$ .

2. **Asymptotic Optimality.** If assumptions (a), (c) hold true then:

$$\lim_{n \rightarrow \infty} d(\mathbf{w}_{k,n}, \Omega_n) = 0, \quad \forall k \in \mathcal{N},$$

where  $d(\mathbf{w}_{k,n}, \Omega_n)$  is the distance of  $\mathbf{w}_{k,n}$  from  $\Omega_n$ . In other words, the distance of the obtained estimates, in each one of the nodes, from the intersection of the respective hyperslabs tends asymptotically to zero.

3. **Asymptotic Consensus.** It has been shown, [31], that in order to achieve asymptotic consensus, *i.e.*, [13]

$$\lim_{n \rightarrow \infty} \|\mathbf{w}_{k,n} - \mathbf{w}_{l,n}\| = 0, \quad \forall k, l \in \mathcal{N}, \quad (4.7)$$

the following must hold true:

$$\lim_{n \rightarrow \infty} [(I_{Nm} - \mathbf{B}\mathbf{B}^T)\underline{\mathbf{w}}_n] = \mathbf{0}_{Nm},$$

where the matrix  $\mathbf{B}$  is defined in Appendix A. The previous statement is true under assumptions (a), (c), (d).

4. **Strong Convergence.** If Assumptions (a), (c), (d), (e) hold true, then there exists  $\underline{\mathbf{w}}_O \in \mathcal{O}$  such that

$$\lim_{n \rightarrow \infty} \underline{\mathbf{w}}_n = \underline{\mathbf{w}}_O.$$

In other words, the sequence generated by (4.5) converges strongly to a point in  $\mathcal{O}$ .

*Proof.* The proof is given in Appendix B. □

**Remark 16.** Notice that asymptotic consensus is achieved despite the fact that the algorithm follows the diffusion optimization rationale. Loosely speaking, after a large number of iterations, the estimates of the nodes will lie close or will belong to the hyperslabs, which

implies that they will not be updated. This is a consequence of the asymptotic optimality property. Henceforth, the only operation taking place at each node is the combination of the estimates coming from the neighborhood. This reminds us of the consensus averaging iteration discussed in Chapter 2. So, if the weights,  $a_{k,l}$ , are properly chosen, then the nodes will reach asymptotic consensus. The consensus based LMS as well as the consensus based RLS, e.g., [97, 98, 123] achieve consensus in the mean, which implies that in a single realization of the algorithm there is no guarantee that the nodes will converge to the same estimate. These results are summarized in Table 4.1.

Table 4.1: Convergence Properties of Adaptive Distributed Algorithms

Algorithm	Consensus	Convergence
Algorithm 1	Asymptotic Consensus	Consensus Subspace
Diffusion LMS	Consensus in the Mean	Convergence in the Mean
Consensus LMS	Consensus in the Mean	Convergence in the Mean
Consensus RLS	Consensus in the Mean	Convergence in the Mean

## 4.3 Introducing Robustness to Cope with a Failure of Nodes

Consider a scenario, in which some of the nodes are damaged and the associated observations are very noisy<sup>2</sup>. In such cases, the use of loss functions, suggested in the framework of robust statistics, are more appropriate to cope with outliers. A popular cost function of this family is the Huber cost function, e.g., [70, 113], defined as<sup>3</sup>,

$$\tilde{\Theta}_{k,n}(\mathbf{w}) = \begin{cases} 0, & \text{if } |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \tilde{\gamma}_k, \\ \frac{1}{2} |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}|^2 - \frac{\tilde{\gamma}_k^2}{2}, & \text{if } \tilde{\gamma}_k < |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \tilde{c}_k, \\ \tilde{c}_k |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| - \frac{\tilde{\gamma}_k^2}{2} - \frac{\tilde{c}_k^2}{2}, & \text{otherwise.} \end{cases} \quad (4.8)$$

---

<sup>2</sup>We assume that the noise remains additive and white. However, its standard deviation becomes larger.

<sup>3</sup>It should be stressed out, that the cost function given in (4.8) is slightly modified compared to the classical Huber function. The difference is that in (4.8) a 0-th level set is introduced, which is determined by the parameter  $\tilde{\gamma}_k$ . The existence of 0-th level set is not in the classical Huber cost function.

The one dimensional version of it is illustrated in Fig. 4.2. The use of this function in the context of sensor networks has also been suggested in [113]. Let us take a closer look at (4.8). First of all, whenever  $|d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \tilde{\gamma}_k$ , we assume that our cost function scores a zero penalty and the non-negative, user-defined parameter,  $\tilde{\gamma}_k$ , defines the 0-th level set (property set) of the function. On the contrary, if  $\tilde{\gamma}_k < |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \tilde{c}_k$ , where  $\tilde{c}_k > \tilde{\gamma}_k$  is also a user defined parameter, then the estimate scores a non-zero penalty, with a square dependence on the error. Finally, in the case when  $|d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| > \tilde{c}_k$ , the measurements have probably occurred from a corrupted node and the cost function now changes to a linear dependence so as to treat it as an outlier.

In order to derive the new algorithm around the cost in (4.8), all that has to change, with respect to the algorithm which was previously developed, are the projection operators. Instead of projecting onto hyperslabs, one has to project onto halfspaces, denoted as  $H_{k,n}^-$ , which are formed by the intersections of the supporting hyperplanes (associated with the subgradients of the cost function) and the space where the solution lies, as already discussed in Chapter 3. Algebraically, this is done by the APSM formula, *i.e.*,

$$P_{H_{k,n}^-}(\mathbf{w}) = \begin{cases} \mathbf{w} - \frac{\tilde{\Theta}'_{k,n}(\mathbf{w})}{\|\tilde{\Theta}'_{k,n}(\mathbf{w})\|^2} \tilde{\Theta}'_{k,n}(\mathbf{w}), & \text{if } \tilde{\Theta}'_{k,n}(\mathbf{w}) \neq 0, \\ \mathbf{w}, & \text{otherwise.} \end{cases} \quad (4.9)$$

The subgradient  $\tilde{\Theta}'_{k,n}$  of the loss function is given by [126]:

$$\tilde{\Theta}'_{k,n}(\mathbf{w}) = \begin{cases} \mathbf{0}_m, & \text{if } |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \tilde{\gamma}_k, \\ \kappa(d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n})(-\mathbf{u}_{k,n}) \text{ with } \kappa \in [0, 1], & \text{if } |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| = \tilde{\gamma}_k, \\ (d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n})(-\mathbf{u}_{k,n}), & \text{if } \tilde{\gamma}_k < |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \tilde{c}_k, \\ \tilde{c}_k \text{sgn}(d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n})(-\mathbf{u}_{k,n}), & \text{otherwise,} \end{cases}$$

where  $\text{sgn}(\cdot)$  denotes the sign function.

We can also include the extra projection step, described in the previous section, by introducing a modified version of (4.8) and following a similar rationale as in Section 4.1. However, instead of projecting the aggregate  $\phi_{k,n}$  onto an external hyperslab, we project it onto a halfspace that is generated by a properly modified cost function (Fig. 4.2). To be

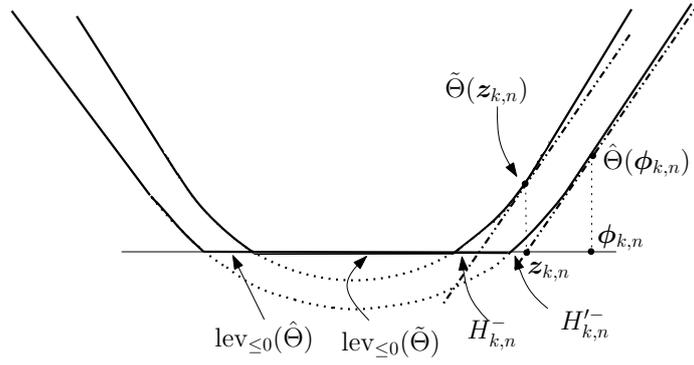


Figure 4.2: Illustration of the cost functions  $\tilde{\Theta}(\cdot)$ ,  $\hat{\Theta}(\cdot)$ . The aggregate,  $\phi_{k,n}$ , is projected onto  $H'_{k,n}$ , which is the intersection of the hyperplane associated with the subgradient at  $\hat{\Theta}(\phi_{k,n})$  with the space  $\mathbb{R}^m$ , to provide  $z_{k,n}$ . In the sequel,  $z_{k,n}$  is used into the adaptation step.

more specific, we define

$$\hat{\Theta}_{k,n}(\mathbf{w}) = \begin{cases} 0, & \text{if } |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \hat{\gamma}_k, \\ \frac{1}{2} |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}|^2 - \frac{\hat{\gamma}_k^2}{2}, & \text{if } \hat{\gamma}_k < |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| \leq \hat{c}_k, \\ \hat{c}_k |d_{k,n} - \mathbf{w}^T \mathbf{u}_{k,n}| - \frac{\hat{\gamma}_k^2}{2} - \frac{\hat{c}_k^2}{2}, & \text{otherwise,} \end{cases}$$

where  $\hat{c}_k > \hat{\gamma}_k$  and  $\hat{\gamma}_k > \tilde{\gamma}_k$ . The latter condition is required to guarantee consensus. The projection of an arbitrary point onto the halfspace  $H'_{k,n}$  is similar to (4.9) and the algorithm is similar to the one given in (4.2a)-(4.2c) with the slight difference that the involved hyperslabs have been replaced by halfspaces. More specifically, the following recursion occurs at each node.

**Algorithm 2:**

$$\phi_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{w}_{l,n}, \quad (4.10a)$$

$$\mathbf{z}_{k,n} = P_{H'_{k,n}}(\phi_{k,n}), \quad (4.10b)$$

$$\mathbf{w}_{k,n+1} = \mathbf{z}_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{H'_{k,j}}(\mathbf{z}_{k,n}) - \mathbf{z}_{k,n} \right), \quad (4.10c)$$

where the parameters  $\omega_{k,j}$ ,  $\mathcal{J}_n$  and  $\mu_{k,n}$  are defined similarly as in Algorithm 1.

As in the case of the hyperslab projection algorithm, the algorithmic scheme, which builds the property sets around the Huber loss function, enjoys monotonicity, asymptotic optimality, asymptotic consensus and strong convergence. Obviously, the assumptions under which the algorithm enjoys these convergence properties have to change compared to the algorithm of the previous section. More specifically,  $\Omega$  now becomes  $\Omega' = \bigcap_{n \geq n_0} \Omega'_n \neq \emptyset$  where  $\Omega'_n = \bigcap_{k \in \mathcal{N}} \bigcap_{j \in \mathcal{J}_n} H_{k,j}^-$ . Furthermore, the stepsizes must lie inside the interval with upper bound determined as in the previous algorithm, with the difference that  $\mathcal{M}_{k,n}$  is replaced by a parameter determined by the projection operators onto the respective halfspaces. Finally, the assumption regarding the consensus is now  $\hat{\gamma}_k > \tilde{\gamma}_k$ . Such a choice guarantees that  $\text{lev}_{\leq 0} \tilde{\Theta}_{k,n} \subset \text{lev}_{\leq 0} \hat{\Theta}_{k,n}$ , which is required in order to prove consensus (see Appendix B).

**Remark 17.** *Note that although the property sets, associated with the loss function used in this section, are quite different compared to the hyperslabs used before, both algorithms, i.e., the one built around the Huber loss function and the one given in (4.2a)-(4.2c), are of the same form. The only difference lies in the projection operators. Moreover, the theoretical analysis is exactly the same. All that matters is the property of convexity. This is a major advantage of the methodology behind this set theoretic approach.*

## 4.4 Numerical Examples

In this section, we present simulation results in order to study the comparative performance of the developed algorithms with respect to previously reported schemes. The general framework of our experiments is the system identification task in ad-hoc networks. A linear system described by (4.1) is adopted and our goal is to estimate the unknown vector  $\mathbf{w}_*$  using the measurements  $d_{k,n}$ ,  $\mathbf{u}_{k,n}$ . The components of the regression vectors, i.e.,  $\mathbf{u}_{k,n} = [u_{k,n}, \dots, u_{k,n-m+1}]^T$ , are generated according to

$$u_{k,n} = \psi_k u_{k,n-1} + \chi_{k,n}, \quad \forall k \in \mathcal{N},$$

where  $\psi_k \in (0, \psi_u)$  and it is distributed according to the uniform distribution and  $\psi_u$  is a parameter that we alter throughout our experiments in order to investigate the behaviour of the algorithms for cases where the regressors are strongly or weakly correlated. Finally,  $\chi_{k,n}$

is Gaussian with unit variance. The standard deviation of the noise  $v_{k,n}$ , which is assumed to be white and Gaussian, equals to  $\sigma_k = b_k \sigma_u$  where  $b_k \in (0, 1)$  under the uniform distribution and  $\sigma_u$  is user-defined and will change throughout our experiments. In order to construct the network, the following strategy has been followed. A certain node, say  $k$ , is connected to any other node with probability equal to 0.3, and it is connected to itself with probability 1. Additionally, the combination coefficient are chosen according to the Metropolis rule. Finally, the adopted performance metrics used are:

- Mean Square Error (MSE), which is defined:  $\sum_{k=1}^N \frac{(d_{k,n} - \mathbf{u}_{k,n}^T \mathbf{w}_{k,n})^2}{N}$ ,
- Mean Square Deviation (MSD), defined as:  $\sum_{k=1}^N \frac{\|\mathbf{w}_* - \mathbf{w}_{k,n}\|^2}{N}$ .

The experiments are averaged over 100 realizations for smoothing purposes.

We compare the proposed algorithm 1 with a) the adapt-combine LMS of [28] (A-C LMS), b) the consensus based LMS [98] (D-LMS), and c) with the Adaptive Projected Subgradient Method in Diffusion networks (APSMd) proposed in [31]. In the first experiment, we consider an ad-hoc network with  $N = 20$  nodes and the unknown vector  $\mathbf{w}_*$  is of dimension  $m = 60$ . The noise profile for this network is obtained with  $\sigma_u = 10^{-3}$ ,  $\psi_u = 0.5$ . The parameter  $\epsilon = \sqrt{2}\sigma_k$  for both the proposed algorithm 1 and for the APSMd and the parameter  $\epsilon'_k = 2\epsilon_k$ . Furthermore, the number of hyperslabs onto which we project, in each step, is  $q = 8$ , whereas the convex combination multipliers are all equal, *i.e.*,  $\omega_{k,n} = \frac{1}{q}$ , and we let  $\mu_{k,n} = 1$ ,  $k = 1, \dots, N$ , for algorithm 1 and APSMd. For the A-C LMS, the stepsize equals to  $\mu_{LMS} = 0.01$ , whereas for D-LMS  $\mu_{D-LMS} = 0.006$ . The stepsizes in the LMS-based algorithms are chosen so that the algorithms reach the same steady state error floor in the MSE sense. Throughout our experiments, if we let  $P_{S'_{k,n}}$  be the identity mapping, it turns out that the proposed algorithm 1 and the APSMd have similar performance. Hence, as it can be seen in Figs. 4.3.a, 4.3.b, the extra projection onto the hyperslab  $S'_{k,n}$ , which adds an  $O(m)$  expense on the computational complexity of the algorithm for each node, enhances the results compared to the other algorithms, as it accelerates the convergence speed for the same error floor. Moreover, the proposed algorithm 1 outperforms the LMS-based algorithms and the APSMd, as it achieves a better steady state error floor in the MSD sense. Nevertheless, compared to the LMS-based algorithms, the proposed algorithm 1 and the APSMd require knowledge on the noise statistics, *i.e.*,  $\sigma_k$ ,  $\forall k \in \mathcal{N}$ . Moreover, for  $q > 1$ , the projection-

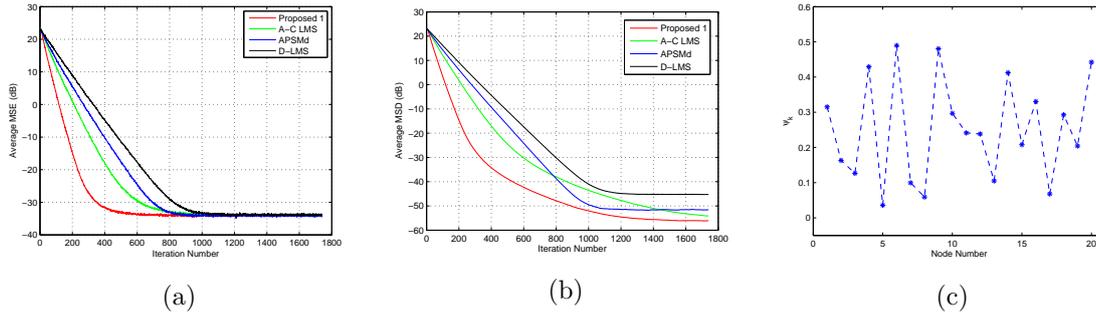


Figure 4.3: (a) MSE for experiment 1. (b) MSD for experiment 1. (c) The statistics of the network’s regressors.

based algorithms require some extra memory in order to store past data. We would also like to mention, that through experiments, we observed small discrepancies between the steady state performances of the nodes, despite the fact that the noise statistics may be different. This is also known as equalization property and it is a common effect met in diffusion based algorithms, *e.g.*, [94, 98].

In the second experiment, (Figs. 4.4.a, 4.4.b), the parameters remain the same as in the previous one. However, we choose a larger  $\psi_u$ , namely 0.9, in order to compare the algorithms in a more correlated environment. Furthermore, we alter the step-sizes for the LMS-based algorithms. More specifically, the values  $\mu_{LMS} = 0.007$  and  $\mu_{D-LMS} = 0.002$  were chosen in a similar philosophy as in the previous experiment. As it is expected, the LMS-based algorithms result in worse performance compared to the previous example of less correlated signals, something that holds true also in the classical LMS case [120]. This performance trend can be seen both in the MSE and the MSD curves, as algorithm 1 outperforms significantly the LMS-based algorithms. Moreover, as it was the case in the first experiment, it can be seen that the extra projection step accelerates the convergence speed of proposed algorithm 1 compared to the APSMd.

The scenario in the third experiment (Figs. 4.5.a, 4.5.b) is the same as the one in the first experiment, but now after a number of iterations there is a sudden change in the channel. At time instant  $n = 1800$ ,  $w^*$  changes to  $-w_*$ . This is a popular experiment in adaptive filter theory in order to test the tracking performance of the algorithm. As it is by now well established, fast convergence speed does not necessarily guarantee a good tracking performance [120]. It can be readily seen that the proposed algorithm 1 shows a

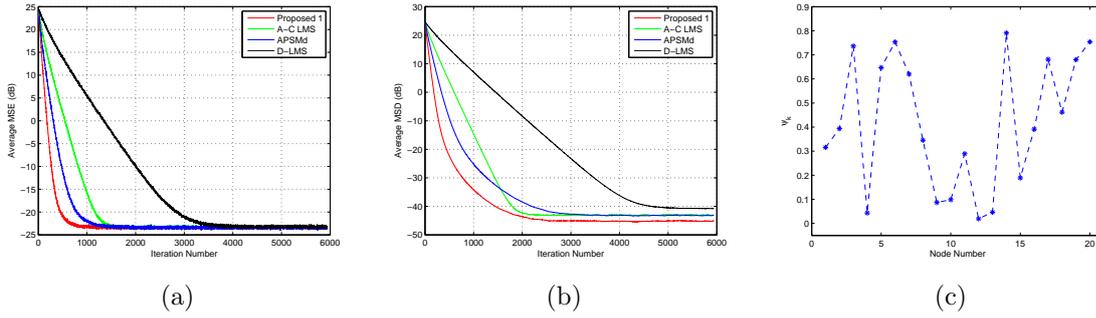


Figure 4.4: (a) MSE for experiment 2. (b) MSD for experiment 2. (c) The statistics of the network's regressors.

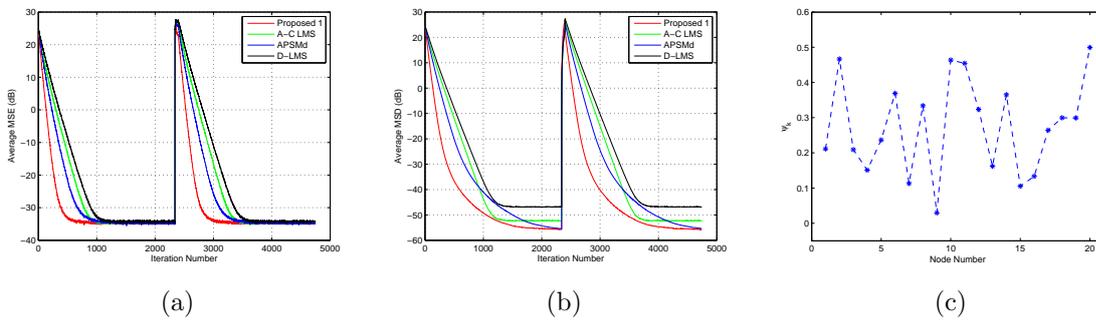


Figure 4.5: (a) MSE for experiment 3. (b) MSD for experiment 3. (c) The statistics of the network's regressors.

large capacity for tracking ability, when a sudden change in the channel takes place. More specifically, until the time instant at which  $\mathbf{w}_*$  changes, the performance of the algorithms coincides with that of Figs. 1.3.a, 1.3.b. After the sudden change, the proposed algorithm 1 exhibits the best tracking performance to the common steady state error floor.

Next, the algorithms are compared in a scenario where a subset of the nodes is malfunctioning. The number of nodes is chosen equal to  $N = 10$  and the vector to be estimated is of dimension  $m = 30$ . Five of the nodes are malfunctioning, so for them  $\sigma'_k = 40\sigma_k$ , and  $\sigma_u = 0.01$ . For algorithm 2,  $\tilde{\gamma}_k = 4\sigma_k$ ,  $\hat{\gamma}_k = 2\tilde{\gamma}_k$ ,  $\tilde{c}_k = 5\tilde{\gamma}_k$ ,  $\hat{c}_k = 5\hat{\gamma}_k$ . Finally, the rest of the parameters are  $\psi_u = 0.5$ ,  $q = 8$  in the projection based algorithms,  $\mu_{LMS} = 0.08$  and  $\mu_{D-LMS} = 0.01$ . The stepsizes, as in the previous experiments, were chosen so that the algorithms converge to the same steady state error floor, in the MSE sense.

From Fig. 4.6.a, it can be observed that the projection based algorithms converge faster to the common error floor. Furthermore, the proposed algorithm 2 and the APSMd have a similar convergence speed. In Fig. 4.6.c, the average MSE, taken over the healthy nodes only,

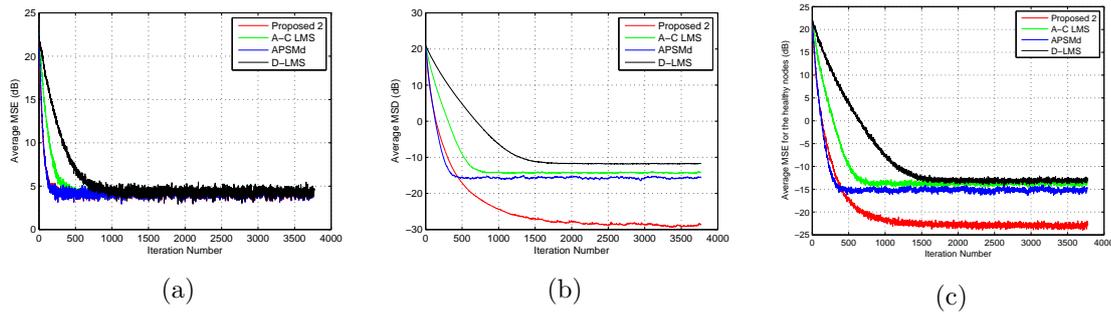


Figure 4.6: (a) MSE for experiment 4 by considering all the nodes of the network. (b) MSD for experiment 4. (c) Average MSE computed over the healthy nodes.

is given. It can be seen that the proposed algorithm 2 exhibits a significantly better steady state error floor compared to the other algorithms. The reason that the proposed algorithm 2 achieves this improved error floor, whereas in Fig. 4.6.a the algorithms converge to the same one, is a consequence of the fact that by taking into consideration the malfunctioning nodes, the noise dominates the average MSE. Furthermore, Fig. 4.6.b demonstrates that the proposed algorithm 2 also achieves a significantly improved steady state error floor in the MSD sense, for the whole network. This implies that the estimate occurring is closer to  $\mathbf{w}_*$ . The LMS-based algorithms result in the worst performance compared to the projection based algorithms, which is expected as in the APSMd and the proposed algorithm 2, robust cost functions are employed for minimization. However, the proposed algorithm 2 results in the lowest steady state error floor, since the Huber cost function accounts for the outliers, that occur due to the malfunctioning nodes, in a more focused way, compared to the hyperslabs used in APSMd. Finally, the fact that the proposed algorithm 2 converges slightly slower than APSMd, which can be seen from the curves of Fig. 4.6.b and Fig. 4.6.c, is not a surprising result and it is due to the fact that in the case of hyperslabs the level set is reached with a single projection, whereas in the proposed algorithm 2, the corresponding level set is approached via a sequence of projections onto halfspaces that contain it.

# Appendix A

## Consensus Matrix and the Consensus Subspace

Gathering all the coefficients  $a_{k,l}$  in a matrix, we define the combination matrix  $\mathbf{A}$ , the  $k, l$ -th component of which equals to  $a_{k,l}$ . The  $Nm \times Nm$  consensus matrix, which will be used in the theoretical analysis of the algorithm, is given by  $\mathbf{P} = \mathbf{A} \otimes \mathbf{I}_m$ . Now, let us give the definition of the consensus subspace  $\mathcal{O}$ . This linear subspace has definition:  $\mathcal{O} := \{\underline{\mathbf{w}} \in \mathbb{R}^{Nm} : \underline{\mathbf{w}} = [\mathbf{w}^T, \dots, \mathbf{w}^T]^T, \mathbf{w} \in \mathbb{R}^m\}$ , and its dimension equals to  $m$ . Some very useful properties of the consensus matrix as well as the consensus subspace, which will be used in the derivation of the main theorem, are [31]:

1.  $\mathbf{P}\mathbf{1}_{Nm} = \mathbf{1}_{Nm}$ .
2.  $\|\mathbf{P}\| = 1$ .
3. Any consensus matrix  $\mathbf{P}$  can be decomposed as

$$\mathbf{P} = \mathbf{X} + \mathbf{B}\mathbf{B}^T,$$

where  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_m]$  is an  $Nm \times m$  matrix, and  $\mathbf{b}_k = \frac{(\mathbf{1}_N \otimes \mathbf{e}_k)}{\sqrt{N}}$ ,  $\mathbf{e}_k$  is an  $m \times 1$  vector of zeros except the  $k$ -th entry, which is one and  $\mathbf{X}$  is an  $Nm \times Nm$  matrix for which it holds that  $\|\mathbf{X}\| < 1$ .

4.  $\mathbf{P}\underline{\check{\mathbf{w}}} = \underline{\check{\mathbf{w}}}, \forall \underline{\check{\mathbf{w}}} \in \mathcal{O}$ .

5. The vectors  $\mathbf{b}_k$ ,  $k = 1, \dots, m$  constitute a basis for  $\mathcal{O}$ . The projection of a vector,  $\underline{\mathbf{w}} \in \mathbb{R}^{Nm}$ , onto this linear subspace is given by  $P_{\mathcal{O}}(\underline{\mathbf{w}}) := \mathbf{B}\mathbf{B}^T\underline{\mathbf{w}}$ ,  $\forall \underline{\mathbf{w}} \in \mathbb{R}^{Nm}$ .

# Appendix B

## Proof of Theorem 1

In this appendix, the proof of the main Theorem will be presented.

### B.1 Proof of Theorem 1.1

Assume that for a fixed node, say  $k$ , at time instant  $n$  we have estimated  $\mathbf{z}_{k,n}$ . We define the cost function, for any  $\mathbf{w} \in \mathbb{R}^{m-1}$

$$\Theta_{k,n}(\mathbf{w}) = \begin{cases} \sum_{j \in \mathcal{J}_n} \frac{\omega_{k,j} d(\mathbf{z}_{k,n}, S_{k,j})}{\sum_{l \in \mathcal{J}_n} \omega_{k,l} d(\mathbf{z}_{k,n}, S_{k,l})} d(\mathbf{w}, S_{k,j}), & \text{if } \mathbf{z}_{k,n} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j}, \\ 0, & \text{otherwise.} \end{cases}$$

We also define  $L_n := \sum_{j \in \mathcal{J}_n} \omega_{k,j} d(\mathbf{z}_{k,n}, S_{k,j})$ , for which, by definition,  $L_n \neq 0$  if  $\mathbf{z}_{k,n} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j}$ . The previous statement holds true obviously because if  $\mathbf{z}_{k,n} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j}$  there exists  $j_0 \in \mathcal{J}_n$  for which  $d(\mathbf{z}_{k,n}, S_{k,j_0}) \neq 0$  hence  $\sum_{j \in \mathcal{J}_n} \omega_{k,j} d(\mathbf{z}_{k,n}, S_{k,j}) > 0$ .

It can be seen that this cost function is convex, continuous and subdifferentiable. Its subgradient is given by [124]

$$\Theta'_{k,n}(\mathbf{w}) = \begin{cases} \frac{1}{L_n} \sum_{j \in \mathcal{J}_n} \omega_{k,j} d(\mathbf{z}_{k,n}, S_{k,j}) d'(\mathbf{w}, S_{k,j}), & \text{if } \mathbf{z}_{k,n} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j}, \\ 0, & \text{otherwise,} \end{cases} \quad (\text{B.1})$$

---

<sup>1</sup>The proof when we use projections onto halfspaces, which are involved when the Huber cost function is employed, follows similar steps. All one has to do is to modify the cost function replacing projections onto hyperslabs with projections onto halfspaces. However, the proofs rely on the properties of metric projections and not on their specific form.

## Proof of Theorem 1

---

where  $d'(\mathbf{w}, S_{k,j}) \in \partial d(\mathbf{w}, S_{k,j})$ , and [65]

$$\partial d(\mathbf{w}, S_{k,j}) = \begin{cases} \frac{\mathbf{w} - P_{S_{k,j}}(\mathbf{w})}{d(\mathbf{w}, S_{k,j}(\mathbf{w}))}, & \text{if } \mathbf{w} \notin S_{k,j}, \\ \text{a certain subset of } B_{[\mathbf{0}_m, 1]} \text{ which contains } \mathbf{0}_m, & \text{otherwise.} \end{cases}$$

Additionally, if  $\mathbf{w} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j}$  and  $\mathbf{z}_{k,n} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j}$ , from (B.1) a choice of a subgradient is:

$$\Theta'_{k,n}(\mathbf{w}) = \frac{1}{L_n} \sum_{\{j \in \mathcal{J}_n: \mathbf{w} \notin S_{k,j}\}} \omega_{k,j} d(\mathbf{z}_{k,n}, S_{k,j}) \frac{\mathbf{w} - P_{S_{k,j}}(\mathbf{w})}{d(\mathbf{w}, S_{k,j})}.$$

Finally, if  $\mathbf{z}_{k,n} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j}$ ,

$$\begin{aligned} \Theta'_{k,n}(\mathbf{z}_{k,n}) &= \frac{1}{L_n} \sum_{\{j \in \mathcal{J}_n: \mathbf{z}_{k,n} \notin S_{k,j}\}} \omega_{k,j} (\mathbf{z}_{k,n} - P_{S_{k,j}}(\mathbf{z}_{k,n})) \\ &= \frac{1}{L_n} \sum_{j \in \mathcal{J}_n} \omega_{k,j} (\mathbf{z}_{k,n} - P_{S_{k,j}}(\mathbf{z}_{k,n})). \end{aligned} \tag{B.2}$$

The last equality is true, due to the fact that if there exists  $j_0 \in \mathcal{J}_n$  such that  $\mathbf{z}_{k,n} \in S_{k,j_0}$ , then  $\mathbf{z}_{k,n} = P_{S_{k,j_0}}(\mathbf{z}_{k,n})$ .

Now, recall the definition of  $\mathcal{M}_{k,n}$  given in (4.3). We define

$$\lambda_{k,n} = \frac{\mu_{k,n}}{\mathcal{M}_{k,n}}. \tag{B.3}$$

One should notice here that  $\lambda_{k,n} \in (0, 2)$  under assumption (b). In the case where  $\sum_{j \in \mathcal{J}_n} \omega_{k,j} (\mathbf{z}_{k,n} - P_{S_{k,j}}(\mathbf{z}_{k,n})) \neq \mathbf{0}_m$ , if we go back to the recursion given in (4.2c) and

combining (B.3) with (B.1) and (B.2) we get

$$\begin{aligned}
 \mathbf{w}_{k,n+1} &= \mathbf{z}_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}(\mathbf{z}_{k,n}) - \mathbf{z}_{k,n} \right) \\
 &= \mathbf{z}_{k,n} + \lambda_{k,n} \frac{\sum_{j \in \mathcal{J}_n} \omega_{k,j} d^2(\mathbf{z}_{k,n}, S_{k,j})}{\left\| \sum_{j \in \mathcal{J}_n} \omega_{k,j} (\mathbf{z}_{k,n} - P_{S_{k,j}}(\mathbf{z}_{k,n})) \right\|^2} \\
 &\quad \times \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} (P_{S_{k,j}}(\mathbf{z}_{k,n}) - \mathbf{z}_{k,n}) \right) \\
 &= \mathbf{z}_{k,n} + \lambda_{k,n} \frac{\frac{1}{L_n} \sum_{j \in \mathcal{J}_n} \omega_{k,j} d^2(\mathbf{z}_{k,n}, S_{k,j})}{\frac{1}{L_n^2} \left\| \sum_{j \in \mathcal{J}_n} \omega_{k,j} (\mathbf{z}_{k,n} - P_{S_{k,j}}(\mathbf{z}_{k,n})) \right\|^2} \frac{1}{L_n} \\
 &\quad \times \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} (P_{S_{k,j}}(\mathbf{z}_{k,n}) - \mathbf{z}_{k,n}) \right) = \mathbf{z}_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\mathbf{z}_{k,n})}{\left\| \Theta'_{k,n}(\mathbf{z}_{k,n}) \right\|^2} \Theta'_{k,n}(\mathbf{z}_{k,n}). \quad (\text{B.4})
 \end{aligned}$$

Equation (B.4) is a slight modification of the Adaptive Projected Subgradient Method. Following similar steps as in [124], and under assumption (a), then it can be shown that:

$\mathbf{z}_{k,n} \notin \bigcap_{j \in \mathcal{J}_n} S_{k,j} \Leftrightarrow \|\Theta'_{k,n}(\mathbf{z}_{k,n})\| \neq 0, \forall k \in \mathcal{N}, \forall n \geq n_0$ . So for the whole network we have

$$\underline{\mathbf{w}}_{n+1} = \begin{bmatrix} \mathbf{z}_{1,n} - \lambda_{1,n} \frac{\Theta_{1,n}(\mathbf{z}_{1,n})}{\|\Theta'_{1,n}(\mathbf{z}_{1,n})\|^2} \Theta'_{1,n}(\mathbf{z}_{1,n}) \\ \mathbf{z}_{2,n} - \lambda_{2,n} \frac{\Theta_{2,n}(\mathbf{z}_{2,n})}{\|\Theta'_{2,n}(\mathbf{z}_{2,n})\|^2} \Theta'_{2,n}(\mathbf{z}_{2,n}) \\ \vdots \\ \mathbf{z}_{N,n} - \lambda_{N,n} \frac{\Theta_{N,n}(\mathbf{z}_{N,n})}{\|\Theta'_{N,n}(\mathbf{z}_{N,n})\|^2} \Theta'_{N,n}(\mathbf{z}_{N,n}) \end{bmatrix} \Rightarrow \quad (\text{B.5})$$

$$\begin{aligned} \|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\|^2 &= \left\| \begin{bmatrix} \mathbf{z}_{1,n} - \lambda_{1,n} \frac{\Theta_{1,n}(\mathbf{z}_{1,n})}{\|\Theta'_{1,n}(\mathbf{z}_{1,n})\|^2} \Theta'_{1,n}(\mathbf{z}_{1,n}) \\ \mathbf{z}_{2,n} - \lambda_{2,n} \frac{\Theta_{2,n}(\mathbf{z}_{2,n})}{\|\Theta'_{2,n}(\mathbf{z}_{2,n})\|^2} \Theta'_{2,n}(\mathbf{z}_{2,n}) \\ \vdots \\ \mathbf{z}_{N,n} - \lambda_{N,n} \frac{\Theta_{N,n}(\mathbf{z}_{N,n})}{\|\Theta'_{N,n}(\mathbf{z}_{N,n})\|^2} \Theta'_{N,n}(\mathbf{z}_{N,n}) \end{bmatrix} - \underline{\mathbf{w}}_* \right\|^2 \\ &= \left\| \begin{bmatrix} \mathbf{z}_{1,n} - \hat{\mathbf{w}}_* \\ \mathbf{z}_{2,n} - \hat{\mathbf{w}}_* \\ \vdots \\ \mathbf{z}_{N,n} - \hat{\mathbf{w}}_* \end{bmatrix} \right\|^2 + \sum_{k=1}^N \lambda_{k,n}^2 \frac{(\Theta_{k,n}(\mathbf{z}_{k,n}))^2}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^4} \|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2 \\ &\quad - 2 \sum_{k=1}^N \lambda_{k,n} \frac{\Theta_{k,n}(\mathbf{z}_{k,n}) \Theta_{k,n}^T(\mathbf{z}_{k,n}) (\mathbf{z}_{k,n} - \hat{\mathbf{w}}_*)}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2}. \end{aligned} \quad (\text{B.6})$$

Notice here that  $P_{S'_{k,n}}(\hat{\mathbf{w}}_*) = \hat{\mathbf{w}}_*, \forall k \in \mathcal{N}$ . This argument is true since  $\hat{\mathbf{w}}_* \in \Omega_n \Rightarrow \hat{\mathbf{w}}_* \in S_{k,n} \subset S'_{k,n} \Rightarrow \hat{\mathbf{w}}_* \in S'_{k,n}, \forall k \in \mathcal{N}$ . Furthermore, one basic property of the projection operator onto closed convex sets [129] states that  $\|P_{S'_{k,n}}(\phi_{k,n}) - P_{S'_{k,n}}(\hat{\mathbf{w}}_*)\| \leq \|\phi_{k,n} - \hat{\mathbf{w}}_*\|, \forall k \in \mathcal{N}$ . Hence, recalling the properties of the matrix  $\mathbf{P}$  we have:

$$\begin{aligned} \left\| \begin{bmatrix} \mathbf{z}_{1,n} - \hat{\mathbf{w}}_* \\ \mathbf{z}_{2,n} - \hat{\mathbf{w}}_* \\ \vdots \\ \mathbf{z}_{N,n} - \hat{\mathbf{w}}_* \end{bmatrix} \right\|^2 &= \left\| \begin{bmatrix} P_{S'_{1,n}}(\phi_{1,n}) - P_{S'_{1,n}}(\hat{\mathbf{w}}_*) \\ P_{S'_{2,n}}(\phi_{2,n}) - P_{S'_{2,n}}(\hat{\mathbf{w}}_*) \\ \vdots \\ P_{S'_{N,n}}(\phi_{N,n}) - P_{S'_{N,n}}(\hat{\mathbf{w}}_*) \end{bmatrix} \right\|^2 \leq \left\| \begin{bmatrix} \phi_{1,n} - \hat{\mathbf{w}}_* \\ \phi_{2,n} - \hat{\mathbf{w}}_* \\ \vdots \\ \phi_{N,n} - \hat{\mathbf{w}}_* \end{bmatrix} \right\|^2 \\ &= \|\mathbf{P}\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2 = \|\mathbf{P}\underline{\mathbf{w}}_n - \mathbf{P}\underline{\mathbf{w}}_*\|^2 \\ &\leq \|\mathbf{P}\|^2 \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2 = \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2, \end{aligned} \quad (\text{B.7})$$

Moreover, from the definition of the subgradient  $\Theta_{k,n}^T(\mathbf{z}_{k,n})(\mathbf{z}_{k,n} - \hat{\mathbf{w}}_*) \geq \Theta_{k,n}(\mathbf{z}_{k,n}) -$

$\Theta_{k,n}(\hat{\mathbf{w}}_*) = \Theta_{k,n}(\mathbf{z}_{k,n}) \geq 0$ , where we have used the fact that  $\Theta_{k,n}(\hat{\mathbf{w}}_*) = 0$ , which holds by the definition of the cost function, since  $\hat{\mathbf{w}}_* \in \bigcap_{j \in \mathcal{J}_n} S_{k,j}$ . Combining the last argument with (E.12) and (B.7) we have

$$\begin{aligned}
\|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\|^2 &\leq \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2 + \sum_{k=1}^N \lambda_{k,n}^2 \frac{(\Theta_{k,n}(\mathbf{z}_{k,n}))^2}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2} - 2 \sum_{k=1}^N \lambda_{k,n} \frac{(\Theta_{k,n}(\mathbf{z}_{k,n}))^2}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2} \\
&= \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2 - \sum_{k=1}^N \lambda_{k,n} (2 - \lambda_{k,n}) \frac{(\Theta_{k,n}(\mathbf{z}_{k,n}))^2}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2} \\
&\Leftrightarrow \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2 - \|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\|^2 \\
&\geq \sum_{k=1}^N \lambda_{k,n} (2 - \lambda_{k,n}) \frac{(\Theta_{k,n}(\mathbf{z}_{k,n}))^2}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2} \geq 0.
\end{aligned} \tag{B.8}$$

Hence, we conclude that  $\|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\| \leq \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|$ . This completes the proof of Theorem 1.1.

**Remark 18.** Here, we would like to point out that monotonicity also holds in cases where the subgradients of some nodes equal to zero. Assume, without loss of generality, that the subgradient of node  $l$ , at time instance  $n$ , is zero, which under assumption (a) implies that  $\mathbf{z}_{l,n} \in \bigcap_{j \in \mathcal{J}_n} S_{l,j}$ . Then, from [124] and if assumption (a) holds true, it can be proved that the recursion for this node is  $\mathbf{w}_{l,n} = \mathbf{z}_{l,n}$ . Loosely speaking, the second term of the right hand side in (B.4) is omitted. So, following exactly similar steps as in the previous proof, (B.8) becomes  $\|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\| - \|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\| \geq \sum_{k \in \mathcal{N} \setminus l} \lambda_{k,n} (2 - \lambda_{k,n}) \frac{(\Theta_{k,n}(\mathbf{z}_{k,n}))^2}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2} \geq 0$ .

## B.2 Proof of Theorem 1.2

We want to show that  $\lim_{n \rightarrow \infty} d(\mathbf{w}_{k,n}, \Omega_n) = 0$ . The sequence  $\|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2$  is bounded and monotonically decreasing, hence it converges. So it is a Cauchy sequence, from which we obtain that

$$\|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|^2 - \|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\|^2 \rightarrow 0, \quad n \rightarrow \infty. \tag{B.9}$$

## Proof of Theorem 1

---

So from (B.8), (B.9)

$$\lim_{n \rightarrow \infty} \lambda_{k,n} (2 - \lambda_{k,n}) \frac{(\Theta_{k,n}(\mathbf{z}_{k,n}))^2}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2} = 0, \forall k \in \mathcal{N}. \quad (\text{B.10})$$

Following similar steps as in [82] it can be proved that  $\|\Theta'_{k,n}(\mathbf{z}_{k,n})\| \leq 1, \forall k \in \mathcal{N}, \forall n \in \mathbb{Z}$ . Under assumption (c) and if we take into consideration (B.10), we have that

$$\varepsilon_1^2 \Theta_{k,n}(\mathbf{z}_{k,n}) \leq \frac{\lambda_{k,n}(2 - \lambda_{k,n})}{\|\Theta'_{k,n}(\mathbf{z}_{k,n})\|^2} \Theta_{k,n}(\mathbf{z}_{k,n}) \rightarrow 0, \forall k \in \mathcal{N}.$$

From the last equation we have that

$$\lim_{n \rightarrow \infty} \Theta_{k,n}(\mathbf{z}_{k,n}) = 0, \forall k \in \mathcal{N}. \quad (\text{B.11})$$

Now, if we go back to the recursion given in equation (B.5) and combine it with (B.10) we obtain

$$\lim_{n \rightarrow \infty} \|\mathbf{w}_{k,n+1} - \mathbf{z}_{k,n}\| = 0, \forall k \in \mathcal{N}. \quad (\text{B.12})$$

Let us assume that  $\mathbf{v}$  is an arbitrary point that belongs to  $\Omega(n)$ . We have that  $\|\mathbf{w}_{k,n+1} - \mathbf{v}\| \leq \|\mathbf{w}_{k,n+1} - \mathbf{z}_{k,n}\| + \|\mathbf{z}_{k,n} - \mathbf{v}\|$ , where this holds due to the triangle inequality. Therefore,

$$\begin{aligned} \inf_{\mathbf{v} \in \Omega_n} \|\mathbf{w}_{k,n+1} - \mathbf{v}\| &\leq \|\mathbf{w}_{k,n+1} - \mathbf{z}_{k,n}\| + \inf_{\mathbf{v} \in \Omega_n} \|\mathbf{z}_{k,n} - \mathbf{v}\| \\ \Leftrightarrow \text{d}(\mathbf{w}_{k,n}, \Omega_n) &\leq \|\mathbf{w}_{k,n} - \mathbf{z}_{k,n}\| + \text{d}(\mathbf{z}_{k,n}, \Omega_n). \end{aligned} \quad (\text{B.13})$$

Following the same steps as in [124] and taking into consideration (I.10), it can be proved that

$$\lim_{n \rightarrow \infty} \text{d}(\mathbf{z}_{k,n}, \Omega_n) = 0. \quad (\text{B.14})$$

Taking limits into equation (B.13), and combining (B.12), (B.14) we conclude that:

$$\lim_{n \rightarrow \infty} \text{d}(\mathbf{w}_{k,n+1}, \Omega_n) = 0.$$

## B.3 Proof of Theorem 1.3

First we need to prove the following claim

**Claim 1.** *Assume that  $\epsilon'_k > \epsilon_k$  and that (a), (c) hold true. Then, there exists  $n_2$  such that  $\phi_{k,n} \in S'_{k,n}, \forall n \geq n_2, \forall k \in \mathcal{N}$ .*

*Proof.* Since  $\epsilon'_k > \epsilon_k$ , for any vector  $\mathbf{w}$  on the boundary of  $S'_{k,n}$ , there exists  $\varepsilon_2$ , which depends on the choice of  $\epsilon'_k$  and  $\epsilon_k$ , such that  $d(\mathbf{w}, S_{k,n}) > \varepsilon_2$ . We have shown that  $d(\mathbf{z}_{k,n}, \Omega_n) \rightarrow 0, n \rightarrow \infty$  under assumptions (a), (c). However, since by definition  $\Omega_n \subseteq S_{k,n}$  we have that  $d(\mathbf{z}_{k,n}, S_{k,n}) \rightarrow 0, n \rightarrow \infty$ . Due to the last argument, there exists  $n_2$  such that

$$d(\mathbf{z}_{k,n}, S_{k,n}) \leq \frac{\varepsilon_2}{2}, \forall n \geq n_2. \quad (\text{B.15})$$

However, if  $\exists n \geq n_2$  such that  $\phi_{k,n} \notin S'_{k,n}$  then  $\mathbf{z}_{k,n}$  will lie on the boundary of  $S'_{k,n}$ , as it is the projection of  $\phi_{k,n}$  onto it. Hence  $d(\mathbf{z}_{k,n}, S_{k,n+1}) > \varepsilon_2$  which clearly contradicts equation (B.15). Thus our claim holds true.  $\square$

The fact that  $\phi_{k,n} \in S'_{k,n}$ , after some iterations, implies that  $\mathbf{z}_{k,n} = P_{S'_{k,n}}(\phi_{k,n}) = \phi_{k,n}, \forall n \geq n_2$ . Recalling (B.5) we have  $\forall n \geq n_2$

$$\underline{\mathbf{w}}_{n+1} = \begin{bmatrix} \phi_{1,n} - \lambda_{1,n} \frac{\Theta_{1,n}(\phi_{1,n})}{\|\Theta'_{1,n}(\phi_{1,n})\|^2} \Theta'_{1,n}(\phi_{1,n}) \\ \phi_{2,n} - \lambda_{2,n} \frac{\Theta_{2,n}(\phi_{2,n})}{\|\Theta'_{2,n}(\phi_{2,n})\|^2} \Theta'_{2,n}(\phi_{2,n}) \\ \vdots \\ \phi_{N,n} - \lambda_{N,n} \frac{\Theta_{N,n}(\phi_{N,n})}{\|\Theta'_{N,n}(\phi_{N,n})\|^2} \Theta'_{N,n}(\phi_{N,n}) \end{bmatrix} = \mathbf{P}\underline{\mathbf{w}}_{n+1} - \mathbf{F}_n, \quad (\text{B.16})$$

where

$$\mathbf{F}_n = \begin{bmatrix} \lambda_{1,n} \frac{\Theta_{1,n}(\phi_{1,n})}{\|\Theta'_{1,n}(\phi_{1,n})\|^2} \Theta'_{1,n}(\phi_{1,n}) \\ \lambda_{2,n} \frac{\Theta_{2,n}(\phi_{2,n})}{\|\Theta'_{2,n}(\phi_{2,n})\|^2} \Theta'_{2,n}(\phi_{2,n}) \\ \vdots \\ \lambda_{N,n} \frac{\Theta_{N,n}(\phi_{N,n})}{\|\Theta'_{N,n}(\phi_{N,n})\|^2} \Theta'_{N,n}(\phi_{N,n}) \end{bmatrix}.$$

## Proof of Theorem 1

---

Taking into consideration (B.10) we have that

$$\lim_{n \rightarrow \infty} \|\mathbf{F}_n\|^2 = \lim_{n \rightarrow \infty} \sum_{k=1}^N \left\| \lambda_{k,n} \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|^2} \Theta'_{k,n}(\phi_{k,n}) \right\|^2 = 0. \quad (\text{B.17})$$

Now, going back and iterating equation (B.16),  $\forall n \geq n_2$  we have:

$$\begin{aligned} \underline{\mathbf{w}}_{n+1} &= \mathbf{P}\underline{\mathbf{w}}_n - \mathbf{F}_n = \mathbf{P}\mathbf{P}_n \mathbf{w}_{n-1} - \mathbf{P}\mathbf{F}_{n-1} - \mathbf{F}_n = \\ &\dots = \prod_{j=0}^{n-n_2} \mathbf{P}\underline{\mathbf{w}}_{n_2} - \sum_{j=n_2+1}^n \prod_{l=0}^{n-j} \mathbf{P}\mathbf{F}_j - \mathbf{F}_n. \end{aligned}$$

If we left-multiply the previous equation by  $(\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T)$  we obtain

$$\begin{aligned} (\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T)\underline{\mathbf{w}}_{n+1} &= (\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T) \prod_{j=n_2}^n \mathbf{P}\underline{\mathbf{w}}_{n_2} - \\ &(\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T) \sum_{j=n_2+1}^n \prod_{l=0}^{n-j} \mathbf{P}\mathbf{F}_j - (\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T)\mathbf{F}_n. \end{aligned}$$

If we follow similar steps as in [31, Lemma 2] it can be verified that

$$\lim_{n \rightarrow \infty} (\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T)\underline{\mathbf{w}}_{n+1} = \mathbf{0}_{Nm}, \quad (\text{B.18})$$

which completes our proof.

**Remark 19.** *Note that the result of this theorem can be readily generalized to the algorithm using the Huber loss function. The only condition needed to guarantee asymptotic consensus is  $\text{lev}_{\leq 0} \tilde{\Theta}_{k,n} \subset \text{lev}_{\leq 0} \hat{\Theta}_{k,n}$ , which by construction of the loss functions is true.*

## B.4 Proof of Theorem 1.4

Recall that the projection operator, of an arbitrary vector  $\mathbf{x} \in \mathbb{R}^{Nm}$  onto the consensus subspace equals to  $P_{\mathcal{O}}(\mathbf{x}) = \mathbf{B}\mathbf{B}^T \mathbf{x}, \forall \mathbf{x} \in \mathbb{R}^{Nm}$ . Let assumptions (a), (c), (d), (e) hold. Since assumption (e) holds, together with (E.10), from [149, Lemma 1] we have that there exists  $\hat{\mathbf{w}}_{\mathcal{O}} \in \mathcal{O}$  such that

$$\lim_{n \rightarrow \infty} P_{\mathcal{O}}(\underline{\mathbf{w}}_n) = \hat{\mathbf{w}}_{\mathcal{O}}. \quad (\text{B.19})$$

Now, exploiting the triangle inequality we have that

$$\|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_O\| \leq \|\underline{\mathbf{w}}_n - P_{\mathcal{O}}(\underline{\mathbf{w}}_n)\| + \|\underline{\mathbf{w}}_O - P_{\mathcal{O}}(\underline{\mathbf{w}}_n)\| \rightarrow 0, \quad n \rightarrow \infty, \quad (\text{B.20})$$

where this limit holds from (I.14) and (E.30). The proof is complete since (E.31) implies that  $\lim_{n \rightarrow \infty} \underline{\mathbf{w}}_n = \underline{\mathbf{w}}_O$ .



# Chapter 5

## A Sparsity–Promoting Projection–Based Algorithm for Distributed Learning

*In this chapter, an APSM–based sparsity–promoting adaptive algorithm for distributed learning in ad–hoc networks will be developed. At each time instance and at each node of the network, a hyperslab is constructed based on the received measurements; this defines the region in which the solution is searched for. Sparsity encouraging variable metric projections onto these sets have been adopted. In addition, sparsity is also imposed by employing variable metric projections onto weighted  $\ell_1$  balls. A combine adapt cooperation strategy is followed. The theoretical properties of the algorithm are studied and it is shown that under some mild assumptions, the scheme enjoys monotonicity, asymptotic optimality and strong convergence to a point that lies in the consensus subspace. Finally, numerical examples verify the enhanced performance obtained by the proposed scheme compared to other algorithms, which have been developed in the context of sparsity–aware adaptive learning.*

### 5.1 Sparsity–Aware Learning

Sparsity, *i.e.*, the presence of only a small number of non-zero coefficients in an unknown signal/parameter vector, which is to be estimated, has been recently attracting an overwhelming interest under the Compressed Sensing (CS) framework [20, 53]. The task of

estimating sparse parameter vectors is met in a wide range of both distributed and centralized applications, such as data compression, echo cancellation, spectrum cartography, medical signal processing, just to name a few, *e.g.*, [19, 21, 96].

In its centralized form, our familiar linear model is considered, *i.e.*,

$$d_n = \mathbf{u}_n^T \mathbf{w}_* + v_n, \quad \forall n \in \mathbb{Z}_{\geq 0}. \quad (5.1)$$

We assume that  $\mathbf{w}_*$  is sparse, *i.e.*,  $\|\mathbf{w}_*\|_0 \ll m$ , or, in other words, it has a small number of non-zero coefficients. Suppose that a finite number of measurements, say  $K$ , is available. In that case, (5.1) can be written as

$$\mathbf{d} = \mathbf{U}\mathbf{w}_* + \mathbf{v},$$

where  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K]^T \in \mathbb{R}^{K \times m}$ ,  $\mathbf{d} = [d_1, \dots, d_K]^T \in \mathbb{R}^K$  and  $\mathbf{v} = [v_1, \dots, v_K]^T \in \mathbb{R}^K$ . Classical techniques, such as the celebrated least-squares method, fail to obtain a good estimate of the unknown parameters, since they do not take into consideration the sparsity of  $\mathbf{w}_*$  and, consequently, there is no guarantee that the estimate will predict the support, *i.e.*, the set of non-zero components, while forcing the rest to become zero. This results to an increased misadjustment between the true and the estimated values, [6]. Nevertheless, one can resort to a sparsity promoting technique, namely Least Absolute Shrinkage and Selection Operator (Lasso), and overstep the previously mentioned problem. Analytically, the Lasso estimator promotes sparsity by solving the following optimization task

$$\hat{\mathbf{w}} = \operatorname{argmin}_{\|\mathbf{w}\|_1 \leq \varrho} \|\mathbf{d} - \mathbf{U}\mathbf{w}\|^2,$$

where the term  $\|\mathbf{d} - \mathbf{U}\mathbf{w}\|$  accounts for the error residual in the estimation process; the  $\ell_1$  norm promotes sparsity by shrinking small coefficient values towards zero, *e.g.*, [19]. Most of the emphasis in solving the Lasso problem has been given on batch techniques, see, *e.g.*, [95]. However, such techniques are inappropriate for online learning, where data arrive sequentially and/or the environment is not stationary but it undergoes changes as time evolves. In the sequel, we will give a brief description of sparsity–promoting adaptive algorithms.

### 5.1.1 Sparsity-Promoting Adaptive Algorithms

Although sparsity-promoting adaptive algorithms have drawn the attention of the signal processing community for many years, see, *e.g.*, [8, 54], it is only recently that the topic is being treated in a more theoretically sound framework. There are mainly two routes to follow when the goal is the estimation of a sparse unknown vector:

- Algorithms within the spirit of  $\ell_1$  regularization, *e.g.*, [4, 5, 35, 75, 82, 103]. The a-priori information concerning the underlying sparsity is exploited by constraining the  $\ell_1$  norm. Providing this a-priori information, the convergence rate is improved significantly, and the associated error floor in the steady state is also reduced.
- Greedy algorithms, *e.g.*, [44, 100]. In a nutshell, greedy techniques estimate the positions containing the non-zero coefficients of the unknown target vector to be estimated, and then perform the computations in this subset.

As it is often the case, most of these efforts evolve along the three main axes in adaptive filtering. One is along the gradient descend rationale, as this is represented in the adaptive learning by the LMS [35, 100]. The other direction follows Newton-type arguments, as represented by the RLS [4, 5]. The other route is more recent and builds upon the APSM algorithm, *e.g.*, [82]. The algorithm, which will be developed here and was presented in [39, 45], belongs to the latter algorithmic family.

## 5.2 Set-theoretic Estimation Approach and Variable Metric Projections

As we saw in Chapters 3,4, the philosophy behind the APSM algorithm is that instead of adopting a loss function to be optimized, one obtains an estimate that lies arbitrarily close in the intersection of an infinite number of convex sets. Each one of these (convex) property sets is constructed using information which is provided by the sensed measurements. The goal of computing such a point is accomplished by projecting, in parallel, the currently available estimate onto the  $q$  (user-defined) most recently “received” sets. It has been pointed out (see, for example, [153]), that the sparsity-related a-priori knowledge can be “embedded” in the

projection operators to the benefit of the algorithm’s performance. To this end, the notion of the *variable metric projection* was introduced. The result of a variable metric projection of a vector onto a closed convex set is determined by: a) a positive definite matrix, which defines the induced inner product, b) the specific convex set, onto which the projection takes place and c) the vector itself. As it will become clear later on, for a properly chosen matrix, which is time-dependent and it is constructed via the current estimate at each time instance, the variable metric projection pushes small coefficients to diminish faster. In other words, by employing at each time instance a different inner product in our Euclidean space, we favor sparse solution vectors.

Here, the adopted property sets take the form of hyperslabs, with definition (see also Chapters 2,3)

$$S_n := \{\mathbf{w} \in \mathbb{R}^m : |d_n - \mathbf{u}_n^T \mathbf{w}| \leq \epsilon\}. \quad (5.2)$$

In the previous chapters, we considered the case of the Euclidean projection onto these hyperslabs. Let us introduce, here, the *variable metric projection*, onto the respective hyperslabs, with respect to the matrix  $\mathbf{G}_n$ , defined as [152]:

$$\forall \mathbf{w} \in \mathbb{R}^m, \quad P_{S_n}^{(\mathbf{G}_n)}(\mathbf{w}) := \mathbf{w} + \beta_n \mathbf{G}_n^{-1} \mathbf{u}_n, \quad (5.3)$$

where

$$\beta_n = \begin{cases} \frac{d_n - \mathbf{u}_n^T \mathbf{w} + \epsilon}{\|\mathbf{u}_n\|_{\mathbf{G}_n^{-1}}^2}, & \text{if } d_n - \mathbf{u}_n^T \mathbf{w} < -\epsilon, \\ 0, & \text{if } |d_n - \mathbf{u}_n^T \mathbf{w}| \leq \epsilon, \\ \frac{d_n - \mathbf{u}_n^T \mathbf{w} - \epsilon}{\|\mathbf{u}_n\|_{\mathbf{G}_n^{-1}}^2}, & \text{if } d_n - \mathbf{u}_n^T \mathbf{w} > \epsilon, \end{cases}$$

and  $\|\mathbf{u}_n\|_{\mathbf{G}_n^{-1}}^2$  denotes the weighted norm, with definition  $\|\mathbf{u}_n\|_{\mathbf{G}_n^{-1}}^2 := \mathbf{u}_n^T \mathbf{G}_n^{-1} \mathbf{u}_n$  (see Appendix C). Note that if  $\mathbf{G}_n = \mathbf{I}_m$ , then (5.3) is the Euclidean projection onto a hyperslab. The positive definite diagonal matrix  $\mathbf{G}_n^{-1}$  is constructed following similar philosophy as in [8, 153]. The  $i$ -th coefficient of its diagonal equals to  $g_{i,n}^{-1} = \frac{1-\bar{\alpha}}{m} + \bar{\alpha} \frac{|w_i^{(n)}|}{\|\mathbf{w}_n\|_1}$ , where  $\bar{\alpha} \in [0, 1)$  is a parameter, that determines the extend to which the sparsity level of the unknown vector will be taken into consideration, and  $w_i^{(n)}$  denotes the  $i$ -th component of  $\mathbf{w}_n$ . In order to grasp the reasoning of the variable metric projections, consider the ideal

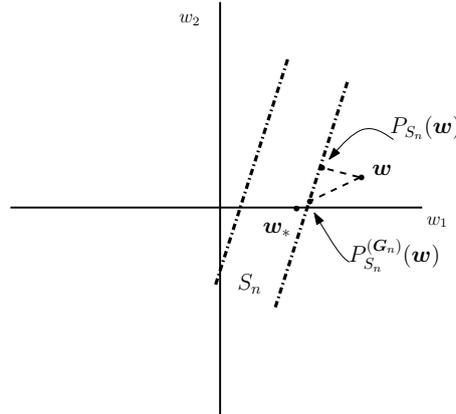


Figure 5.1: Illustration of a hyperslab, the standard metric projection of a vector  $\mathbf{w}$  onto it, denoted by  $P_{S_n}(\mathbf{w})$ , and the variable metric projection onto it.

situation, in which  $\mathbf{G}_n^{-1}$  is generated by the unknown vector  $\mathbf{w}_*$ . It is easy to verify that  $g_{i,n}^{-1} > g_{i',n}^{-1}$ , if  $i \in \text{supp}(\mathbf{w}_*)$ , and  $i' \notin \text{supp}(\mathbf{w}_*)$ , where  $\text{supp}(\cdot)$  stands for the support set of a vector, *i.e.*, the set of the non-zero coefficients. Hence, employing the variable metric projection, the amplitude of each coefficient of the vector used to construct  $\mathbf{G}_n^{-1}$  determines the weight that will be assigned to the corresponding coefficient of the second term of the right hand side in (5.3). That is, components with smaller magnitude are multiplied with small coefficients of  $\mathbf{G}_n^{-1}$ . Loosely speaking, the variable metric projections accelerate the convergence speed when tracking a sparse vector, since by assigning different weights pushes the coefficients of the estimates with small amplitude to diminish faster. The geometric implication of it is that the projection is made to “lean” towards the direction of the more significant components of the currently available estimate. Another viewpoint, as documented in [8], is the following. The coefficients of the matrix  $\mathbf{G}_n^{-1}$ , which are multiplied with the second term of the right hand side in (5.3), can be seen as  $m$  individual step-sizes, one for each coefficient. As it is by now well established in adaptive filter community (see for example [120]), the larger the step-size the faster the convergence. Hence, coefficients of large amplitude are assigned a large “step-size”, whereas the rest are multiplied by a smaller “step-size”. This results to a faster convergence speed compared to the case where the same step-size is adopted to each one of coefficients; the latter case results in  $\mathbf{G}_n = \mathbf{I}_m$ , as it has been shown in [8, 54]. Obviously, since  $\mathbf{w}_*$  is unknown, the weights rely on the available estimates, *i.e.*,  $\mathbf{w}_n$ , at each time instance. These concepts are depicted in Fig. 5.1.

**Remark 20.** *The variable metric projections rationale is in line with the so called propor-*

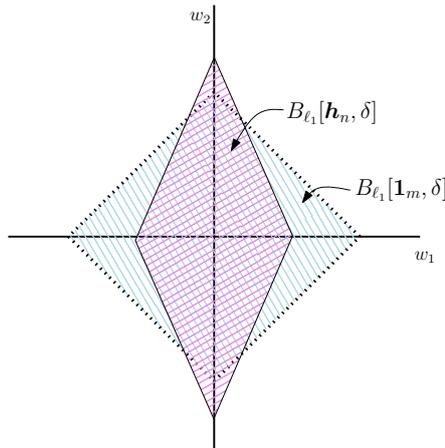


Figure 5.2: Illustration of a weighted  $\ell_1$  ball (solid line magenta) and an unweighted  $\ell_1$  ball (dashed line blue).

tionate algorithms [8, 54, 154]. At the heart of these algorithms lies the fact that at every time instance different weights are assigned to the coordinates of the vector, which leads to the next estimate.

In the algorithm which is presented here, we go one step further, as far as sparsity is concerned. In a second stage, additional sparsity-related constraints, which are built around the weighted  $\ell_1$  ball, are employed, [22]. A sparsity promoting adaptive scheme, based on set-theoretic estimation arguments, in which the constraints are weighted  $\ell_1$  balls, was presented in [82]. Given a vector of weights  $\mathbf{h}_n = [h_1^{(n)}, \dots, h_m^{(n)}]^T$ , where  $h_i^{(n)} > 0, \forall i = 1, \dots, m$ , and a positive radius,  $\delta$ , the weighted  $\ell_1$  ball is defined as:  $B_{\ell_1}[\mathbf{h}_n, \delta] := \{\mathbf{w} \in \mathbb{R}^m : \sum_{i=1}^m h_i^{(n)} |w_i| \leq \delta\}$ . Notice, that the classical  $\ell_1$  ball occurs if  $\mathbf{h}_n = \mathbf{1}_m$ . The projection onto  $B_{\ell_1}[\mathbf{h}_n, \delta]$ , is given in [82, Theorem 1], and the geometry of these sets is illustrated in Fig. 5.2.

It was shown that the estimates of the algorithm proposed in [82] converge asymptotically to a point, which lies arbitrarily close to the intersection of the hyperslabs with the weighted  $\ell_1$  balls, with the possible exception of a finite number of outliers. A generalized version of the algorithm presented in [82] will be developed in the next section.

**Remark 21.** *The weighted  $\ell_1$  ball is determined by the vector of weights and the user–defined radius  $\delta$ . Strategies of constructing the weights have been proposed in [20, 82]. For example,  $h_i^{(n)} = 1/(|w_i^{(n)}| + \tilde{\epsilon}_n)$ ,  $i = 1, \dots, m$ , where  $\tilde{\epsilon}_n$  is a sequence of positive numbers used in order to avoid divisions by zero. It has been shown, e.g., [82], that by choosing the weights according to the previously mentioned strategy and if one sets  $\delta \geq \|\mathbf{w}_*\|_0$ , then it holds that*

$\mathbf{w}_* \in B_{\ell_1}[\mathbf{h}_n, \delta]$ .

Here we should note that in [82], standard metric projections onto the hyperslabs and the weighted  $\ell_1$  balls take place. However, since we employ variable metric projections onto the hyperslabs, the induced inner product is time varying and it is determined by the matrix  $\mathbf{G}_n$ . This fact forces us to employ variable metric projections onto the respective  $\ell_1$  balls, too.

**Claim 2.** Recall the definition of the diagonal matrix  $\mathbf{G}_n$ . The variable metric projection onto  $B_{\ell_1}[\mathbf{h}_n, \delta]$  is given by  $P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathbf{G}_n)} = \mathbf{G}_n^{-\frac{1}{2}} P_{B_{\ell_1}[\mathbf{G}_n^{-\frac{1}{2}} \mathbf{h}_n, \delta]} \mathbf{G}_n^{\frac{1}{2}}$ .

*Proof.* The proof is given in Appendix D. □

## 5.3 Proposed Algorithmic Scheme

The goal is to bring together the sparsity promoting “tools”, which were discussed in Section 5.2 and to reformulate them in a fashion that is appropriate for distributed learning. We will proceed by adopting the combine–adapt diffusion strategy, which was presented in the Chapter 2. The main steps of the algorithm, for node  $k$  and at time instance  $n$ , can be summarized as follows:

### Algorithm 1:

1. The estimates from the neighborhood are received and combined with respect to the adopted combination strategy, in order to produce  $\phi_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{w}_{l,n}, \forall k \in \mathcal{N}$ , where  $a_{k,l}$  are the combination weights, which are defined in a similar way as in Chapter 4.
2. Exploiting the most recently received measurements,  $d_{k,n}, \mathbf{u}_{k,n}$ , the following hyperslab is defined:  $S_{k,n} = \{\mathbf{w} \in \mathbb{R}^m : |d_{k,n} - \mathbf{u}_{k,n}^T \mathbf{w}| \leq \epsilon_k\}$ , where the parameter  $\epsilon_k$  is allowed to vary from node to node. The aggregate  $\phi_{k,n}$  is projected, using variable metric projections, onto the  $q$  most recent hyperslabs, constructed locally in node  $k$ ; in the sequel, their convex combination is computed. Analytically, the sliding window  $\mathcal{J}_n := \overline{\{0, n - q + 1\}}, n$  is defined, and it determines the hyperslabs that will be considered at time instance  $n$ . Given the set of weights  $\forall j \in \mathcal{J}_n, \omega_{k,j}$ , where

$\sum_{j \in \mathcal{J}_n} \omega_{k,j} = 1, \forall k \in \mathcal{N}$ , the convex combination of the projections onto the hyperslabs, *i.e.*,  $\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}^{(\mathbf{G}_n)}(\boldsymbol{\phi}_{k,n})$  is computed. The effect of projecting onto a  $q > 1$  number of hyperslabs is to speed up convergence [82].

3. The result of the previous step is projected onto the sparsity constraint set, *i.e.*, the weighted  $\ell_1$  ball.

The previous steps can be encoded in the following mathematical formula:

$$\mathbf{w}_{k,n+1} = P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathbf{G}_n)} \left( \boldsymbol{\phi}_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}^{(\mathbf{G}_n)}(\boldsymbol{\phi}_{k,n}) - \boldsymbol{\phi}_{k,n} \right) \right), \quad (5.4)$$

where  $\mu_{k,n} \in (0, 2\mathcal{M}_{k,n})$ , and

$$\mathcal{M}_{k,n} := \begin{cases} \frac{\sum_{j \in \mathcal{J}_n} \omega_{k,j} \|P_{S_{k,j}}^{(\mathbf{G}_n)}(\boldsymbol{\phi}_{k,n}) - \boldsymbol{\phi}_{k,n}\|^2}{\left\| \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}^{(\mathbf{G}_n)}(\boldsymbol{\phi}_{k,n}) - \boldsymbol{\phi}_{k,n} \right\|^2}, & \text{if } \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}^{(\mathbf{G}_n)}(\boldsymbol{\phi}_{k,n}) \neq \boldsymbol{\phi}_{k,n} \\ 1, & \text{otherwise.} \end{cases} \quad (5.5)$$

The algorithm has an elegant geometrical interpretation, which can be seen in Fig. 5.3.

From the theoretical analysis concerning convergence, it turns out that the weighted  $\ell_1$  ball, as well as  $\mathbf{G}_n$ , have to be the same for every node of the network, which implies that this information cannot be constructed locally. This fact, as it will be established in Appendix E, is essential in order to guarantee consensus. Hence, a reasonable strategy is to construct  $\mathbf{h}_n$  and  $\mathbf{G}_n$ , using the methodology described in Section 5.2, via  $\mathbf{w}_{k_{opt},n}$ , where  $k_{opt}$  is the node with the smallest noise variance. It is obvious that this requires knowledge, in every node, of  $\mathbf{w}_{k_{opt},n}$  something that is in general infeasible. However, it is not essential to update the parameters at every time instance. Instead,  $\mathbf{h}_n$  and  $\mathbf{G}_n$  can be updated at every,  $n' \geq 1$ , time instances, where  $n'$  are the time steps required for  $\mathbf{w}_{k_{opt},n}$  to be distributed over the network. Experiments regarding the robustness of the proposed algorithm with respect to  $n'$  are given in the Numerical Examples section. Moreover, as it will become clear in the Numerical Examples section, it turns out that the algorithm is robust in cases where the knowledge of the less noisy node is not available, and/or in cases where the assumption that these quantities must be common to all nodes is violated and each node uses the locally

available values. It should be pointed out that in the adaptive filtering such deviations between theory and practice are not uncommon. The most celebrated example is the so called *independence assumption* adopted in the LMS, which is commonly employed to prove convergence, although in practice it does not hold, *e.g.*, [120].

Regarding the complexity of the algorithm, it has been shown in [82], that if standard metric projections take place, then the complexity of the respective algorithm is  $O(qm)$ , coming from the projection operators and  $O(m\log_2 m)$  occurring from the projection onto the weighted  $\ell_1$  ball. If we employ the variable metric projections, at each node, it is obvious that the term  $\mathbf{G}_n^{-1}\mathbf{u}_{k,j}$ ,  $j \in \mathcal{J}_n$  has to be computed, and this adds  $qm$  multiplication operations.

**Remark 22.** *The algorithm presented in [82] is a special case of the scheme in (5.4), if  $N = 1$  and  $\mathbf{G}_n = \mathbf{I}_m$ . The same also holds for the IPNLMS [8] if we let  $N = 1$ ,  $q = 1$ ,  $\epsilon_k = 0$  and  $P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathbf{G})} = I$ , where  $I$  stands for the identity operator.*

As it will be verified in Appendix E, the Algorithm 1 in (5.4) enjoys monotonicity, asymptotic optimality and strong convergence to a point that lies in the consensus subspace. The assumptions under which the previous hold are the following.

**Assumptions.**

- (a) Define  $\forall n \in \mathbb{Z}_{\geq 0}$ ,  $\Omega_n = B_{\ell_1}[\mathbf{h}_n, \delta] \cap \left( \bigcap_{j \in \mathcal{J}_n} \bigcap_{k \in \mathcal{N}} S_{k,j} \right)$ . Assume that there exists  $n_0 \in \mathbb{Z}_{\geq 0}$ , such that  $\Omega := \bigcap_{n \geq n_0} \Omega_n \neq \emptyset$ .
- (b) There exists  $n_1 \in \mathbb{Z}_{\geq 0}$ , such that  $\mathbf{G}_n = \mathbf{G}_{n_1} =: \mathbf{G}, \forall n \geq n_1$ . In other words, the update of the matrix  $\mathbf{G}_n$  pauses after a finite number of iterations<sup>1</sup>.
- (c) Assume a sufficiently small  $\epsilon_1$ , such that  $\forall k \in \mathcal{N}$ ,  $\frac{\mu_{k,n}}{\mathcal{M}_{k,n}} \in [\epsilon_1, 2 - \epsilon_1]$ .
- (d) Assume  $\forall k \in \mathcal{N}$   $\tilde{\omega}_k := \inf\{\omega_{k,j} : j \in \mathcal{J}_n, n \in \mathbb{Z}_{\geq 0}\} > 0$ .
- (e) Define  $\mathfrak{C} := \Omega \cap \mathcal{O}$ , where the cartesian product space  $\Omega := \underbrace{\Omega \times \dots \times \Omega}_N$ , where  $\mathcal{O}$  is the consensus subspace (see Chapter 4). We assume that  $\text{ri}_{\mathcal{O}}\Omega \neq \emptyset$ , where this term stands for the relative interior of  $\Omega$  with respect to  $\mathcal{O}$ .

<sup>1</sup>Notice that the matrix  $\mathbf{G}_n$  is constructed via  $\mathbf{w}_{k_{opt},n}$ , hence  $\forall n \geq n_1$ , the variable metric projections is determined by  $\mathbf{w}_{k_{opt},n_1}$ . In practice, for sufficiently large  $n_1$ , the algorithm has converged and the fact that  $\mathbf{G}_n$  is not updated does not affect the performance of the algorithm.

**Theorem 7.** *Under the previous assumptions, the following hold:*

- (1) **Monotonicity.** *Under assumptions (a), (b), (c), it holds that  $\forall n \geq z_0, \forall \hat{\mathbf{w}} \in \mathcal{C}, \|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}} \leq \|\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}$ , where  $z_0 := \max\{n_0, n_1\}$ ,  $\underline{\mathbf{G}}$  is the  $Nm \times Nm$  block-diagonal matrix, with definition  $\underline{\mathbf{G}} := \text{diag}\{\underbrace{\mathbf{G}, \dots, \mathbf{G}}_N\}$ , and  $\underline{\mathbf{w}}_n = [\mathbf{w}_{1,n}^T, \dots, \mathbf{w}_{N,n}^T]^T \in \mathbb{R}^{Nm}, \forall n \in \mathbb{Z}_{\geq 0}$ .*
- (2) **Asymptotic Optimality.** *If assumptions (a), (b), (c), (d) hold true then  $\lim_{n \rightarrow \infty} \max\{d(\mathbf{w}_{k,n+1}, S_{k,j}) \mid j \in \mathcal{J}_n\} = 0, \forall k \in \mathcal{N}$ , where  $d(\cdot, S_{k,j})$  denotes the distance of  $\mathbf{w}_{k,n+1}$  from  $S_{k,j}$ . The previous implies that the distance of the estimates from the respective hyperslabs will tend asymptotically to zero.*
- (3) **Asymptotic Consensus.** *Consider that assumptions (a), (b), (c), (d) hold. Then  $\lim_{n \rightarrow \infty} \|\mathbf{w}_{k,n} - \mathbf{w}_{l,n}\| = 0, \forall k, l \in \mathcal{N}$ .*
- (4) **Strong Convergence.** *Under assumptions (a), (b), (c), (d), (e), it holds that  $\lim_{n \rightarrow \infty} \underline{\mathbf{w}}_n = \hat{\mathbf{w}}_{\mathcal{O}}, \hat{\mathbf{w}}_{\mathcal{O}} \in \mathcal{O}$ . So, the estimates for the whole network, converge to a point that lies in the consensus subspace.*

*Proof.* The proof is given in Appendix E. □

## 5.4 Numerical Examples

In this section, the performance of the proposed algorithm is validated within the system identification framework. In the first experiment, we compare the proposed algorithm against others in the context of a non-distributed system identification task. This essentially allow us to evaluate the use of the variable metric projections scheme, together with the weighted  $\ell_1$  ball. More specifically, we compare the proposed algorithm with the Adaptive Projection based algorithm using Weighted  $\ell_1$  Balls (APWL1) [82], with the Online Cyclic Coordinate Descent Time Weighted Lasso (OCCD-TWL), the Online Cyclic Coordinate Descent Time and Norm Weighted LASSO (OCCD-TNWL), both proposed in [4], and with the LMS-based, Sparse Adaptive Orthogonal Matching Pursuit (Spadomp) [100]. The unknown vector is of dimension  $m = 512$  and the number of non-zero coefficients, equals to 20. Moreover, the

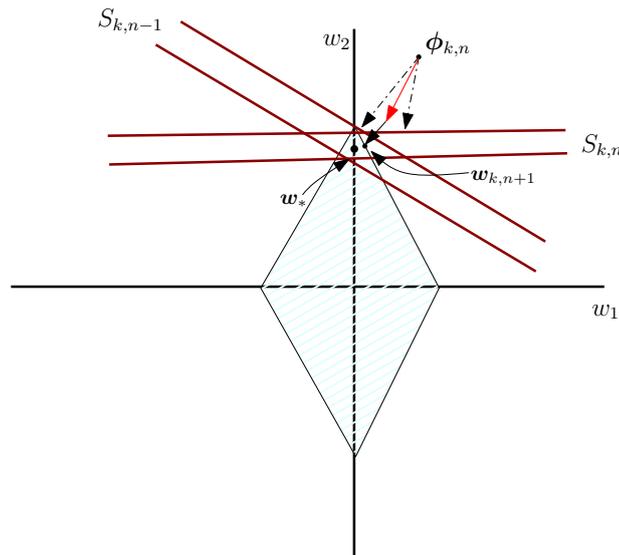


Figure 5.3: Geometrical interpretation of the algorithm. The number of hyperslabs onto which  $\phi_{k,n}$  is projected, using variable metric projections, is  $q = 2$ . The result of these two projections, which are illustrated by the dash dotted black line, is combined (red line) and the result is projected (solid black line) onto the sparsity promoting weighted  $\ell_1$  ball, in order to produce the next estimate.

input samples  $\mathbf{u}_n = [u_n, \dots, u_{n-m+1}]^T$  are drawn from a Gaussian distribution, with zero mean and standard deviation equal to 1. The noise process is Gaussian with variance equal to  $\sigma^2 = 0.01$ . Finally, the adopted performance metric, which will be used, is the average Mean Square Deviation (MSD), given by  $\text{MSD}(n) = 1/N \sum_{k=1}^N \|\mathbf{w}_{k,n} - \mathbf{w}_*\|^2$ , and the curves occur from an averaging of 100 realizations for smoothing purposes.

In the projection-based algorithms, *i.e.*, the proposed and the APWL1, the number of hyperslabs used per time update equals to  $q = 55$ , the width of the hyperslabs equals to  $\epsilon = 1.3 \times \sigma$ , and the step-size equals to  $\mu_n = 0.2 \times \mathcal{M}_n$ , where  $\mathcal{M}_n$  is given in (5.5) and the node subscript is omitted. It should be pointed out that the performance of the algorithm turns out to be relatively insensitive to different choices of the parameter  $\epsilon$ . A detailed experimental analysis on how different choices of  $\epsilon$  affect the projection-based algorithms, has taken place in [82]. Moreover, for the weights we choose  $\omega_n = 1/q$ . These choices are not necessarily optimal, albeit they lead to a good trade-off between the convergence speed and the steady state error floor. Regarding the choice of  $q$ , the larger the  $q$  the faster the convergence. This behaviour is also met in the Affine Projection Algorithm (APA), where the larger the number of affine sets, employed at each time instance, the faster the convergence. The parameter  $q$  is not a critical parameter, and one can choose it depending

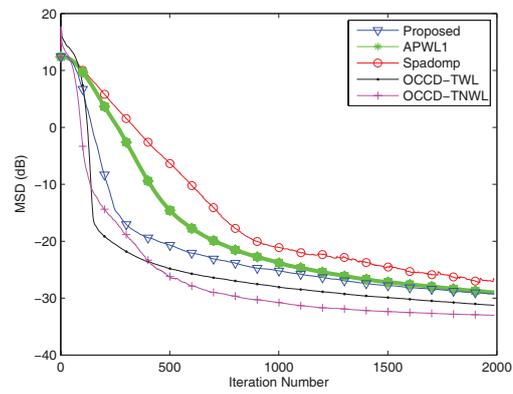


Figure 5.4: MSD for the experiment 1.

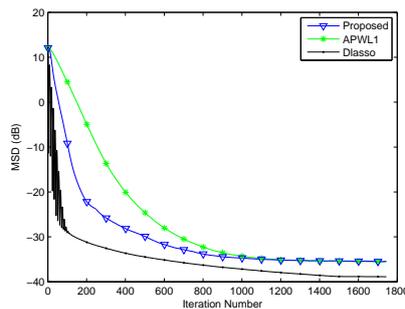


Figure 5.5: MSD for the experiment 2.

on the complexity load that can be afforded by the algorithm in real time operations. The radius of the weighted  $\ell_1$  ball equals to  $\delta = \|\mathbf{w}_*\|_0$  and the weights are constructed according to the discussion in Section 5.2. It should be stressed out that we experimentally observed that the proposed algorithm is rather insensitive to overestimated values of the sparsity level, which implies that even if we do not know the exact value of  $\|\mathbf{w}_*\|_0$ , if we set  $\delta \geq \|\mathbf{w}_*\|_0$  the proposed algorithm exhibits a good performance; this behavior was also observed in [82]. Furthermore, we set  $\tilde{\epsilon}_n = 10^{-2}$ . The weighting matrix  $\mathbf{G}_n$  is defined according to the strategy presented in Section 5.2. Regarding the parameter  $\bar{\alpha}$ , we observed that a value close to 1 leads to a fast convergence speed but it increases the steady state error floor, and vice versa. So, at the beginning of the adaptation, we choose  $\bar{\alpha} = 0.99$  and at every 250 time instances, we set  $\bar{\alpha} = \bar{\alpha}/2$ . Finally,  $\mathbf{h}_n$  and  $\mathbf{G}_n$  are updated at every time instance, *i.e.*,  $n' = 1$ . In the OCCD-TWL and the OCCD-TNWL, the regularization parameter is chosen to be  $\lambda_{\text{TWL}} = \sqrt{2\sigma^2 n \log m}$ ,  $\lambda_{\text{TNWL}} = \sqrt{2\sigma^2 n^{4/3} \log m}$ , respectively, as advised in [4]. The step size, adopted in the Spadomp, equals to 0.2, due to the fact that this choice gives similar steady state error floor with the projection-based algorithms<sup>2</sup>. The forgetting factor of OCCD-TWN, OCCD-TNWL and Spadomp equals to 1 since, in the specific example, the system under consideration does not change with time. From Fig. 5.4, it can be seen that the proposed algorithm exhibits faster convergence speed compared to the APWL1 to a common error floor. Moreover, the proposed algorithm outperforms the Spadomp, since it converges faster and the steady state error floor is slightly better. We should point out, that the complexity of the Spadomp is  $O(m)$ , which implies that for the previously mentioned choice of  $q$ , the proposed algorithm is of larger complexity, yet still of linear dependence

<sup>2</sup>Extensive experiments have shown that a choice of a smaller step-size, results in a slower convergence speed, without significant improvement in the steady state error floor.

on the number of unknown parameters. Regarding the OCCD-TWL, we observe that its performance is slightly better, compared to the proposed one, albeit the complexity of the algorithm is  $O(m^2)$ . Finally, the OCCD-TNWL outperforms the rest of the algorithms, at the expense of even higher complexity, which is approximately twice that of OCCD-TWL.

In the second experiment, we consider a network consisted of  $N = 10$  nodes, in which the nodes are tasked to estimate an unknown parameter  $\mathbf{w}_*$  of dimension  $m = 256$ . The number of non-zero coefficients, of the unknown parameter equals to 20 and each node has access to the measurements  $(d_{k,n}, \mathbf{u}_{k,n})$ , where the regressors are defined as in the previous experiment. The variance of the noise at each node is  $\sigma_k^2 = 0.01\zeta_k$ , where  $\zeta_k \in [0.5, 1]$ , following the uniform distribution. We compare the proposed algorithm with the distributed APWL1, *i.e.*, if we let  $\mathbf{G}_n = \mathbf{I}_m$ , and the distributed Lasso (Dlasso) [96]. The Dlasso is a batch algorithm, which implies that the data have to be available prior to start the processing. So, here we assume that at every time instance, in which a new pair of data samples becomes available, the algorithm is re-initialized so as to solve a new optimization problem. For the projection-based algorithms,  $q = 20$  and the rest of the parameters are chosen as in the previous experiment. Moreover, the combiners  $a_{k,l}$  are chosen with respect to the Metropolis rule (see Chapter 2). Finally, the regularization parameter in the Dlasso is set via the distributed cross-validation procedure, which is proposed in [96]. From Fig. 5.5, we observe that the Dlasso outperforms the projection-based algorithms and that the proposed algorithm converges faster than APWL1. However, the complexity of the proposed algorithm is significantly lower than that of the Dlasso. Dlasso, at every time instance, requires the inversion of a  $m \times m$  matrix.

In the third experiment, we study the sensitivity of the proposed algorithm to the choice of the parameter  $n'$ , *i.e.*, the frequency at which  $\mathbf{h}_n$  and  $\mathbf{G}_n$  are updated. To this end, the parameters are the same as in the previous experiment, but we set different values to  $n'$ . Fig. 5.6 illustrates that the algorithm is relatively insensitive to the frequency of the updates, as even in the case where  $n' = 20$  the algorithm exhibits fast convergence speed. This is important, since the robustness of the proposed scheme to choice of the parameter  $n'$  makes it suitable to be adopted in distributed learning.

In the fourth experiment, the performance of the algorithm in a non-stationary environment is validated. It is by now well established that a fast convergence speed does not

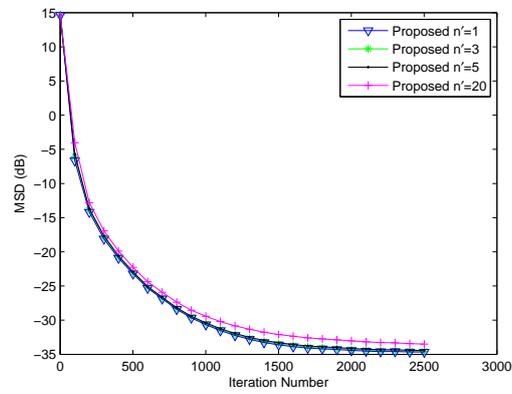


Figure 5.6: MSD for the experiment 3.

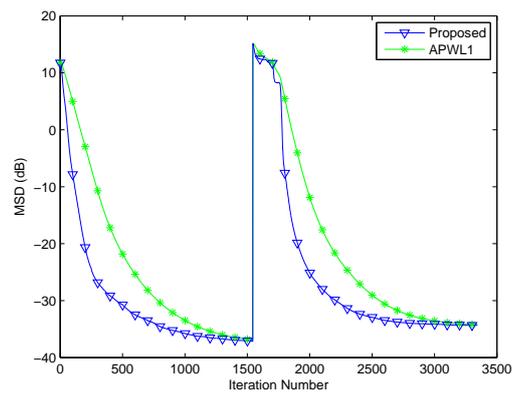


Figure 5.7: MSD for the experiment 4.

necessarily imply a good tracking ability [64]. To this end, we consider that a sudden change in the unknown parameter takes place. So, until  $\mathbf{w}_*$  changes, the parameters remain the same as in the second experiment and after the sudden change, we have that  $\|\mathbf{w}_*\|_0 = 15$ . The radius of the weighted  $\ell_1$  ball is set equal to 23, since we have observed that the performance is relative insensitive to choices of  $\delta$ , as long as it remains larger than  $\|\mathbf{w}_*\|_0$ . Furthermore, we assume the algorithm is able to monitor sudden changes of the orbit  $(\mathbf{w}_{k,n})_{n \in \mathbb{Z}_{\geq 0}}$ , in order to reset the value of  $\bar{\alpha}$  when the channel changes. To be more specific, we reset the value of  $\bar{\alpha}$ , if the ratio  $\|\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}\| / \|\mathbf{w}_{k,n} - \mathbf{w}_{k,n-1}\|$ ,  $\forall k \in \mathcal{N}$ , is greater than a threshold, which is chosen, here, to be equal to 10. This strategy is adopted since we observed that if the algorithm has converged, the previously mentioned ratio takes values close to 1, whereas if an abrupt change takes place in the unknown parameter, then the value of the ratio increases significantly. From Fig. 5.7, it can be observed that both the projection-based algorithms enjoy good tracking ability, when a sudden change occurs. Moreover, as in the previous experiments, the proposed algorithm converges faster than the APWL1 to a similar error floor.

Finally, in the fifth experiment, we study the robustness of the proposed scheme, with respect to adopting different strategies in order to construct  $\mathbf{h}_n$  and  $\mathbf{G}_n$ . To this end, we consider the following strategies: a) the previously mentioned quantities are constructed using the node with the smallest noise variance (Proposed a), b)  $\mathbf{h}_n$  and  $\mathbf{G}_n$  are generated via the node with the largest variance (Proposed b) and c)  $\mathbf{h}_n$  and  $\mathbf{G}_n$  are constructed locally at every node (Proposed c). Obviously, the latter one violates the theoretical assumption of having common weights to all nodes. In order to verify whether the nodes reach consensus, we plot the squared distance of  $\underline{\mathbf{w}}_n$  from the consensus subspace, i.e.,  $\|\underline{\mathbf{w}}_n - P_{\mathcal{O}}(\underline{\mathbf{w}}_n)\|^2$ . As in the previous experiments, the curves occur from an averaging of 100 independent experiments. From Fig. 5.8, it can be readily seen that the distance of  $\underline{\mathbf{w}}_n$  from the consensus subspace, is decreasing as time steps increase. It is interesting that even in the Proposed c, where the assumption for achieving asymptotic consensus is violated, the estimates for the whole network tend asymptotically to the consensus subspace. Loosely speaking, even if there cannot be theoretical guarantees that the nodes will achieve asymptotic consensus in the case where each node constructs  $\mathbf{h}_n$  and  $\mathbf{G}_n$  using local information, the fact that the estimates received from the neighbourhood are combined at each step, leads the nodes to

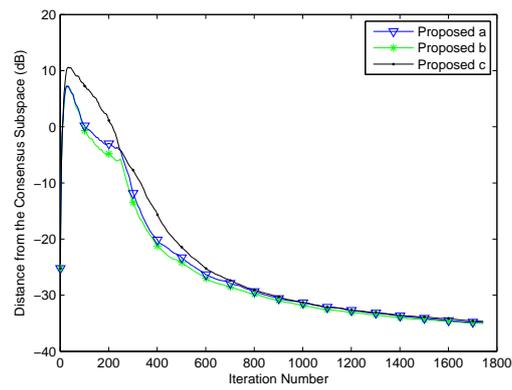


Figure 5.8: Squared distance from the consensus subspace, for experiment 5.

asymptotic consensus.

# Appendix C

## Basic Concepts Of Convex Analysis Employing Weighted Inner Products

The stage of discussion will be  $\mathbb{R}^m$ , the induced inner product, given a positive definite  $m \times m$  matrix  $\mathbf{V}$ , is  $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle_{\mathbf{V}} = \mathbf{w}_1^T \mathbf{V} \mathbf{w}_2$  and the weighted norm is  $\|\mathbf{w}\|_{\mathbf{V}}^2 = \mathbf{w}^T \mathbf{V} \mathbf{w}$ . Given a convex function,  $\Theta$ , the subdifferential of  $\Theta$ , with respect to  $\mathbf{V}$ , at an arbitrary point,  $\mathbf{w}_1$ , is defined as the set of all subgradients of  $\Theta$  at  $\mathbf{w}_1$ , *i.e.*,

$$\partial_{(\mathbf{V})}\Theta(\mathbf{w}_1) := \{\mathbf{s} \in \mathbb{R}^m : \Theta(\mathbf{w}_1) + \langle \mathbf{w} - \mathbf{w}_1, \mathbf{s} \rangle_{\mathbf{V}} \leq \Theta(\mathbf{w}), \forall \mathbf{w} \in \mathbb{R}^m\}.$$

The distance of an arbitrary point  $\mathbf{w}$  from a closed non-empty convex set  $\mathcal{C}$ , with respect to  $\mathbf{V}$ , is given by the distance function

$$\begin{aligned} d_{(\mathbf{V})}(\cdot, \mathcal{C}) : \mathbb{R}^m &\rightarrow [0, +\infty) \\ &: \mathbf{w} \mapsto \inf \{\|\mathbf{w} - \mathbf{x}\|_{\mathbf{V}} : \mathbf{x} \in \mathcal{C}\}, \end{aligned}$$

and if we let  $\mathbf{V}$  be the identity matrix, the Euclidean distance is given. Finally, the projection mapping,  $P_{\mathcal{C}}^{(\mathbf{V})}$  onto  $\mathcal{C}$ , is defined as  $P_{\mathcal{C}}^{(\mathbf{V})}(\mathbf{w}) := \operatorname{argmin}_{\mathbf{x} \in \mathcal{C}} \|\mathbf{w} - \mathbf{x}\|_{\mathbf{V}}$ , and as in the distance function, if  $\mathbf{V} = \mathbf{I}_m$  the standard metric projection is obtained.

# Appendix D

## Variable Metric Projection onto the Weighted $\ell_1$ Ball

The variable metric projection of  $\mathbf{w}$ , onto  $B_{\ell_1}[\mathbf{h}_n, \delta]$ , is given by

$$\begin{aligned} & \min_{\mathbf{x} \in B_{\ell_1}[\mathbf{h}_n, \delta]} \|\mathbf{w} - \mathbf{x}\|_{\mathbf{G}_n}^2 \\ & \text{s.t. } \sum_{i=1}^m h_i^{(n)} |x_i| \leq \delta, \end{aligned}$$

where  $\mathbf{x} := [x_1, \dots, x_m]^T$ . However,  $\|\mathbf{w} - \mathbf{x}\|_{\mathbf{G}_n}^2 = \|\mathbf{G}_n^{\frac{1}{2}}(\mathbf{w} - \mathbf{x})\|^2 = \|\mathbf{G}_n^{\frac{1}{2}}\mathbf{w} - \boldsymbol{\xi}\|^2$ , where  $\boldsymbol{\xi} := \mathbf{G}_n^{\frac{1}{2}}\mathbf{x}$ . Moreover,  $\mathbf{x} = \mathbf{G}_n^{-\frac{1}{2}}\boldsymbol{\xi} \Leftrightarrow x_i = \sqrt{g_{i,n}^{-1}}\xi_i, i = 1, \dots, m$ , where  $\xi_i$  are the coefficients of  $\boldsymbol{\xi}$ . From the previous, it holds that  $\sum_{i=1}^m h_i^{(n)} |x_i| = \sum_{i=1}^m \sqrt{g_{i,n}^{-1}} h_i^{(n)} |\xi_i|$ . Hence the initial optimization problem, is equivalent to

$$\begin{aligned} & \min_{\boldsymbol{\xi}} \|\mathbf{G}_n^{\frac{1}{2}}\mathbf{w} - \boldsymbol{\xi}\|^2 \\ & \text{s.t. } \sum_{i=1}^m \sqrt{g_{i,n}^{-1}} h_i^{(n)} |\xi_i| \leq \delta. \end{aligned}$$

The solution of the previous optimization, is the standard metric projection of  $\mathbf{G}_n^{\frac{1}{2}}\mathbf{w}$  onto  $B_{\ell_1}[\mathbf{G}_n^{-\frac{1}{2}}\mathbf{h}_n, \delta]$  and it can be found in [82]. So, from the previous  $\boldsymbol{\xi}_{\text{opt}} = P_{B_{\ell_1}[\mathbf{G}_n^{-\frac{1}{2}}\mathbf{h}_n, \delta]}(\mathbf{G}_n^{\frac{1}{2}}\mathbf{w}) \Leftrightarrow P_{B_{\ell_1}[\mathbf{G}_n^{-\frac{1}{2}}\mathbf{h}_n, \delta]}^{(\mathbf{G}_n)}(\mathbf{w}) = \mathbf{G}_n^{-\frac{1}{2}} P_{B_{\ell_1}[\mathbf{G}_n^{-\frac{1}{2}}\mathbf{h}_n, \delta]}(\mathbf{G}_n^{\frac{1}{2}}\mathbf{w})$ .

# Appendix E

## Proof of Theorem 1

### E.1 Monotonicity

**Lemma 3.** Define the following non-negative loss functions,  $\forall k \in \mathcal{N}$ :

$$\forall n \in \mathbb{Z}_{\geq 0}, \forall \mathbf{w} \in \mathbb{R}^m, \quad \Theta_{k,n}(\mathbf{w}) := \begin{cases} \sum_{j \in \mathcal{I}_{k,n}} \frac{\omega_{k,j} d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j})}{L_{k,n}} d_{(\mathbf{G})}(\mathbf{w}, S_{k,j}), & \text{if } \mathcal{I}_{k,n} \neq \emptyset, \\ 0, & \text{if } \mathcal{I}_{k,n} = \emptyset, \end{cases} \quad (\text{E.1})$$

where  $\mathcal{I}_{k,n} := \{j \in \mathcal{J}_n : \phi_{k,n} \notin S_{k,j}\}$  and  $L_{k,n} := \sum_{j \in \mathcal{J}_n} \omega_{k,j} d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j})$ . Then (5.4) is equivalent to<sup>1</sup>

$$\forall n \in \mathbb{Z}_{\geq 0}, \forall k \in \mathcal{N}, \quad \mathbf{w}_{k,n+1} = \begin{cases} P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathbf{G})} \left( \phi_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\mathbf{G}}^2} \Theta'_{k,n}(\phi_{k,n}) \right), & \text{if } \mathcal{I}_{k,n} \neq \emptyset, \\ P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathbf{G})}(\phi_{k,n}), & \text{if } \mathcal{I}_{k,n} = \emptyset, \end{cases} \quad (\text{E.2})$$

where  $\Theta'_{k,n}(\phi_{k,n})$  is the subgradient of the function and  $\lambda_{k,n} \in (0, 2)$ .

*Proof.* First of all, notice that if  $\mathcal{I}_{k,n} \neq \emptyset$ , then there exists  $j_0 \in \mathcal{J}_n$  such that  $\phi_{k,n} \notin S_{k,j_0} \Leftrightarrow d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j_0}) > 0$ . Hence,  $L_{k,n} \geq \omega_{k,j_0} d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j_0}) > 0$ , which implies that the denominator in (E.2) is positive and the cost function is well defined. Now, a subgradient of the distance function, *i.e.*,  $d_{(\mathbf{G})}(\cdot, S_{k,j})$ , is the following [151]:

$$d'_{(\mathbf{G})}(\mathbf{w}, S_{k,j}) = \begin{cases} \frac{\mathbf{w} - P_{S_{k,j}}^{(\mathbf{G})}(\mathbf{w})}{d_{(\mathbf{G})}(\mathbf{w}, S_{k,j})}, & \text{if } \mathbf{w} \notin S_{k,j} \\ \mathbf{0}, & \text{otherwise.} \end{cases} \quad (\text{E.3})$$

<sup>1</sup>The time dependence on  $\mathbf{G}_n$  is omitted for simplicity in notation.

Recalling basic properties of the subdifferential (see for example [65]), we have that

$$\partial\Theta_{k,n}(\mathbf{w}) = \begin{cases} \sum_{j \in \mathcal{I}_{k,n}} \frac{\omega_{k,j} d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j})}{L_{k,n}} \partial d_{(\mathbf{G})}(\mathbf{w}, S_{k,j}), & \text{if } \mathcal{I}_{k,n} \neq \emptyset, \\ \{\mathbf{0}\}, & \text{if } \mathcal{I}_{k,n} = \emptyset. \end{cases} \quad (\text{E.4})$$

So, combining (E.3), (E.4) and if  $\mathcal{I}_{k,n} \neq \emptyset$  we have

$$\begin{aligned} \Theta'_{k,n}(\phi_{k,n}) &= \sum_{j \in \mathcal{I}_{k,n}} \frac{\omega_{k,j} d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j})}{L_{k,n}} \frac{\phi_{k,n} - P_{S_{k,j}}^{(\mathbf{G})}(\phi_{k,n})}{d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j})} \\ &= \frac{1}{L_{k,n}} \sum_{j \in \mathcal{I}_{k,n}} \omega_{k,j} \left( \phi_{k,n} - P_{S_{k,j}}^{(\mathbf{G})}(\phi_{k,n}) \right) \\ &= \frac{1}{L_{k,n}} \sum_{j \in \mathcal{J}_n} \omega_{k,j} \left( \phi_{k,n} - P_{S_{k,j}}^{(\mathbf{G})}(\phi_{k,n}) \right). \end{aligned} \quad (\text{E.5})$$

Nevertheless, since  $\mathcal{I}_{k,n} \neq \emptyset$ , then there exists  $j_0 \in \mathcal{J}_n$  such that  $\phi_{k,n} \notin S_{k,j_0} \Leftrightarrow P_{S_{k,j_0}}^{(\mathbf{G})}(\phi_{k,n}) \neq \phi_{k,n}$ . So, if  $\mathcal{I}_{k,n} \neq \emptyset$  then  $\Theta'_{k,n}(\phi_{k,n}) \neq \mathbf{0}_m$ . Following similar steps as in [82], it can be proved that  $\forall n \geq z_0, \forall j \in \mathcal{J}_n, \forall k \in \mathcal{N}, \Theta'_{k,n}(\phi_{k,n}) = \mathbf{0}_m \Leftrightarrow \phi_{k,n} = \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j}}^{(\mathbf{G})}(\phi_{k,n})$ . From this fact, if we define  $\mu_{k,n} := \mathcal{M}_{k,n} \lambda_{k,n}$ , and if we substitute (E.5) in (E.2) the lemma is proved.  $\square$

**Claim 3.** *It holds that  $\|\mathbf{P}\underline{\mathbf{w}} - \check{\underline{\mathbf{w}}}\|_{\underline{\mathbf{G}}} \leq \|\underline{\mathbf{w}} - \check{\underline{\mathbf{w}}}\|_{\underline{\mathbf{G}}}, \forall \check{\underline{\mathbf{w}}} \in \mathcal{O}, \forall \underline{\mathbf{w}} \in \mathbb{R}^{Nm}$ , where  $\mathbf{P}$  is a  $Nm \times Nm$  consensus matrix with  $\|\mathbf{P}\| = 1$ .*

*Proof.* From the definition of  $\|\cdot\|_{\underline{\mathbf{G}}}$ , it can be readily seen that  $\|\mathbf{P}\underline{\mathbf{w}} - \check{\underline{\mathbf{w}}}\|_{\underline{\mathbf{G}}} = \|\underline{\mathbf{G}}^{\frac{1}{2}}(\mathbf{P}\underline{\mathbf{w}} - \check{\underline{\mathbf{w}}})\| = \|\underline{\mathbf{G}}^{\frac{1}{2}}\mathbf{P}(\underline{\mathbf{w}} - \check{\underline{\mathbf{w}}})\|$ , where this holds since  $\check{\underline{\mathbf{w}}} \in \mathcal{O}$ . Moreover,  $\underline{\mathbf{w}} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_N \end{bmatrix}, \mathbf{w}_k \in \mathbb{R}^m, k \in \mathcal{N}$

and  $\check{\underline{\mathbf{w}}} \in \mathcal{O} \Leftrightarrow \check{\underline{\mathbf{w}}} = \begin{bmatrix} \check{\underline{\mathbf{w}}} \\ \vdots \\ \check{\underline{\mathbf{w}}} \end{bmatrix}, \check{\underline{\mathbf{w}}} \in \mathbb{R}^m$ . Recalling the definition of the consensus matrix, with

coefficients  $a_{k,l}, k, l \in \mathcal{N}$ , we have the following

$$\begin{aligned}
 \|\underline{\mathbf{G}}^{\frac{1}{2}} \mathbf{P}(\underline{\mathbf{w}} - \underline{\check{\mathbf{w}}})\| &= \left\| \begin{bmatrix} \mathbf{G}^{\frac{1}{2}} \sum_{l \in \mathcal{N}_1} a_{1,l} (\mathbf{w}_l - \check{\mathbf{w}}) \\ \vdots \\ \mathbf{G}^{\frac{1}{2}} \sum_{l \in \mathcal{N}_N} a_{N,l} (\mathbf{w}_l - \check{\mathbf{w}}) \end{bmatrix} \right\| \\
 &= \left\| \begin{bmatrix} \sum_{l \in \mathcal{N}_1} a_{1,l} \mathbf{G}^{\frac{1}{2}} (\mathbf{w}_l - \check{\mathbf{w}}) \\ \vdots \\ \sum_{l \in \mathcal{N}_N} a_{N,l} \mathbf{G}^{\frac{1}{2}} (\mathbf{w}_l - \check{\mathbf{w}}) \end{bmatrix} \right\| \\
 &= \left\| \mathbf{P} \begin{bmatrix} \mathbf{G}^{\frac{1}{2}} (\mathbf{w}_1 - \check{\mathbf{w}}) \\ \vdots \\ \mathbf{G}^{\frac{1}{2}} (\mathbf{w}_N - \check{\mathbf{w}}) \end{bmatrix} \right\| \leq \|\mathbf{P}\| \left\| \begin{bmatrix} \mathbf{G}^{\frac{1}{2}} (\mathbf{w}_1 - \check{\mathbf{w}}) \\ \vdots \\ \mathbf{G}^{\frac{1}{2}} (\mathbf{w}_N - \check{\mathbf{w}}) \end{bmatrix} \right\| \\
 &= \|\underline{\mathbf{w}} - \underline{\check{\mathbf{w}}}\|_{\underline{\mathbf{G}}} \tag{E.6}
 \end{aligned}$$

From (E.6), our claim is proved.  $\square$

First of all, given a convex function  $\Theta : \mathbb{R}^m \rightarrow \mathbb{R}$ , with non-empty level set, where the level set is defined  $\text{lev}_{\leq 0} \Theta := \{\mathbf{w} \in \mathbb{R}^m : \Theta(\mathbf{w}) \leq 0\}$ , let us define the subgradient projection mapping, as follows  $T_{\Theta}^{(\mathbf{G})} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  [151]:

$$T_{\Theta}^{(\mathbf{G})}(\mathbf{w}) := \begin{cases} \mathbf{w} - \frac{\Theta(\mathbf{w})}{\|\Theta'(\mathbf{w})\|_{\mathbf{G}}^2} \Theta'(\mathbf{w}), & \mathbf{w} \notin \text{lev}_{\leq 0} \Theta \\ \mathbf{w}, & \mathbf{w} \in \text{lev}_{\leq 0} \Theta, \end{cases}$$

where  $\Theta'(\mathbf{w})$  is any subgradient of  $\Theta$ , at  $\mathbf{w}$ . Similarly, we define the relaxed subgradient projection mapping,  $T_{\Theta, \lambda}^{(\mathbf{G})}(\mathbf{w}) := I + \lambda(T_{\Theta}^{(\mathbf{G})}(\mathbf{w}) - I)$ ,  $\lambda \in (0, 2)$ , where  $I$  is the identity mapping.

Now, given a non-empty closed convex set, say  $\mathcal{C} \subset \mathbb{R}^m$ , and a convex function  $\Theta : \mathbb{R}^m \rightarrow \mathbb{R}$ , such that  $\mathcal{C} \cap \text{lev}_{\leq 0} \Theta \neq \emptyset$  it holds that [151]:

$$\forall \mathbf{w} \in \mathbb{R}^m, \forall \hat{\mathbf{w}} \in \mathcal{C} \cap \text{lev}_{\leq 0} \Theta : \frac{2 - \lambda}{2} \|\mathbf{w} - P_{\mathcal{C}} T_{\Theta, \lambda}^{(\mathbf{G})}(\mathbf{w})\|_{\mathbf{G}}^2 \leq \|\mathbf{w} - \hat{\mathbf{w}}\|_{\mathbf{G}}^2 - \|P_{\mathcal{C}} T_{\Theta, \lambda}^{(\mathbf{G})}(\mathbf{w}) - \hat{\mathbf{w}}\|_{\mathbf{G}}^2. \tag{E.7}$$

Following similar steps as in [82], it can be proved that  $\forall n \geq z_0$ ,  $\phi_{k,n} \in \text{lev}_{\leq 0} \Theta_{k,n} \Leftrightarrow \mathcal{I}_{k,n} = \emptyset$  and  $\forall n \geq z_0$ ,  $\phi_{k,n} \notin \text{lev}_{\leq 0} \Theta_{k,n} \Leftrightarrow \mathcal{I}_{k,n} \neq \emptyset$ . Moreover,  $\text{lev}_{\leq 0} \Theta_{k,n} = \bigcap_{j \in \mathcal{I}_{k,n}} S_{k,j} \supset \Omega_n \supset \Omega$ .

Recall the definition of the relaxed projection mapping; it can be readily seen that  $\mathbf{w}_{k,n+1} = P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathcal{G})} T_{\Theta_{k,n}, \lambda_{k,n}}^{(\mathcal{G})}(\boldsymbol{\phi}_{k,n})$ . Exploiting this fact, under Assumptions (a), (b), and (E.7) we have that

$\forall n \geq z_0, \forall k \in \mathcal{N}, \forall \hat{\mathbf{w}} \in \Omega :$

$$\begin{aligned} 0 &\leq \frac{2 - \lambda_{k,n}}{2} \|\boldsymbol{\phi}_{k,n} - \mathbf{w}_{k,n+1}\|_{\mathcal{G}}^2 = \frac{2 - \lambda_{k,n}}{2} \|\boldsymbol{\phi}_{k,n} - P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathcal{G})} T_{\Theta_{k,n}, \lambda_{k,n}}^{(\mathcal{G})}(\boldsymbol{\phi}_{k,n})\|_{\mathcal{G}}^2 \\ &\leq \|\boldsymbol{\phi}_{k,n} - \hat{\mathbf{w}}\|_{\mathcal{G}}^2 - \|P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathcal{G})} T_{\Theta_{k,n}, \lambda_{k,n}}^{(\mathcal{G})}(\boldsymbol{\phi}_{k,n}) - \hat{\mathbf{w}}\|_{\mathcal{G}}^2. \end{aligned} \quad (\text{E.8})$$

Recalling the definitions  $\underline{\mathbf{w}}_n = [\mathbf{w}_{1,n}^T, \dots, \mathbf{w}_{N,n}^T]^T \in \mathbb{R}^{Nm}$ ,  $\mathbf{P}\underline{\mathbf{w}}_n = [\boldsymbol{\phi}_{1,n}^T, \dots, \boldsymbol{\phi}_{N,n}^T]^T \in \mathbb{R}^{Nm}$ , and (E.8), we have

$\forall n \geq z_0, \forall \hat{\mathbf{w}} \in \mathfrak{C} :$

$$\begin{aligned} 0 &\leq \min_k \left\{ \frac{2 - \lambda_{k,n}}{2} \right\} \|\mathbf{P}\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_{n+1}\|_{\mathcal{G}}^2 \\ &\leq \|\mathbf{P}\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\mathcal{G}}^2 - \|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\mathcal{G}}^2. \end{aligned} \quad (\text{E.9})$$

Nevertheless, from Claim 2, the previous inequality can be rewritten

$$0 \leq \|\mathbf{P}\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\mathcal{G}}^2 - \|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\mathcal{G}}^2 \leq \|\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\mathcal{G}}^2 - \|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\mathcal{G}}^2.$$

Hence,

$$\forall n \geq z_0, \forall \hat{\mathbf{w}} \in \mathfrak{C} : \|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\mathcal{G}}^2 \leq \|\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\mathcal{G}}^2, \quad (\text{E.10})$$

which completes our proof.

## E.2 Asymptotic optimality

A well known property of the projection operator (see for example [151]), is the non-expansivity, *i.e.*, given a non-empty set  $\mathcal{C}$ ,  $\|P_{\mathcal{C}}^{(\mathcal{G})}(\mathbf{w}_1) - P_{\mathcal{C}}^{(\mathcal{G})}(\mathbf{w}_2)\|_{\mathcal{G}} \leq \|\mathbf{w}_1 - \mathbf{w}_2\|_{\mathcal{G}}, \forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^m$ . Recall the definition of the algorithm given in (E.2). Then,  $\forall k \in \mathcal{N}, \forall n \geq z_0, \forall \hat{\mathbf{w}} \in \Omega$ ,

we have

$$\begin{aligned}
 \|\mathbf{w}_{k,n+1} - \hat{\mathbf{w}}\|_{\mathcal{G}} &= \left\| P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathcal{G})} \left( \phi_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\mathcal{G}}^2} \Theta'_{k,n}(\phi_{k,n}) \right) - \hat{\mathbf{w}} \right\|_{\mathcal{G}} \\
 &= \left\| P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathcal{G})} \left( \phi_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\mathcal{G}}^2} \Theta'_{k,n}(\phi_{k,n}) \right) - P_{B_{\ell_1}[\mathbf{h}_n, \delta]}^{(\mathcal{G})}(\hat{\mathbf{w}}) \right\|_{\mathcal{G}} \\
 &\leq \left\| \phi_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\mathcal{G}}^2} \Theta'_{k,n}(\phi_{k,n}) - \hat{\mathbf{w}} \right\|_{\mathcal{G}}, \tag{E.11}
 \end{aligned}$$

where the equality in the second line holds since, by definition,  $\hat{\mathbf{w}} \in \Omega \subset B_{\ell_1}[\mathbf{h}_n, \delta]$  and the inequality, from the non-expansivity of the projection operator. Assuming that  $\Theta'_{k,n}(\phi_{k,n}) \neq \mathbf{0}_m, \forall k \in \mathcal{N}$ , and rewriting (E.11) for all the nodes of the network we have

$$\begin{aligned}
 \|\underline{\mathbf{w}}_{n+1} - \underline{\hat{\mathbf{w}}}\|_{\underline{\mathcal{G}}}^2 &\leq \left\| \begin{bmatrix} \phi_{1,n} - \lambda_{1,n} \frac{\Theta_{1,n}(\phi_{1,n})}{\|\Theta'_{1,n}(\phi_{1,n})\|_{\mathcal{G}}^2} \Theta'_{1,n}(\phi_{1,n}) \\ \vdots \\ \phi_{N,n} - \lambda_{N,n} \frac{\Theta_{N,n}(\phi_{N,n})}{\|\Theta'_{N,n}(\phi_{N,n})\|_{\mathcal{G}}^2} \Theta'_{N,n}(\phi_{N,n}) \end{bmatrix} - \underline{\hat{\mathbf{w}}} \right\|_{\underline{\mathcal{G}}}^2 \\
 &= \left\| \begin{bmatrix} \phi_{1,n} - \hat{\mathbf{w}} \\ \vdots \\ \phi_{N,n} - \hat{\mathbf{w}} \end{bmatrix} \right\|_{\underline{\mathcal{G}}}^2 + \sum_{k \in \mathcal{N}} \lambda_{k,n}^2 \frac{(\Theta_{k,n}(\phi_{k,n}))^2}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\mathcal{G}}^2} \\
 &\quad - 2 \sum_{k \in \mathcal{N}} \lambda_{k,n} \frac{\Theta_{k,n}(\phi_{k,n}) \langle \Theta'_{k,n}(\phi_{k,n}), (\phi_{k,n} - \hat{\mathbf{w}}) \rangle_{\mathcal{G}}}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\mathcal{G}}^2}. \tag{E.12}
 \end{aligned}$$

Nevertheless,

$$\left\| \begin{bmatrix} \phi_{1,n} - \hat{\mathbf{w}} \\ \vdots \\ \phi_{N,n} - \hat{\mathbf{w}} \end{bmatrix} \right\|_{\underline{\mathcal{G}}} = \|\mathbf{P}\underline{\mathbf{w}}_n - \underline{\hat{\mathbf{w}}}\|_{\underline{\mathcal{G}}} \leq \|\underline{\mathbf{w}}_n - \underline{\hat{\mathbf{w}}}\|_{\underline{\mathcal{G}}}. \tag{E.13}$$

From the definition of the subgradient, we have

$$\langle \Theta'_{k,n}(\phi_{k,n}), (\phi_{k,n} - \hat{\mathbf{w}}) \rangle_{\mathcal{G}} \geq \Theta_{k,n}(\phi_{k,n}) - \Theta_{k,n}(\hat{\mathbf{w}}) = \Theta'_{k,n}(\phi_{k,n}), \tag{E.14}$$

where the last equation, holds due to the fact that  $\hat{\mathbf{w}} \in \Omega \Leftrightarrow \Theta'_{k,n}(\hat{\mathbf{w}}) = \mathbf{0}$ . Taking (E.13) and (E.14) into consideration, we obtain

$$\|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}^2 \leq \|\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}^2 - \sum_{k \in \mathcal{N}} \lambda_{k,n} (2 - \lambda_{k,n}) \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\underline{\mathbf{G}}}^2}. \quad (\text{E.15})$$

Here, notice that the sequence  $\|\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}$  is bounded and monotone decreasing, hence it converges. The latter fact implies that

$$\lim_{n \rightarrow \infty} \left( \|\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}} - \|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}} \right) = 0. \quad (\text{E.16})$$

Under Assumption (c), (E.15) can be rewritten

$$\sum_{k \in \mathcal{N}} \varepsilon_1^2 \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\underline{\mathbf{G}}}^2} \leq \sum_{k \in \mathcal{N}} \lambda_{k,n} (2 - \lambda_{k,n}) \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\underline{\mathbf{G}}}^2} \leq \|\underline{\mathbf{w}}_n - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}^2 - \|\underline{\mathbf{w}}_{n+1} - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}^2. \quad (\text{E.17})$$

Taking limits in (E.17) and recalling (E.16) we have that

$$\lim_{n \rightarrow \infty} \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\underline{\mathbf{G}}}^2} = 0, \quad \forall k \in \mathcal{N}.$$

If we follow similar steps as in [82], it can be verified that  $\forall n \in \mathbb{Z}_{\geq 0}, \forall k \in \mathcal{N}, \forall \mathbf{w} \in \mathbb{R}^m : \|\Theta'_{k,n}(\mathbf{w})\|_{\underline{\mathbf{G}}} \leq 1$ . So, if  $\Theta'_{k,n}(\phi_{k,n}) \neq \mathbf{0}_m$

$$\Theta_{k,n}(\phi_{k,n}) \leq \frac{\Theta_{k,n}(\phi_{k,n})}{\|\Theta'_{k,n}(\phi_{k,n})\|_{\underline{\mathbf{G}}}^2} \rightarrow 0, \quad n \rightarrow \infty. \quad (\text{E.18})$$

Obviously, recalling the previous discussion,  $\Theta'_{k,n}(\phi_{k,n}) = \mathbf{0}_m \Leftrightarrow \Theta_{k,n}(\phi_{k,n}) = 0, \forall n \geq z_0$ . Combining this fact together with (E.18), we have that

$$\forall k \in \mathcal{N}, \lim_{n \rightarrow \infty} \Theta_{k,n}(\phi_{k,n}) = 0. \quad (\text{E.19})$$

Now, following similar steps as in [82], it can be shown that there exists  $D > 0$  such that  $L_{k,n} \leq D, \forall k \in \mathcal{N}, \forall n \in \mathbb{Z}_{\geq 0}$ . From the definition of  $\Theta_{k,n}$ , and under Assumption (d), we

## Proof of Theorem 1

---

have  $\forall k \in \mathcal{N}$

$$\begin{aligned}
\frac{D}{\tilde{\omega}_k} \Theta_{k,n}(\phi_{k,n}) &\geq \frac{D}{\tilde{\omega}_k} \sum_{j \in \mathcal{J}_n} \omega_{k,j} \frac{d_{(\mathbf{G})}^2(\phi_{k,n}, S_{k,j})}{L_{k,n}} \\
&\geq \frac{D}{\tilde{\omega}_k} \frac{\tilde{\omega}_k}{D} \sum_{j \in \mathcal{J}_n} d_{(\mathbf{G})}^2(\phi_{k,n}, S_{k,j}) \\
&\geq \max\{d_{(\mathbf{G})}^2(\phi_{k,n}, S_{k,j}) : j \in \mathcal{J}_n\}.
\end{aligned}$$

Taking limits in the previous inequality, we obtain that

$$\lim_{n \rightarrow \infty} \max\{d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j}) : j \in \mathcal{J}_n\} = 0. \quad (\text{E.20})$$

Combining (E.9) with the result of Claim 2, we have

$$\begin{aligned}
\forall n \geq z_0, \forall \hat{\mathbf{w}} \in \mathfrak{C} : \\
0 &\leq \min_k \left\{ \frac{2 - \lambda_{k,n}}{2} \right\} \|\mathbf{P}_n \mathbf{w}_n - \mathbf{w}_{n+1}\|_{\underline{\mathbf{G}}}^2 \\
&\leq \|\mathbf{w}_n - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}^2 - \|\mathbf{w}_{n+1} - \hat{\mathbf{w}}\|_{\underline{\mathbf{G}}}^2.
\end{aligned} \quad (\text{E.21})$$

Taking limits in (E.21) and recalling (E.16) gives us

$$\lim_{n \rightarrow \infty} \|\mathbf{P}_n \mathbf{w}_n - \mathbf{w}_{n+1}\|_{\underline{\mathbf{G}}}^2 = 0 \Leftrightarrow \lim_{n \rightarrow \infty} \sum_{k \in \mathcal{N}} \|\phi_{k,n} - \mathbf{w}_{k,n+1}\|_{\underline{\mathbf{G}}}^2 = 0. \quad (\text{E.22})$$

Fix an arbitrary point  $\mathbf{v} \in S_{k,j}$ ,  $\forall k \in \mathcal{N}$ ,  $\forall j \in \mathcal{J}_n$ . Then from the triangle inequality we have

$$\begin{aligned}
\|\mathbf{w}_{k,n+1} - \mathbf{v}\|_{\mathbf{G}} &\leq \|\mathbf{w}_{k,n+1} - \phi_{k,n}\|_{\mathbf{G}} + \|\phi_{k,n} - \mathbf{v}\|_{\mathbf{G}} \Rightarrow \\
\inf_{\mathbf{v} \in S_{k,j}} \|\mathbf{w}_{k,n+1} - \mathbf{v}\|_{\mathbf{G}} &\leq \|\mathbf{w}_{k,n+1} - \phi_{k,n}\|_{\mathbf{G}} + \inf_{\mathbf{v} \in S_{k,j}} \|\phi_{k,n} - \mathbf{v}\|_{\mathbf{G}} \Rightarrow \\
d_{(\mathbf{G})}(\mathbf{w}_{k,n+1}, S_{k,j}) &\leq \|\mathbf{w}_{k,n+1} - \phi_{k,n}\|_{\mathbf{G}} + d_{(\mathbf{G})}(\phi_{k,n}, S_{k,j})
\end{aligned} \quad (\text{E.23})$$

If we take limits in (E.23), from (E.20) and (E.22), it can be seen that

$$\lim_{n \rightarrow \infty} d_{(\mathbf{G})}(\mathbf{w}_{k,n+1}, S_{k,j}) = 0, \forall k \in \mathcal{N}, \forall j \in \mathcal{J}_n \Leftrightarrow \lim_{n \rightarrow \infty} \sum_{j \in \mathcal{J}_n} d_{(\mathbf{G})}(\mathbf{w}_{k,n+1}, S_{k,j}) = 0, \forall k \in \mathcal{N}. \quad (\text{E.24})$$

The definitions of the distance function and the projection operator, yield

$$\begin{aligned} d(\mathbf{w}_{k,n+1}, S_{k,j}) &= \|\mathbf{w}_{k,n+1} - P_{S_{k,j}}(\mathbf{w}_{k,n+1})\| \\ &\leq \|\mathbf{w}_{k,n+1} - P_{S_{k,j}}^{(\mathbf{G})}(\mathbf{w}_{k,n+1})\|. \end{aligned} \quad (\text{E.25})$$

Nevertheless, the Rayleigh-Ritz theorem implies [67]  $\forall \mathbf{w} \in \mathbb{R}^m : \|\mathbf{w}\| \leq \tau_{\min}^{-\frac{1}{2}} \|\mathbf{w}\|_{\mathbf{G}}$ , where  $\tau_{\min}$  is the smallest eigenvalue of  $\mathbf{G}$ . Combining this fact as well as (E.25) we obtain

$$\begin{aligned} d(\mathbf{w}_{k,n+1}, S_{k,j}) &\leq \|\mathbf{w}_{k,n+1} - P_{S_{k,j}}^{(\mathbf{G})}(\mathbf{w}_{k,n+1})\| \\ &\leq \tau_{\min}^{-\frac{1}{2}} \|\mathbf{w}_{k,n+1} - P_{S_{k,j}}^{(\mathbf{G})}(\mathbf{w}_{k,n+1})\|_{\mathbf{G}} \rightarrow 0, n \rightarrow \infty, \forall k \in \mathcal{N}, \end{aligned} \quad (\text{E.26})$$

where the limit holds from (E.24). From the previous, it is not difficult to obtain that

$$\lim_{n \rightarrow \infty} \max\{d(\mathbf{w}_{k,n+1}, S_{k,j}) : j \in \mathcal{J}_n\} = 0,$$

which completes our proof.

## E.3 Asymptotic Consensus

In [31] it has been proved, that the algorithmic scheme achieves asymptotic consensus, *i.e.*,  $\|\mathbf{w}_{k,n} - \mathbf{w}_{l,n}\| \rightarrow 0, n \rightarrow \infty, \forall k, l \in \mathcal{N}$  if and only if

$$\lim_{n \rightarrow \infty} \|\underline{\mathbf{w}}_n - P_{\mathcal{O}}(\underline{\mathbf{w}}_n)\| = 0. \quad (\text{E.27})$$

Let Assumptions (a), (b), (c), (d), hold true. We define the following quantity

$$\underline{\boldsymbol{\epsilon}}_n := \underline{\mathbf{w}}_{n+1} - \mathbf{P}\underline{\mathbf{w}}_n. \quad (\text{E.28})$$

Obviously from (E.22)

$$\lim_{n \rightarrow \infty} \underline{\boldsymbol{\epsilon}}_n = \mathbf{0} \quad (\text{E.29})$$

Now, if we rearrange the terms in (E.28) and if we iterate the resulting equation, we have:

$$\begin{aligned} \underline{\boldsymbol{w}}_{n+1} &= \mathbf{P}\underline{\boldsymbol{w}}_n + \underline{\boldsymbol{\epsilon}}_n \\ &= \mathbf{P}\mathbf{P}\underline{\boldsymbol{w}}_{n-1} + \mathbf{P}_n\underline{\boldsymbol{\epsilon}}_{n-1} + \underline{\boldsymbol{\epsilon}}_n = \dots \\ &= \prod_{i=1}^n \mathbf{P}\underline{\boldsymbol{w}}_0 + \sum_{j=1}^n \prod_{l=0}^{n-j} \mathbf{P}\underline{\boldsymbol{\epsilon}}_{j-1} + \underline{\boldsymbol{\epsilon}}_n \end{aligned}$$

If we left-multiply the previous equation by  $(\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T)$  and follow similar steps as in [31, Lemma 2] it can be verified that  $\lim_{n \rightarrow \infty} \|(\mathbf{I}_{Nm} - \mathbf{B}\mathbf{B}^T)\underline{\boldsymbol{w}}_{n+1}\| = \lim_{n \rightarrow \infty} \|\underline{\boldsymbol{w}}_{n+1} - P_{\mathcal{O}}(\underline{\boldsymbol{w}}_{n+1})\| = 0$  which completes our proof.

## E.4 Strong Convergence

We will prove, that under assumptions (a), (b), (c), (d), (e),  $\lim_{n \rightarrow \infty} \underline{\boldsymbol{w}}_n = \hat{\underline{\boldsymbol{w}}}_{\mathcal{O}}, \hat{\underline{\boldsymbol{w}}}_* \in \mathcal{O}$ . Recall that the projection operator, of an arbitrary vector  $\underline{\boldsymbol{w}} \in \mathbb{R}^{Nm}$  onto the consensus subspace equals to  $P_{\mathcal{O}}(\underline{\boldsymbol{w}}) = \mathbf{B}\mathbf{B}^T\underline{\boldsymbol{w}}, \forall \underline{\boldsymbol{w}} \in \mathbb{R}^{Nm}$ . Taking into consideration Assumption (e) together with (E.10), from [149, Lemma 1] we have that there exists  $\hat{\underline{\boldsymbol{w}}}_{\mathcal{O}} \in \mathcal{O}$  such that

$$\lim_{n \rightarrow \infty} P_{\mathcal{O}}(\underline{\boldsymbol{w}}_n) = \hat{\underline{\boldsymbol{w}}}_{\mathcal{O}}. \quad (\text{E.30})$$

Now, exploiting the triangle inequality we have that

$$\|\underline{\boldsymbol{w}}_n - \hat{\underline{\boldsymbol{w}}}_{\mathcal{O}}\| \leq \|\underline{\boldsymbol{w}}_n - P_{\mathcal{O}}(\underline{\boldsymbol{w}}_n)\| + \|\hat{\underline{\boldsymbol{w}}}_{\mathcal{O}} - P_{\mathcal{O}}(\underline{\boldsymbol{w}}_n)\| \rightarrow 0, \quad n \rightarrow \infty, \quad (\text{E.31})$$

where this limit holds from (I.14) and (E.30). The proof is complete since (E.31) implies that  $\lim_{n \rightarrow \infty} \underline{\boldsymbol{w}}_n = \hat{\underline{\boldsymbol{w}}}_{\mathcal{O}}$ .

## Chapter 6

# Dimensionality Reduction in Distributed Adaptive Learning via Krylov Subspaces

*In this chapter, the problem of dimensionality reduction in adaptive distributed learning is studied. We consider a network obeying the ad-hoc topology, in which the nodes sense an amount of data and cooperate with each other, by exchanging information, in order to estimate an unknown, parameter vector, which is common to all nodes. As in the previous chapters, the algorithm, to be presented here, is based on the APSM algorithmic family. At each time instant and at each node of the network, a hyperslab is constructed based on the received measurements and this defines the region in which the solution is searched for. Moreover, in order to reduce the number of transmitted coefficients, which is dictated by the dimension of the unknown vector, we seek for possible solutions in a subspace of lower dimensionality; the technique will be developed around the Krylov subspace rationale. Our goal is to find a point that belongs to the intersection of this infinite number of hyperslabs and the respective Krylov subspaces. This is achieved via a sequence of projections onto the property sets as well as the Krylov subspaces. Finally, the case of highly correlated inputs, which, usually, degrades the performance of an algorithm is also considered. This is bypassed via a transformation which whitens the input. The proposed schemes are brought in a decentralized form by adopting the combine-adapt cooperation strategy among the nodes.*

## 6.1 Problem Formulation

We consider a network consisting of  $N$  spatially distributed nodes. Our task is to estimate an unknown parameter vector of interest,  $\mathbf{w}_* \in \mathbb{R}^m$ , through measurements  $(d_{k,n}, \mathbf{u}_{k,n}) \in \mathbb{R} \times \mathbb{R}^m$ , which are related according to the linear model

$$d_{k,n} = \mathbf{u}_{k,n}^T \mathbf{w}_* + v_{k,n}, \quad \forall n \in \mathbb{Z}_{\geq 0}, \quad \forall k \in \mathcal{N}, \quad (6.1)$$

where  $\mathcal{N}$  denotes the node set:  $\mathcal{N} = \{1, \dots, N\}$  and  $v_{k,n}$  is the additive noise process with variance equal to  $\sigma_k^2$ ,  $\forall k \in \mathcal{N}$ . As we have already discussed in the previous chapters, the unknown parameter estimation process can be benefited by the node cooperation; that is, the nodes exchange estimates and/or measurements with their neighbors and exploit them accordingly. Obviously, this type of cooperation demands that at every time instant each node will transmit a number of coefficients, which is proportionate to the dimension of the vector to be estimated. In applications where this dimension is large, the exchange of information among the nodes can be a burden. In the current study, in order to achieve *dimensionality reduction* and consequently to reduce the number of transmitted coefficients, the reduced rank adaptive filtering rationale is adopted. Algorithms whose goal is to reduce the amount of transmitted information, by performing dimensionality reduction, have been proposed in the context of distributed quantized Kalman Filtering [102, 145], and quantized consensus algorithms, e.g., [108]. Finally, reduced rank algorithms able for adaptive operation in ad-hoc networks, following the diffusion optimization strategy, were proposed in [42, 43] and will be presented in this chapter.

The basic concept of our reduced rank adaptive filtering task can be summarized as follows: instead of seeking for the unknown vector in the original space, one seeks for the projection of it onto a lower dimension subspace. Via this procedure, the obtained estimates are optimally forced in a lower dimension space, and each node transmits fewer coefficients than the ones originally needed, in the case where the full dimensionality of the unknown vector was exploited. Here, the associated subspaces are the so-called Krylov subspaces, constructed by exploiting the statistics of the sensed information. The Krylov subspaces have been used in several applications, e.g., in the reduced rank adaptive filtering [155], in the Multistage Nested Wiener Filter [73], in the auxiliary vector filtering, [76], etc.

### 6.1.1 Krylov Subspaces and the Reduced Rank Wiener Solution

Our kickoff point is the Wiener filtering task. Throughout this section, the notational dependence on the nodes is suppressed for simplicity purposes, since the results hold true for all nodes. It can be shown, *e.g.*, [120], that the solution that minimizes the mean-square error (MSE), *i.e.*,  $\mathbb{E}[(d_n - \mathbf{u}_n^T \mathbf{w})^2]$ , where  $d_n$ ,  $\mathbf{u}_n$  are related via (6.1), satisfies the celebrated Wiener-Hopf equation, given by

$$\mathbf{p} = \mathbf{R}\mathbf{w}, \quad (6.2)$$

where the  $m \times m$  matrix  $\mathbf{R} = \mathbb{E}[\mathbf{u}_n \mathbf{u}_n^T]$  is the so-called input autocorrelation matrix, and the vector  $\mathbf{p} = \mathbb{E}[d_n \mathbf{u}_n]$  is the crosscorrelation vector between the input and the desired response. If the matrix  $\mathbf{R}$  is invertible, which is usually the case, then the solution of (6.2) is the unknown vector  $\mathbf{w}_*$ , *e.g.*, [64]. Our main goal, here, is to use the Wiener MSE solution in its constrained form. Since our objective is to reduce dimensionality, we are going to search for the filter that minimizes the MSE and at the same time lies in a lower dimension subspace. This brings Krylov spaces into the scene.

Given an  $m \times m$  matrix  $\mathbf{A}$  and a vector  $\mathbf{w} \in \mathbb{R}^m$ , the Krylov subspace of dimension  $D < m$  is defined as  $K_D(\mathbf{A}, \mathbf{w}) := \text{span}\{\mathbf{w}, \mathbf{A}\mathbf{w}, \dots, \mathbf{A}^{D-1}\mathbf{w}\}$ . The Krylov subspaces play a central role and they have been employed in the reduced rank adaptive filtering task, *e.g.*, [66, 155]. It has been observed that they provide a good trade-off between the dimensionality reduction and the performance of the developed algorithms, due to their strong connection with the Wiener solution. In the sequel, we will comment on the physical reasoning of these subspaces. Following a similar rationale as in [61] and in [66], we denote by  $\mathbf{w}_{WF}^{(D)} \in \mathbb{R}^m$  the solution of the Wiener-Hopf equation in the Krylov subspace,  $K_D(\mathbf{R}, \mathbf{p})$ . In words,  $\mathbf{w}_{WF}^{(D)}$  is the vector we obtain if we solve the Wiener-Hopf equation and constraint the solution to lie inside  $K_D(\mathbf{R}, \mathbf{p})$ . This vector is the optimum one, in the MSE sense, which belongs to this subspace, *e.g.*, [66]. Moreover, it has an elegant geometrical property; it is the projection of  $\mathbf{w}_*$  with respect to the  $\mathbf{R}$ -norm (see also Chapter 5 and Appendix F) onto  $K_D(\mathbf{R}, \mathbf{p})$ , *i.e.*,  $\mathbf{w}_{WF}^{(D)} = P_{K_D(\mathbf{R}, \mathbf{p})}^{(\mathbf{R})}(\mathbf{w}_*)$ , where the operator  $P_{K_D(\mathbf{R}, \mathbf{p})}^{(\mathbf{R})}(\mathbf{w}_*)$  stands for the previously mentioned projection. Analytically, it is given by [155]:

$$\mathbf{w}_{WF}^{(D)} = \mathbf{T}(\mathbf{T}^T \mathbf{R} \mathbf{T})^{-1} \mathbf{T}^T \mathbf{p} = \mathbf{T}(\mathbf{T}^T \mathbf{R} \mathbf{T})^{-1} \mathbf{T}^T \mathbf{R} \mathbf{w}_*,$$

## Dimensionality Reduction in Distributed Adaptive Learning via Krylov Subspaces

---

where  $\mathbf{T} \in \mathbb{R}^{m \times D}$  is a matrix whose columns form an orthonormal basis for the subspace  $K_D(\mathbf{R}, \mathbf{p})$ .

Now, let us examine one more viewpoint which clarifies the connection between  $\mathbf{w}_{WF}^{(D)}$  and  $\mathbf{w}_*$ . Our starting point will be the MultiStage Nested Wiener Filter (MSNWF), proposed in [61]. Put in general terms, the MSNWF solves the Wiener-Hopf equation, without inversion of the matrix  $\mathbf{R}$ . The MSNWF consists of  $m$  filters,  $\mathbf{t}_i \in \mathbb{R}^m$ ,  $i = 1, \dots, m$ , which produce  $m$  outputs  $d_i[n] = \mathbf{t}_i^T \mathbf{u}_n$ ,  $i = 1, \dots, m$ , and they are computed via the following optimization

$$\begin{aligned} i = 2, \dots, m, \quad \mathbf{t}_i = \arg \max_{\mathbf{t}} \{ \mathbf{t}^T \mathbf{R} \mathbf{t}_{i-1} \} &= \arg \max_{\mathbf{t}} \mathbb{E} \{ d_i[n] d_{i-1}[n] \} & (6.3) \\ \text{s.t.} \quad \mathbf{t}^T \mathbf{t} &= 1 \\ \mathbf{t}^T \mathbf{t}_r &= 0, \quad r = 1, \dots, i-1, \end{aligned}$$

and  $\mathbf{t}_1$  occurs by maximization of  $\mathbf{t}_1 = \arg \max_{\mathbf{t}} \mathbb{E} \{ \mathbf{t}^T \mathbf{u}_n d_n \} = \arg \max_{\mathbf{t}} \mathbb{E} \{ d_1[n] d_n \}$ , s.t.  $\mathbf{t}^T \mathbf{t} = 1$ . The physical reasoning of the previous optimization problem can be summarized as follows. The first filter  $\mathbf{t}_1$  is obtained so as to maximize the correlation of the output  $d_1[n]$  and the desired one  $d_n$ . The  $i$ -th filter is computed in a similar notion, which is the maximization of the correlation between the current and the previous outputs, *i.e.*,  $d_i[n]$  and  $d_{i-1}[n]$ . Furthermore, as it can be seen by (6.3), we restrict the filters to be orthonormal. It has been proved, *e.g.*, in [73], that the  $m$ -th output response occurring by the MSNWF, equals to the one occurring by the unknown vector, *i.e.*,  $\hat{d}_n = \mathbf{u}_n^T \mathbf{w}_*$ .

It is very interesting to see what happens if one stops the iterations in (6.3), at step  $D$ . It turns out that the obtained solution corresponds to the reduced rank Wiener Filter (WF),  $\mathbf{w}_{WF}^{(D)}$ . Moreover, as it has been proved, *e.g.*, [61], the filters  $\mathbf{t}_i$ ,  $i = 1, \dots, D$ , form a basis in the Krylov subspace; in other words, if we group them in a matrix, we obtain the matrix  $\mathbf{T}$ .

Now, let us see how the previous arguments can be employed in the adaptive filtering task. As we have already mentioned, in the reduced rank adaptive filtering, instead of seeking for the unknown solution, which in our case is  $\mathbf{w}_*$ , one seeks for the projection of it onto a subspace of reduced dimension; in our case this is the projection, in the  $\mathbf{R}$  norm sense, onto  $K_D(\mathbf{R}, \mathbf{p})$ . Obviously, the fact that instead of tracking  $\mathbf{w}_*$ , one tracks for its projection in a subspace of lower dimension, results at an increased error floor in the steady state, which depends on the distance between the true solution and the reduced rank one. These issues

will be clarified in the sequel.

A natural question rising is how accurately can  $\mathbf{w}_*$  be identified by employing the Krylov subspace rationale. It has been proved, *e.g.*, [155], that

$$\|\mathbf{w}_* - \mathbf{w}_{WF}^{(D)}\| \leq 2\tau_{\min}^{-1/2} \mathbf{w}_*^T \mathbf{R} \mathbf{w}_* \alpha_\kappa^D, \quad (6.4)$$

where  $\tau_{\min}$  is the smallest eigenvalue of the matrix  $\mathbf{R}$ ,  $\alpha_\kappa := (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$  with  $\kappa := \|\mathbf{R}\| \|\mathbf{R}^{-1}\| \geq 1$ . From the previous findings, it can be readily observed that the input statistics play a central role in the performance of the algorithms built around the Krylov subspaces. More specifically, if the eigenvalue spread of the matrix  $\mathbf{R}$  is large, which yields a large value of  $\alpha_\kappa$ , the upper bound in the previous inequality is larger, and it has also been experimentally verified that the performance of the respective algorithm is degraded.

### 6.1.2 Set-theoretic estimation: the reduced rank case

According to the discussion in the previous chapters, the property sets are constructed so as to contain the unknown vector  $\mathbf{w}_*$  with a high probability. The question now is which strategy to follow in the case of reduced rank scenarios. Our kick off point will be the reduced rank Wiener solution. More specifically, the property sets will be constructed so as to contain the vector  $\mathbf{w}_{WF}^{(D)}$  with a high probability. As it will become clear later on, this can be guaranteed by seeking for points that lie in the intersection of the hyperslabs and the Krylov subspace, *i.e.*,  $K_D(\mathbf{R}, \mathbf{p})$ . Let us define the set  $\bar{S}_n := S_n \cap K_D(\mathbf{R}, \mathbf{p}) = \{\mathbf{w} \in K_D(\mathbf{R}, \mathbf{p}) : |d_n - \mathbf{u}_n^T \mathbf{w}| \leq \epsilon\}$ , where  $S_n$  is the hyperslab constructed via  $d_n, \mathbf{u}_n$ . Recall from the previous discussion that  $\mathbf{w}_{WF}^{(D)} \in K_D(\mathbf{R}, \mathbf{p})$ . In order to have  $\mathbf{w}_{WF}^{(D)} \in \bar{S}_n$ , the following must hold true

$$|d_n - \mathbf{u}_n^T \mathbf{w}_{WF}^{(D)}| \leq \epsilon \Leftrightarrow |\mathbf{u}_n^T \mathbf{w}_* + v_n - \mathbf{u}_n^T \mathbf{w}_{WF}^{(D)}| \leq \epsilon \Leftrightarrow |\mathbf{u}_n^T (\mathbf{w}_* - \mathbf{w}_{WF}^{(D)}) + v_n| \leq \epsilon. \quad (6.5)$$

From (6.5), it can be seen that the parameter  $\epsilon$ , which determines the width of the hyperslab, determines the probability that  $\mathbf{w}_{WF}^{(D)} \in \bar{S}_n$ , in the sense that the larger the  $\epsilon$ , the larger the possibility that the previously mentioned condition will hold. Obviously, in the full rank case, in which the condition to be satisfied is  $\mathbf{w}_* \in S_n$ , the only term, which dictates the

## Dimensionality Reduction in Distributed Adaptive Learning via Krylov Subspaces

---

choice of  $\epsilon$ , is  $v_n$ . Hence, the width of the hyperslabs is chosen with respect to the statistics of the noise. In the reduced rank case, besides the noise, one has to take into consideration the term  $\mathbf{u}_n^T(\mathbf{w}_* - \mathbf{w}_{\text{WF}}^{(D)})$ . However, in practice, as it has been documented in [155], in cases where the eigenvalue spread of  $\mathbf{R}$  is close to 1, which implies that the distance between  $\mathbf{w}_*$  and  $\mathbf{w}_{\text{WF}}^{(D)}$  is small (see also (6.4)), the noise term is the dominant one. Hence, if the user-controlled parameter,  $\epsilon$ , is defined according to the noise statistics, the condition of having  $\mathbf{w}_{\text{WF}}^{(D)} \in \bar{S}_n$ , holds with a high probability. In the sequel, a technique appropriate for the case where the eigenvalue spread is large, will be proposed in order to overstep this limitation.

In order to construct the subspace, knowledge on the statistics of the input and the desired response, i.e.,  $\mathbf{R}, \mathbf{p}$ , is required. A reasonable strategy is to rely on estimates of the previously mentioned quantities. To this end, the autocorrelation is estimated via  $\hat{\mathbf{R}}_n := \sum_{j=0}^{n-1} \zeta^{n-1-j} \mathbf{u}_j \mathbf{u}_j^T$  and the crosscorrelation via  $\hat{\mathbf{p}}_n := \sum_{j=0}^{n-1} \zeta^{n-1-j} d_j \mathbf{u}_j$ , where  $\zeta \in (0, 1]$  is the so-called forgetting factor, employed in order to "forget" past values in time varying scenarios. The estimates,  $\hat{\mathbf{R}}_n, \hat{\mathbf{p}}_n$  are updated  $\forall n \in \mathbb{Z}_{\geq 0}$ , according to the following formulas:  $\hat{\mathbf{R}}_n = \zeta \hat{\mathbf{R}}_{n-1} + \mathbf{u}_{n-1} \mathbf{u}_{n-1}^T$  and  $\hat{\mathbf{p}}_n = \zeta \hat{\mathbf{p}}_{n-1} + d_{n-1} \mathbf{u}_{n-1}$ . Having obtained the estimates of  $\hat{\mathbf{R}}_n$  and  $\hat{\mathbf{p}}_n$ , our goal now is to develop the projection operator that projects an estimate to the intersection of the corresponding hyperslab and the current estimate of the Krylov subspace, i.e.,  $S_n \cap K_n$ , where  $K_n := K_D(\hat{\mathbf{R}}_n, \hat{\mathbf{p}}_n)$ .

**Claim 4.** *The projection of a vector lying in  $K_n$  onto  $S_n \cap K_n$  is given by*

$$\forall \mathbf{w} \in K_n : \quad P_{S_n \cap K_n}(\mathbf{w}) = \mathbf{w} + \tilde{\beta} \hat{\mathbf{T}}_n \hat{\mathbf{T}}_n^T \mathbf{u}_n, \quad (6.6)$$

where  $\hat{\mathbf{T}}_n$  is an  $m \times D$  matrix, whose columns form an orthonormal basis of  $K_n$  and

$$\tilde{\beta} = \begin{cases} \frac{d_n - \mathbf{w}^T \hat{\mathbf{T}}_n \hat{\mathbf{T}}_n^T \mathbf{u}_n + \epsilon}{\|\hat{\mathbf{T}}_n^T \mathbf{u}_n\|^2}, & \text{if } d_n - \mathbf{w}^T \hat{\mathbf{T}}_n \hat{\mathbf{T}}_n^T \mathbf{u}_n < -\epsilon, \\ 0, & \text{if } |d_n - \mathbf{w}^T \hat{\mathbf{T}}_n \hat{\mathbf{T}}_n^T \mathbf{u}_n| \leq \epsilon \\ \frac{d_n - \mathbf{w}^T \hat{\mathbf{T}}_n \hat{\mathbf{T}}_n^T \mathbf{u}_n - \epsilon}{\|\hat{\mathbf{T}}_n^T \mathbf{u}_n\|^2}, & \text{if } d_n - \mathbf{w}^T \hat{\mathbf{T}}_n \hat{\mathbf{T}}_n^T \mathbf{u}_n > \epsilon. \end{cases}$$

*Proof.* The proof is given in Appendix G. □

Now, let us see how the case where the denominator in the previous equation equals to

zero is treated. First of all, recall that the columns of  $\hat{\mathbf{T}}_n$  form a basis for the Krylov subspace. If  $\hat{\mathbf{T}}_n^T \mathbf{u}_n = \mathbf{0}_D$ , this means that the vector  $\mathbf{u}_n$  is perpendicular to the Krylov subspace. Moreover, it holds, *e.g.*, [133], that the vector  $\mathbf{u}_n$  is perpendicular to the hyperplanes  $H_{1,n} = \{\mathbf{w} \in \mathbb{R}^m : d_n - \mathbf{u}_n^T \mathbf{w} = -\epsilon\}$  and  $H_{2,n} = \{\mathbf{w} \in \mathbb{R}^m : d_n - \mathbf{u}_n^T \mathbf{w} = +\epsilon\}$ , which constitute the hyperplanes that define the hyperslab. These two facts imply that  $\hat{\mathbf{T}}_n^T \mathbf{u}_n = \mathbf{0}_D$  in the case where the subspace is “parallel” to the hyperslab. This case is treated as in the full rank case, *i.e.*, when the input vector is  $\mathbf{0}_m$ , *e.g.*, [134]. To be more specific, if such an input vector occurs, it is not taken into consideration in the algorithmic flow.

## 6.2 Proposed Scheme

First of all, it has to be pointed out, that despite the fact that the nodes seek for the same unknown vector, the input as well as the noise statistics differ from node to node. Hence, in contrast to the non-distributed scenario, here, we should take into consideration the statistics from all the nodes. Let us define the mean square error loss function  $\mathcal{L} : \mathbb{R}^m \rightarrow [0, +\infty)$ , for the whole network

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \frac{1}{N} \sum_{k \in \mathcal{N}} \mathbb{E} \{ (d_{k,n} - \mathbf{u}_{k,n}^T \mathbf{w})^2 \} \\ &= \frac{1}{N} \sum_{k \in \mathcal{N}} (\mathbf{w}^T \mathbf{R}_k \mathbf{w} - 2\mathbf{w}^T \mathbf{p}_k + \sigma_{d_k}^2) \\ &= \mathbf{w}^T \mathbf{R}' \mathbf{w} - 2\mathbf{w}^T \mathbf{p}' + \frac{1}{N} \sum_{k \in \mathcal{N}} \sigma_{d_k}^2, \end{aligned} \quad (6.7)$$

where  $\sigma_{d_k}^2 = E\{d_{k,n}^2\}$ ,  $\mathbf{R}' = \frac{1}{N} \sum_{k \in \mathcal{N}} E\{\mathbf{u}_{k,n} \mathbf{u}_{k,n}^T\} = \frac{1}{N} \sum_{k \in \mathcal{N}} \mathbf{R}_k$  and  $\mathbf{p}' = \frac{1}{N} \sum_{k \in \mathcal{N}} E\{d_{k,n} \mathbf{u}_{k,n}\} = \frac{1}{N} \sum_{k \in \mathcal{N}} \mathbf{p}_k$ . It can be readily shown following similar steps as in [155], that the solution minimizing (6.7) is given by  $\mathbf{w}_* = \mathbf{R}'^{-1} \mathbf{p}'$ . This argument indicates that a reasonable strategy in order to achieve dimensionality reduction is to construct the Krylov subspace relying on  $\mathbf{R}'$  and  $\mathbf{p}'$ ; *i.e.*, the average values relying on approximations of the previously mentioned quantities. To this end, at each node, the following approximations are computed:  $\hat{\mathbf{R}}'_n = \frac{1}{N} \sum_{k \in \mathcal{N}} \hat{\mathbf{R}}'_{k,n}$ , where  $\hat{\mathbf{R}}'_{k,n} = \zeta \hat{\mathbf{R}}_{k,n-1} + \mathbf{u}_{k,n-1} \mathbf{u}_{k,n-1}^T$  and  $\hat{\mathbf{p}}'_n = \frac{1}{N} \sum_{k \in \mathcal{N}} \hat{\mathbf{p}}'_{k,n}$ , with  $\hat{\mathbf{p}}_{k,n} = \zeta \hat{\mathbf{p}}_{k,n-1} + d_{k,n-1} \mathbf{u}_{k,n-1}$  and  $\zeta$  is the forgetting factor. From the previous relations, it can be observed that in order to construct the respective subspace,

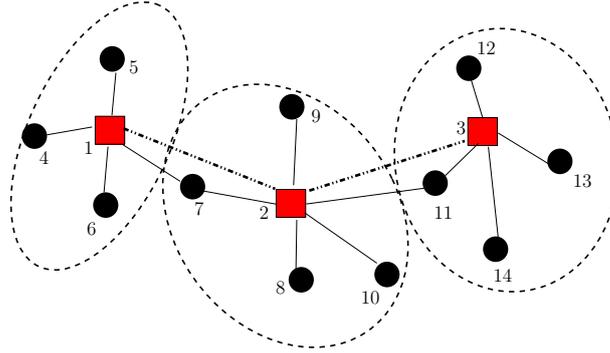


Figure 6.1: Illustration of a hierarchical network with  $L = 5$ . The solid lines denote the simple communication links, whereas the dashed-dotted ones the hierarchical communication links.

every node must have access to measurements coming out from the other nodes of the network, *i.e.*,  $\mathbf{u}_{k,n}, d_{k,n}$ ; however, this is, in general, infeasible in distributed networks. In the sequel, we present two techniques which will help us overstep this obstacle.

### 6.2.1 Enhancing the Information Flow

First of all, it should be stressed out that in system identification problems the input is defined as follows:  $\mathbf{u}_{k,n} = [u_{k,n} \ u_{k,n-1} \ \dots \ u_{k,n-m+1}]^T$ . Hence, the novel information at each time instant comprises of two numbers:  $u_{k,n}$  and  $d_{k,n}$ . In order to enhance the information flow, the following strategies are adopted.

1. We assume that  $\hat{\mathbf{R}}'_n$  and  $\hat{\mathbf{p}}'_n$  will not be updated every time instant but every  $L$  time instants instead. Thus the coefficients  $u_{k,n}$  and  $d_{k,n}$  will be delivered to the other nodes of the network within a time window of size  $L$ . This parameter is chosen with respect to the size of the network as well as the maximum distance between two nodes. As it will become clear in the simulations section, the larger the  $L$  the worse the performance of the algorithm; this behaviour is due to the fact that for a large time window,  $\hat{\mathbf{R}}'_n$  and  $\hat{\mathbf{p}}'_n$  are updated less frequently and their convergence to a good approximation is slowed down. Nevertheless, as it will become apparent in the simulations section, provided that  $L$  does not take too large values, the algorithm turns out to be relatively insensitive to its choice.
2. We adopt a multi-cluster architecture (see for example [59]) for the network in order to improve the flow of transmitted information. In principle, nodes which are connected

to a large number of neighbours are “equipped” with better transmission capabilities. Despite the fact that the issue of clustering the nodes according to predefined protocols has been extensively discussed in the literature, see [59] and references therein, complex protocols are beyond the scope of this paper. So, we adopt a simple hierarchical protocol, which has been employed in the context of adaptive distributed learning in [23]. More specifically, we classify the nodes, according to the number of their neighbors, into two subclasses: the hierarchical and the non-hierarchical ones. The former are able to communicate over three nodes, whereas the latter are not, and every non-hierarchical node is connected to a hierarchical one. The rationale is to assign enhanced transmission capabilities to the nodes which have many neighbours; through this procedure the information is delivered faster throughout the network, *e.g.*, [59].

Now, let us see how the information needed to construct the Krylov subspace is distributed over such a network, which is illustrated in Fig. 6.1. Notice that the network comprises of  $N = 14$  nodes and the number of the hierarchical nodes equals to 3. At each time instant, nodes have to transmit  $D$  coefficients to their neighbourhood; these are the updated components lying in the reduced space  $\mathbb{R}^D$ . At time instant 1, node 1 transmits to node 2,  $u_{1,1}, d_{1,1}$ , at time instant 2,  $u_{4,1}, d_{4,1}$ , at 3,  $u_{5,1}, d_{5,1}$  and at time instant 4,  $u_{6,1}, d_{6,1}$ . Node 2, at time instant  $n = 1$ , transmits to 3,  $u_{2,1}, d_{2,1}, u_{7,1}, d_{7,1}$ . At  $n = 2$ ,  $u_{1,1}, d_{1,1}, u_{8,1}, d_{8,1}$ , at  $n = 3$ ,  $u_{4,1}, d_{4,1}, u_{10,1}, d_{10,1}$ , at  $n = 4$ ,  $u_{5,1}, d_{5,1}, u_{9,1}, d_{9,1}$  and at  $n = 5$ ,  $u_{6,1}, d_{6,1}$ . The rest of the exchanges follow a similar philosophy. The largest number of coefficients is transmitted by node 2 and amounts to  $D + 4$ , where  $D$  comes from the  $D$  coefficients of the estimate and the other four from the information needed to construct the subspace. In the full rank scenario, every node has to transmit  $m$  coefficients to each neighboring node. Hence, if  $D$  is much smaller than  $m$ , which is the case of our interest, then the nodes transmit fewer coefficients, if they seek for a reduced rank solution.

Unfortunately, in networks with a large number of nodes and/or in scenarios where the longest path, among the nodes of the network, is large, the previously mentioned techniques may fail. Nevertheless, as it will become apparent in the simulations section, another route can be followed. Indeed, the Krylov subspace can be constructed by exploiting information coming from a single node, *e.g.*, a master node, without significant degradation of the per-

formance of the algorithm. It can be readily obtained that, if the information of a single node has to be delivered throughout the network, each node transmits  $D + 2$  coefficients at most. Hence in this scenario, the only limitation is to use a large enough  $L$ , which depends on the longest path among the nodes of the network, and then distribute the two coefficients, which are used to construct the subspace. A possible criterion in order to choose the master node is to find the node with the smallest eigenvalue spread, as (6.4) suggests. Techniques for finding this “optimum” node are beyond the scope of this paper and will be presented in a future work. In the simulations section, the case of choosing the “worst” node is also adopted in order to study the sensitivity of this scenario in failing to choose the “best” node.

Finally, if the statistics are the same for the nodes of the network, then the Krylov subspaces can be constructed locally, and then the information transmitted by each node drops to  $D$  coefficients, *i.e.*, the length of the reduced rank estimate.

### 6.2.2 The Algorithmic Scheme

As it has been already mentioned, our goal has now become to search for estimates that lie in the reduced  $D$ -dimensional Krylov subspace. However, in general, any vector in such a subspace is expressed in terms of  $m$  components, since it is a subset of  $\mathbb{R}^m$ . Our next goal becomes to map the respective estimates in the  $\mathbb{R}^D$  subspace; this mapping will result in the description of the estimates in terms of  $D$  components. Nevertheless, the mapping which leads vectors from the Krylov subspace to  $\mathbb{R}^D$ , is known. Moreover, the inverse mapping leading vectors from  $\mathbb{R}^D$  to the subspace is also known. This correspondence between vectors of the Krylov subspace with vectors lying in  $\mathbb{R}^D$  will be the kick-off point in order to reduce the communication load. More specifically, at each node, vectors which belong in  $\mathbb{R}^D$  will be computed and transmitted, reducing the communication load; these vectors can be readily mapped, locally at each node, back to the original Krylov subspace where they belong.

Let us define the  $m \times D$  matrix  $\hat{\mathbf{T}}_n$  the columns of which form a basis for  $K_n = K_D(\hat{\mathbf{R}}'_n, \hat{\mathbf{p}}'_n)$ . The following holds:  $\forall \tilde{\mathbf{w}} \in \mathbb{R}^D, \exists \mathbf{w} \in K_n : \mathbf{w} = \hat{\mathbf{T}}_n \tilde{\mathbf{w}}$  and  $\tilde{\mathbf{w}} = \hat{\mathbf{T}}_n^T \mathbf{w}$ <sup>1</sup>, [155]. According to the previous discussion, the matrix  $\hat{\mathbf{T}}_n^T$  maps vectors, of dimension  $m$  which belong in  $K_n$ , to the reduced dimension space, *i.e.*,  $\mathbb{R}^D$ , whereas  $\hat{\mathbf{T}}_n$  maps vectors lying in  $\mathbb{R}^D$  to  $K_n \subset \mathbb{R}^m$ .

---

<sup>1</sup>From now on, the tilded vectors will stand for vectors lying in  $\mathbb{R}^D$ .

The steps of the algorithm for each node  $k$  and at time instant  $n$ , can be summarized as follows:

- The estimates, of reduced dimension, from the neighbourhood, i.e.,  $\tilde{\mathbf{w}}_{l,n} \in \mathbb{R}^D$ ,  $\forall l \in \mathcal{N}_k$ , are received and convexly combined, with respect to the adopted combination strategy in order to produce  $\tilde{\boldsymbol{\phi}}_{k,n} := \sum_{l \in \mathcal{N}_k} a_{k,l} \tilde{\mathbf{w}}_{l,n}$ , where  $a_{k,l}$  are the combination weights. As already said, these estimates are related to their counterparts in the Krylov subspace in  $\mathbb{R}^m$  ones, according to:  $\forall n \in \mathbb{Z}_{\geq 0}$ ,  $\forall k \in \mathcal{N}$ ,  $\tilde{\mathbf{w}}_{k,n} = \hat{\mathbf{T}}_n^T \mathbf{w}_{k,n}$  (see also Appendix G).
- Taking into consideration the newly received information, i.e.,  $(d_{k,n}, \mathbf{u}_{k,n})$  the following hyperslab is defined in  $\mathbb{R}^D$ :  $\tilde{S}_{k,n} := \{\tilde{\mathbf{w}} \in \mathbb{R}^D : |d_{k,n} - \mathbf{u}_{k,n}^T \hat{\mathbf{T}}_n \tilde{\mathbf{w}}| \leq \epsilon_k\}$ , where  $\epsilon_k > 0$  can vary from node to node, depending on the noise statistics. The aggregate  $\tilde{\boldsymbol{\phi}}_{k,n}$ , which was computed at the previous step, is projected onto the  $q$  most recent hyperslabs, and then a convex combination of the resulting projections is computed. It has been verified, that by projecting onto a  $q > 1$  number of hyperslabs the convergence speed is accelerated [82, 150].
- The information needed in order to update the subspace is distributed over the network, using one of the techniques described previously. If  $\text{mod}(n, L) = 0$ , then  $\hat{\mathbf{R}}_{k,n}$ ,  $\hat{\mathbf{p}}_{k,n}$  are updated, and the matrix  $\hat{\mathbf{T}}_{n+1}$  is computed.

The previous can be encoded in the following formula.

**Algorithm 1:**

$$\tilde{\mathbf{w}}_{k,n+1} = \tilde{\boldsymbol{\phi}}_{k,n} + \tilde{\mu}_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{\tilde{S}_{k,j}}(\tilde{\boldsymbol{\phi}}_{k,n}) - \tilde{\boldsymbol{\phi}}_{k,n} \right), \quad (6.8)$$

where  $\mathcal{J}_n := \overline{\{0, n - q + 1\}}, n$ ,  $\sum_{j \in \mathcal{J}_n} \omega_{k,j} = 1$ ,  $\forall k \in \mathcal{N}$  and  $\tilde{\mu}_{k,n} \in (0, 2\tilde{\mathcal{M}}_{k,n})$  where [134]:

$$\tilde{\mathcal{M}}_{k,n} = \begin{cases} \frac{\sum_{j \in \mathcal{J}_n} \omega_{k,j} \|P_{\tilde{S}_{k,j}}(\tilde{\boldsymbol{\phi}}_{k,n}) - \tilde{\boldsymbol{\phi}}_{k,n}\|^2}{\|\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{\tilde{S}_{k,j}}(\tilde{\boldsymbol{\phi}}_{k,n}) - \tilde{\boldsymbol{\phi}}_{k,n}\|^2}, & \text{if } \|\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{\tilde{S}_{k,j}}(\tilde{\boldsymbol{\phi}}_{k,n}) - \tilde{\boldsymbol{\phi}}_{k,n}\| \neq 0, \\ 1, & \text{otherwise.} \end{cases} \quad (6.9)$$

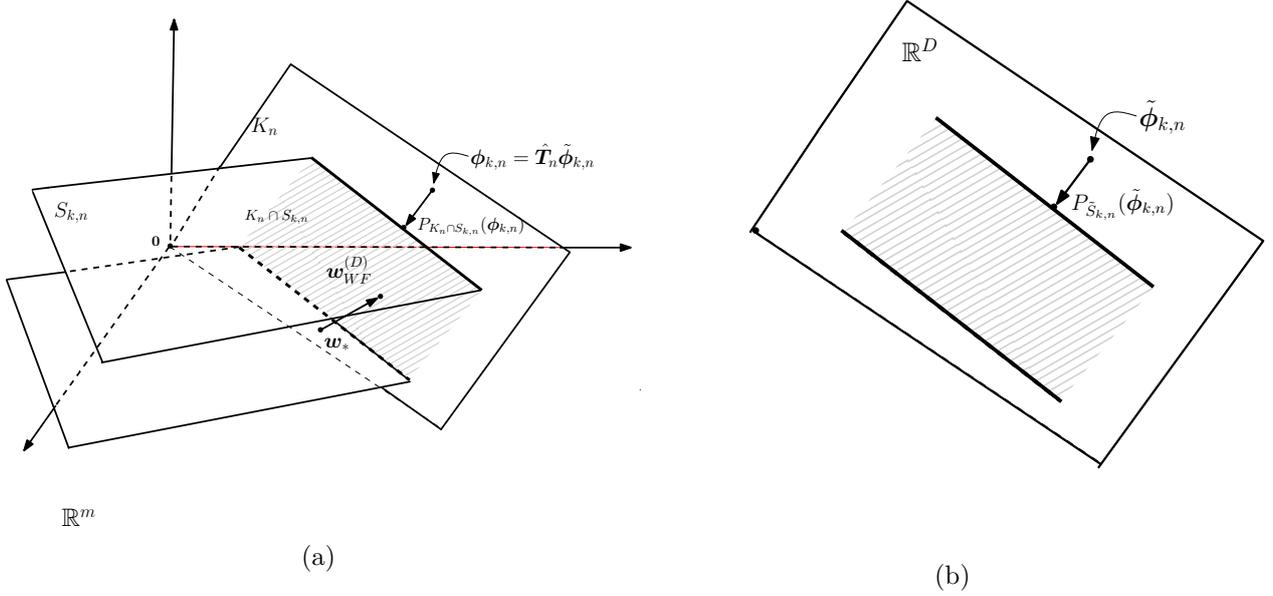


Figure 6.2: (a) Geometrical illustration of the algorithm for  $q = 1$ . The aggregate  $\phi_{k,n}$ , which belongs in the subspace, is projected onto the intersection of the subspace and the hyperslab, generated by the measurement data. (b) The algorithmic scheme in the reduced dimension space, i.e.,  $\mathbb{R}^D$ .

In the previously described scheme, the obtained estimates lie in  $\mathbb{R}^D$ , which implies that each sensor will transmit  $D$  coefficients at each time instant. The following claim clarifies the connection between the algorithm in (6.8) and the Krylov subspaces, discussed in the previous section.

**Claim 5.** *Eq. (6.8) is equivalent to*

$$\mathbf{w}_{k,n+1} = \hat{\mathbf{T}}_{n+1} \hat{\mathbf{T}}_n^T \left( \phi_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j} \cap K_n}(\phi_{k,n}) - \phi_{k,n} \right) \right), \quad (6.10)$$

where  $\mu_{k,n} \in (0, 2\mathcal{M}_{k,n})$ ,  $\phi_{k,n} = \hat{\mathbf{T}}_n \tilde{\phi}_{k,n}$ , that is, the corresponding aggregate in the respective Krylov space, and

$$\mathcal{M}_{k,n} = \begin{cases} \frac{\sum_{j \in \mathcal{J}_n} \omega_{k,j} \|P_{S_{k,j} \cap K_n}(\phi_{k,n}) - \phi_{k,n}\|^2}{\|\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j} \cap K_n}(\phi_{k,n}) - \phi_{k,n}\|^2}, & \text{if } \|\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S_{k,j} \cap K_n}(\phi_{k,n}) - \phi_{k,n}\| \neq 0 \\ 1, & \text{otherwise.} \end{cases}$$

*Proof.* The proof is given in Appendix H. □

The geometrical interpretation of the algorithm is given in Fig. 6.2. The complexity

of the algorithm is of order  $O(qD)$  coming from (6.8),  $O(\frac{Nm}{L})$  from the update of  $\hat{\mathbf{R}}'_n$ , and  $O(\frac{Dm^2}{L})$  due to the computation of  $\hat{\mathbf{T}}_n$ , *e.g.*, [155]. It is important to notice that the dominant complexity-contributing terms, which are involved in the subspace computation, depend also on the frequency with which the subspace is constructed. Hence, if one is to reduce the computational load, a larger  $L$  must be chosen. Obviously, this results to a performance degradation; however as it will become clear in the simulations section, the algorithms turn out to be relatively insensitive to this parameter.

As it will become clear shortly, the algorithm enjoys a number of nice convergence properties. Despite the fact that at each node the recursion given in (6.8) is employed, the theoretical properties for (6.10) will be studied since the estimates computed by this scheme belong to the same subspace with  $\mathbf{w}_{\text{WF}}^{(D)}$  and from the fact that the two schemes are equivalent. For the algorithm in (6.10), we prove a number of nice convergence properties such as: monotonicity, asymptotic optimality and strong convergence to a point which lies in the consensus subspace. Moreover, we prove that the estimates at each node converge to a vector which belongs to the Krylov subspace. It is important to notice that asymptotic optimality implies that the distance of the computed estimates from the intersection of the hyperslabs with the Krylov subspace will tend asymptotically to zero. Moreover, recalling the discussion in subsection 6.1.2, these sets contain  $\mathbf{w}_{\text{WF}}^{(D)}$  with a high probability.

**Assumptions 1:**

- (a) There exists a non-negative integer, say  $n_0$ , for which  $\Omega = \bigcap_{n \geq n_0} \Omega_n \neq \emptyset$  where  $\Omega_n = K_n \cap \Omega'_n$  with  $\Omega'_n := \bigcap_{k \in \mathcal{N}} \bigcap_{j \in \mathcal{J}_n} S_{k,j}$ . In words, the hyperslabs together with the Krylov subspaces share a non-empty intersection.
- (b) There exists  $n_1$  such that  $\hat{\mathbf{T}}_n = \hat{\mathbf{T}}_{n_1}$ ,  $\forall n \geq n_1$ . In other words, after a finite number of iterations, the subspace remains fixed<sup>2</sup>.
- (c) Let some sufficient small  $\varepsilon_1 > 0$  such that  $\mu_{k,n} \in (\varepsilon_1 \mathcal{M}_{k,n}, \mathcal{M}_{k,n}(2 - \varepsilon_1))$ ,  $k \in \mathcal{N}$ .
- (d) Let us define  $\mathfrak{C} := \tilde{\Omega} \cap \tilde{\mathcal{O}}$ , where the cartesian product space  $\tilde{\Omega} := \underbrace{\tilde{\Omega} \times \dots \times \tilde{\Omega}}_N$ ,  $\tilde{\Omega} := \bigcap_{n \geq n_0} \bigcap_{k \in \mathcal{N}} \bigcap_{j \in \mathcal{J}_n} \tilde{S}_{k,j}$  and  $\tilde{\mathcal{O}} := \{\tilde{\mathbf{w}} \in \mathbb{R}^{ND} : \tilde{\mathbf{w}} = [\tilde{\mathbf{w}}^T, \dots, \tilde{\mathbf{w}}^T]^T, \tilde{\mathbf{w}} \in \mathbb{R}^D\}$ . We

---

<sup>2</sup>For a large choice of  $n_1$  the approximations of the quantities used in order to construct the subspace are good and, consequently, this assumption does not lead to performance degradation.

assume that  $\text{ri}_{\tilde{\mathcal{O}}}\tilde{\Omega} \neq \emptyset$ , where this term stands for the relative interior of  $\mathfrak{C}$  with respect to  $\tilde{\mathcal{O}}$  (see Chapter 3).

**Theorem 8.** *Under the previously adopted assumptions, the following properties can be proved.*

- **Monotonicity.** *Under assumptions (a), (b), (c) for the recursion given in (6.10) it holds that*

$$\|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}_*\| \leq \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}_*\|, \quad \forall n \geq n'_0$$

$$\text{where } n'_0 = \max\{n_0, n_1\}, \quad \underline{\mathbf{w}}_* = \begin{bmatrix} \hat{\mathbf{w}}_* \\ \vdots \\ \hat{\mathbf{w}}_* \end{bmatrix} \in \mathbb{R}^{Nm}, \quad \forall \hat{\mathbf{w}}_* \in \Omega \quad \text{and} \quad \underline{\mathbf{w}}_n = \begin{bmatrix} \mathbf{w}_{1,n} \\ \vdots \\ \mathbf{w}_{N,n} \end{bmatrix}.$$

- **Asymptotic Optimality.** *If assumptions (a), (b), (c) hold true, we have that*

$$\lim_{n \rightarrow \infty} d(\mathbf{w}_{k,n+1}, \Omega_n) = 0, \quad \forall k \in \mathcal{N},$$

where  $d_{(\mathfrak{G})}(\cdot, \Omega_n)$  denotes the distance of a vector from  $\Omega_n$ . In other words, the distance of the estimates from the intersection set  $\Omega_n$ , tends asymptotically to zero.

- **Asymptotic Consensus.** *Consider that assumptions (a), (b), (c), hold. Then  $\lim_{n \rightarrow \infty} \|\mathbf{w}_{k,n} - \mathbf{w}_{l,n}\| = 0$ ,  $\forall k, l \in \mathcal{N}$ .*
- **Strong Convergence.** *Under assumptions (a), (b), (c), (d), it holds that  $\lim_{n \rightarrow \infty} \underline{\mathbf{w}}_n = \underline{\mathbf{w}}_{\mathcal{O}}, \underline{\mathbf{w}}_{\mathcal{O}} \in \mathcal{O}$ , where  $\mathcal{O}$  stands for the consensus subspace. Moreover, if we define  $\underline{\mathbf{w}}_{\mathcal{O}} := [\mathbf{w}_{\mathcal{O}}^T, \dots, \mathbf{w}_{\mathcal{O}}^T]^T$ , it holds that  $\underline{\mathbf{w}}_{\mathcal{O}} \in K_{n_1}$ . The previous relation yields that the estimates for the whole network converge to a point that lies in the consensus subspace and the estimate at each node converges to a point which lies in the estimated Krylov subspace.*

*Proof.* The proof is provided in Appendix I. □

### 6.3 Whitening the input

Recall the discussion in Section 6.1.1 regarding (6.4). As it was documented there, the performance of the Krylov based reduced rank algorithm is dictated, mainly, by the input

statistics. In other words, in cases where the input is highly correlated and, henceforth, the eigenvalue spread of the autocorrelation matrix takes a large value, then the upper bound of the distance between the unknown vector and the one, which is tracked inside the Krylov subspace, is large and as it has been experimentally verified, the performance of the algorithms built around the Krylov subspaces is degraded. This results to an increased error floor in the steady state, as we will see in the Numerical Examples section. Hence, a reasonable strategy, which will be adopted here, is to employ a transformation that “whitens” the input. To this end, at each time instant the input vectors are multiplied with a properly chosen matrix, such that the autocorrelation matrix of the “new” input to be as close as possible to the identity matrix. A first approach could be to employ the celebrated Karhunen Loeve transform in order to produce a transformed input for which the eigenvalue spread of the autocorrelation matrix would be equal to 1. Nevertheless, as it has been also documented in [120], this approach requires a-priori knowledge of the input statistics, which is in general infeasible. Hence, an alternative route has to be followed. In the non-distributed scenario, the following transformation has been proposed [120]:  $\boldsymbol{\psi}_n = \mathbf{Z}^{\frac{1}{2}} \mathbf{Y} \mathbf{u}_n \in \mathbb{R}^m$ , where  $\mathbf{Y}$  is the  $m \times m$  Discrete Cosine Transformation (DCT) transformation matrix<sup>3</sup>, and  $\mathbf{Z} = \text{diag}\{\frac{1}{\hat{\sigma}_1^2} \dots \frac{1}{\hat{\sigma}_m^2}\}$ , where  $\hat{\sigma}_i$ ,  $i = 1, \dots, m$  is the  $i$ -th element in the diagonal of the matrix  $E\{\mathbf{Y} \mathbf{u}_n \mathbf{u}_n^T \mathbf{Y}^T\}$ . The physical reasoning of this transformation can be summarized as follows<sup>4</sup>. The left and right multiplication with the DCT matrix, approximately diagonalizes the matrix  $\mathbf{R}$  (see also [120]) so as to produce

$$E\{\mathbf{Y} \mathbf{u}_n \mathbf{u}_n^T \mathbf{Y}^T\} \approx \begin{bmatrix} \hat{\sigma}_1^2 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \hat{\sigma}_m^2 \end{bmatrix}. \quad (6.11)$$

Now, it is not difficult to see that the multiplication with the matrix  $\mathbf{Z}^{\frac{1}{2}}$ , normalizes the diagonal entries of the matrix in (6.11) so that the resulting autocorrelation matrix approximates the identity matrix. In practice, since the coefficients  $\hat{\sigma}_i$ ,  $i = 1, \dots, m$  are unknown, one relies on the following recursive approximation of them:  $\hat{\sigma}_{i,n}^2 = \gamma \hat{\sigma}_{i,n-1}^2 + [\mathbf{Y} \mathbf{u}_n]_i^2$ , where

<sup>3</sup>For the DCT transformation matrix holds that  $\mathbf{Y} \mathbf{Y}^T = \mathbf{Y}^T \mathbf{Y} = \mathbf{I}_m$ . It should be pointed out that a variety of transformations could be employed, e.g., the Fourier Transformation. However, the DCT one is usually adopted [120].

<sup>4</sup>For a more detailed analysis the reader is referenced to [120].

## Dimensionality Reduction in Distributed Adaptive Learning via Krylov Subspaces

---

$\gamma \in (0, 1]$ , and with  $[\cdot]_i$  we denote the  $i$ -th component of a vector, e.g., [120, 155]. Obviously the performance of the previously mentioned transformation, *i.e.*, how “close” will be the final matrix to the identity one, depends on  $\mathbf{R}$ . However, in practice it has been observed that the previously mentioned transformation results in autocorrelation matrices, which are reasonably close to diagonal.

In the distributed scenario, our goal is to impose a transformation, which is common to all the nodes of the network, and which whitens the autocorrelation matrix used for the construction of the subspace, *i.e.*,  $\mathbf{R}'$ . Assuming that the input vectors between any two different nodes of the network are independent and have zero mean, which is usually the case, *e.g.*, [28, 98], we have that  $\mathbf{R}' = \frac{1}{N}E[\mathbf{u}'_n \mathbf{u}'_n{}^T]$ , where  $\mathbf{u}'_n = \sum_{k \in \mathcal{N}} \mathbf{u}_{k,n}$ . The transformed input takes the form ([120]):  $\boldsymbol{\psi}_{k,n} = \mathbf{Z}'^{\frac{1}{2}} \mathbf{Y} \mathbf{u}_{k,n}$  where  $\mathbf{Z}' = \text{diag}\{\frac{1}{\sigma_1^2} \dots \frac{1}{\sigma_m^2}\}$ , and  $\sigma'_i$ ,  $i = 1, \dots, m$  is the  $i$ -th element of the diagonal of the matrix  $\mathbf{Y} \mathbf{R}' \mathbf{Y}^T$ . Employing the transformed input in the linear model, we get that  $d_{k,n} = \mathbf{u}_{k,n}^T \mathbf{w}_* + v_{k,n} = \mathbf{h}_{k,n}^T \mathbf{h}_* + v_{k,n}$ , where

$$\mathbf{h}_* = \mathbf{Z}'^{-\frac{1}{2}} \mathbf{Y} \mathbf{w}_* \Leftrightarrow \mathbf{w}_* = \mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{h}_*. \quad (6.12)$$

It should be pointed out that, by employing a transformed input, the generated estimates do not track the original unknown vector, but the transformed one, *i.e.*,  $\mathbf{h}_*$ . Nevertheless, by multiplying them with the inverse transformation, which is in our case  $\mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}}$ , one obtains estimates tracking the original unknown vector (see also [52]).

It is obvious that the definition of the corresponding Krylov subspace changes, since the input changes. Let us define

$$\mathcal{R}' = \frac{1}{N} \sum_{k \in \mathcal{N}} \mathbb{E}\{\mathbf{h}_{k,n} \mathbf{h}_{k,n}^T\} = \frac{1}{N} \sum_{k \in \mathcal{N}} \mathbf{Z}'^{\frac{1}{2}} \mathbf{Y} E\{\mathbf{u}_{k,n} \mathbf{u}_{k,n}^T\} \mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} = \mathbf{Z}'^{\frac{1}{2}} \mathbf{Y} \mathbf{R}' \mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}}, \quad (6.13)$$

and  $\mathbf{r}' = \frac{1}{N} \sum_{k \in \mathcal{N}} E\{d_{k,n} \mathbf{h}_{k,n}\}$ . Using a similar rationale as in section 6.1.2, the algorithm, after employing the transformed input, tracks the following vector

$$\hat{\mathbf{h}} = \mathbf{T}' (\mathbf{T}'^T \mathcal{R}' \mathbf{T}')^{-1} \mathbf{T}'^T \mathcal{R}' \mathbf{h}_* = P_{K_D(\mathcal{R}', \mathbf{r}')}^{(\mathcal{R}')}(\mathbf{h}_*), \quad (6.14)$$

where  $\mathbf{T}'$  is an  $m \times D$  matrix whose column form an orthonormal basis of  $K_D(\mathcal{R}', \mathbf{r}')$ . Now, let us shed some light on the connection between the estimates generated exploiting the

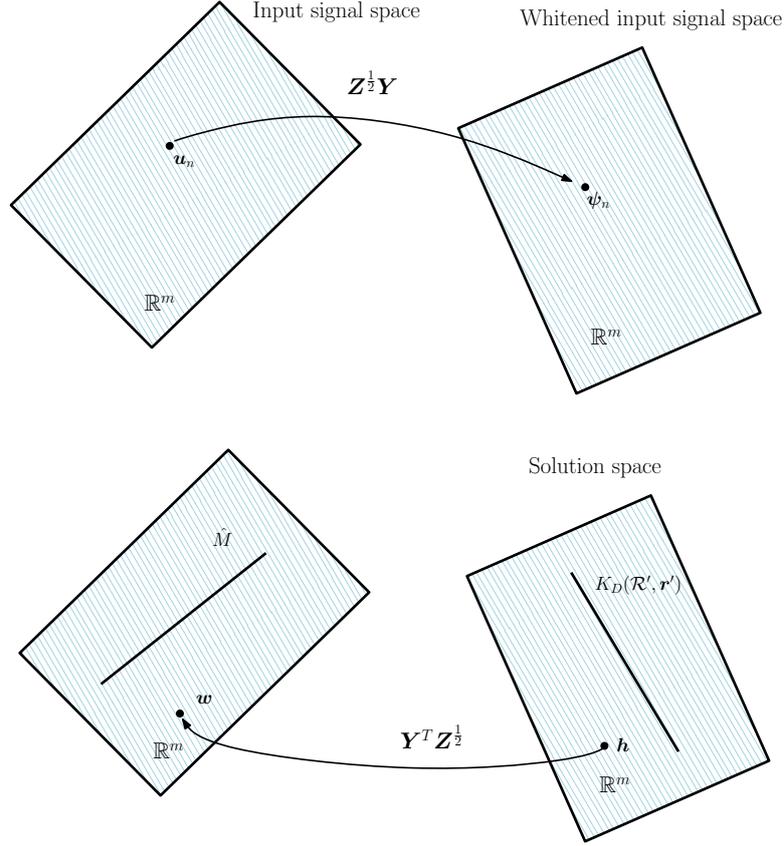


Figure 6.3: Illustration of  $K_D(\mathcal{R}', \mathbf{r}')$ ,  $\hat{M}$  and the connection between points that belong to them.

transformed input, and the estimates, which are produced relying on the original input. If we substitute (6.12) and (6.13) into (6.14) we obtain

$$\begin{aligned} \hat{\mathbf{h}} &= \mathbf{T}' (\mathbf{T}'^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{Y} \mathbf{R}' \mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{T}')^{-1} \mathbf{T}'^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{Y} \mathbf{R}' \mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{h}_* \\ &= \mathbf{T}' (\mathbf{T}'^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{Y} \mathbf{R}' \mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{T}')^{-1} \mathbf{T}'^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{Y} \mathbf{R}' \mathbf{w}_*. \end{aligned} \quad (6.15)$$

Notice that  $\mathbf{S}_{\hat{M}} := \mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{T}'$  is an  $m \times D$  matrix, of rank  $D$  ([68]), hence its columns form a basis for a new subspace  $\hat{M} := \text{range}\{\mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} \mathbf{T}'\}$ . Fig. 6.3 illustrates the connection between the points of the two subspaces. Now, according to the previous discussion, if we left multiply (6.15) by  $\mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}}$ , in order to employ the inverse transformation, we get

$$\mathbf{Y}^T \mathbf{Z}'^{\frac{1}{2}} \hat{\mathbf{h}} = \mathbf{S}_{\hat{M}} (\mathbf{S}_{\hat{M}}^T \mathbf{R}' \mathbf{S}_{\hat{M}})^{-1} \mathbf{S}_{\hat{M}}^T \mathbf{R}' \mathbf{w}_* = \hat{\mathbf{w}} = P_{\hat{M}}^{(\mathbf{R}')}(\mathbf{w}_*). \quad (6.16)$$

## Dimensionality Reduction in Distributed Adaptive Learning via Krylov Subspaces

---

From (6.16) we conclude that

$$\mathbf{Y}^T \mathbf{Z}^{\frac{1}{2}} \hat{\mathbf{h}} = \hat{\mathbf{w}} \Leftrightarrow \hat{\mathbf{h}} = \mathbf{Z}'^{-\frac{1}{2}} \mathbf{Y} \hat{\mathbf{w}} \quad (6.17)$$

Equation (6.17) establishes the connection among estimates occurring in the case where the input is  $\mathbf{h}_{k,n}, \forall k \in \mathcal{N}, \forall n \in \mathbb{N}$ , with the ones produced by the input  $\mathbf{u}_{k,n}, \forall k \in \mathcal{N}, \forall n \in \mathbb{N}$ .

It should be pointed out that, if we employ the whitening transformation, the estimates obtained from the original input (which are produced by employing the inverse transformation), lie in  $\hat{M}$ , which is also a subspace of dimension equal to  $D$ , instead of  $K_D(\mathbf{R}', \mathbf{p}')$ , which would be the case if the original input  $\mathbf{u}_{k,n}$  were employed. Despite the fact that the reduced rank Wiener Filter  $\mathbf{w}_{WF}^{(D)}$  does not belong to  $\hat{M}$ , in general, as it will become apparent in the Numerical Examples section, if the input is highly correlated it is better to seek for  $\hat{\mathbf{w}}$  instead of  $\mathbf{w}_{WF}^{(D)}$ , since the misadjustment between  $\mathbf{w}_{WF}^{(D)}$  and  $\mathbf{w}_*$  is large.

Obviously, in order to construct the matrix  $\mathbf{Z}'$ , knowledge on the statistic has to be available. As in the previous section, we rely on estimates of the unknown statistics in order to construct  $\mathbf{Z}'$ . More specifically, the approximated matrix is given by  $\hat{\mathbf{Z}}'_n = \text{diag}\{\frac{1}{\hat{\sigma}_{1,n}^2} \dots \frac{1}{\hat{\sigma}_{m,n}^2}\}$ , where  $\hat{\sigma}_{i,n}^2 = \gamma \hat{\sigma}_{i,n-1}^2 + \frac{1}{N} \mathbf{Y}[\mathbf{u}_n]_i^2$ ,  $\gamma \in (0, 1]$ .

The algorithm is similar to the one developed in the previous section and its mathematical formula is given by

**Algorithm 2:**

$$\mathbf{h}_{k,n+1} = \hat{\mathbf{T}}'_{n+1} \hat{\mathbf{T}}_n'^T \left( \boldsymbol{\varphi}_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{k,j} \cap \hat{K}_n}(\boldsymbol{\varphi}_{k,n}) - \boldsymbol{\varphi}_{k,n} \right) \right), \quad (6.18)$$

with  $\boldsymbol{\varphi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{h}_{l,n}$  and  $S'_{k,n} := \{\mathbf{w} \in \mathbb{R}^m : |d_{k,n} - \mathbf{w}^T \mathbf{h}_{k,n}| \leq \epsilon_k\}$ . Furthermore, the  $m \times D$  matrix  $\hat{\mathbf{T}}'_n$  is defined similarly to  $\hat{\mathbf{T}}_n$ , and its columns form an orthonormal basis of  $\hat{K}_n := K_D(\hat{\mathcal{R}}'_n, \hat{\mathbf{r}}'_n)$ , where  $\hat{\mathcal{R}}'_n, \hat{\mathbf{r}}'_n$ , are approximations of the  $\mathcal{R}'$  and  $\mathbf{r}'$  respectively, and they are computed recursively in a similar way as in the previous section.

Recall the assumptions of Theorem 1. In order to derive the convergence analysis of the algorithm in (6.18), we consider that the assumptions of Theorem 1 hold true, with the following slight modifications:

- The intersection  $\Omega$  becomes  $\hat{\Omega} = \bigcap_{n \geq n_0} \hat{\Omega}_n \neq \emptyset$ , where  $\hat{\Omega}_n = \hat{K}_n \cap \hat{\Omega}'_n$ , and  $\hat{\Omega}'_n :=$

$\bigcap_{k \in \mathcal{N}} \bigcap_{j \in \mathcal{J}_n} S'_{k,j}$  (Assumption (a')).

- There exists  $n_1$  such that  $\hat{\mathbf{T}}'_n = \hat{\mathbf{T}}'_{n_1}$ ,  $\forall n \geq n_1$  (Assumption (b')).
- After a finite number of iterations, say  $n_2$ ,  $\mathbf{Z}'_n = \mathbf{Z}'_{n_2}$ ,  $\forall n \geq n_2$  and, for compact notations, we define  $n'_0 = \max\{n_0, n_1, n_2\}$  (Assumption (c')).
- The upper bound of the step size equals to  $2\mathcal{M}'_{k,n}$ , where

$$\mathcal{M}'_{k,n} = \begin{cases} \frac{\sum_{j \in \mathcal{J}_n} \omega_{k,j} \|P_{S'_{k,j} \cap \hat{K}_n}(\varphi_{k,n}) - \varphi_{k,n}\|^2}{\|\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{k,j} \cap \hat{K}_n}(\varphi_{k,n}) - \varphi_{k,n}\|^2}, & \text{if } \|\sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{k,j} \cap \hat{K}_n}(\varphi_{k,n}) - \varphi_{k,n}\| \neq 0 \\ 1, & \text{otherwise,} \end{cases}$$

(Assumption (d')).

- The set  $\mathfrak{C}$ , now becomes  $\mathfrak{C}'$ , with  $\mathfrak{C}' := \hat{\Omega} \cap \tilde{\mathcal{O}}$ , where  $\hat{\Omega} := \underbrace{\hat{\Omega} \times \dots \times \hat{\Omega}}_N$  and  $\hat{\Omega} := \bigcap_{n \geq n_0} \bigcap_{k \in \mathcal{N}} \bigcap_{j \in \mathcal{J}_n} \tilde{S}'_{k,j}$  employing the modified input (Assumption (e')).

### Theorem 9.

- **Monotonicity:** Assume that assumptions (a'), (b'), (c'), (d'), hold true. It holds that

$$\|\underline{\mathbf{w}}_{n+1} - \underline{\hat{\mathbf{w}}}'_*\|_{\mathbf{G}} \leq \|\underline{\mathbf{w}}_n - \underline{\hat{\mathbf{w}}}'_*\|_{\mathbf{G}}, \quad \forall n \geq n'_0,$$

where  $\mathbf{G} = \text{diag}\{\underbrace{\mathbf{A}, \dots, \mathbf{A}}_N\}$ ,  $(Nm \times Nm)$ ,  $\mathbf{A} = \mathbf{Y}^T \mathbf{Z}_{n_2}^{-1} \mathbf{Y}$ ,  $(m \times m)$ ,  $\underline{\hat{\mathbf{w}}}'_* = [\hat{\mathbf{w}}_*'^T, \dots, \hat{\mathbf{w}}_*'^T]^T$ ,

where  $\hat{\mathbf{w}}'_* \in \bar{\Omega}$ ,  $\bar{\Omega} = \bigcap_{n \geq n'_0} \bar{\Omega}_n$ ,  $\bar{\Omega}_n = \bar{M} \cap \Omega'_n$  and  $\bar{M} := \text{range}\{\mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}} \mathbf{T}'_{n_1}\}$  is an approximation of  $\hat{M}$ . The last equation states that the algorithm enjoys monotonicity, in the  $\mathbf{G}$  norm sense.

- **Asymptotic Optimality:** Under Assumptions (a'), (b'), (c'), (d'), it holds that

$$\lim_{n \rightarrow \infty} d(\mathbf{w}_{k,n+1}, \bar{\Omega}_n) = 0, \quad \forall k \in \mathcal{N}.$$

- **Strong Convergence to a point that lies in the Consensus subspace:** Consider that (a'), (b'), (c'), (d'), (e'), hold true it holds that  $\lim_{n \rightarrow \infty} \underline{\mathbf{w}}_n = \underline{\mathbf{w}}'_O$ ,  $\underline{\mathbf{w}}'_O \in \mathcal{O}$ . As in Theorem 1, if we define  $\underline{\mathbf{w}}'_O := [\mathbf{w}'_O{}^T, \dots, \mathbf{w}'_O{}^T]^T$ , it holds that  $\underline{\mathbf{w}}'_O \in \bar{M}$ . In other

words, as in Theorem 1, the estimates for the whole network converge to a point that lies in the consensus subspace and the estimate at each node converges to a point which lies in  $\overline{M}$ .

*Proof.* The proof is given in Appendix I. □

## 6.4 Numerical Examples

In this section, the performance of the proposed algorithms is validated within the system identification framework. In order to evaluate the performance of the proposed algorithms, we compare it with a modified version of the proposed scheme, denoted as subsampled Adaptive Projected Subgradient Method (sAPSM), where each node, instead of transmitting the whole estimate vector, at every time instant, transmits a subset of  $D$  coefficients of it. More specifically, at time instant 1, the first  $D$  coefficients are transmitted, at time instant 2, the coefficients  $\#D + 1, \dots, \#2D$  and so on. Moreover, the proposed algorithms are compared with the full rank APSM, *i.e.*, the proposed where the full vector estimate is transmitted and with the diffusion based Adapt-Combine LMS (A-C LMS) [28].

In the first experiment, we consider an ad-hoc network, in which the number of nodes equals to  $N = 20$ . The unknown vector is of dimension  $m = 160$ . We consider that the input samples,  $\mathbf{u}_n = [u_n, \dots, u_{n-m+1}]^T$ , obey the following model  $u_{k,n} = \theta_k u_{k,n-1} + \sqrt{1 - \theta_k^2} \chi_{k,n}$ , where  $\theta_k$  is a parameter, which we will alter throughout the experiments, so as to validate the proposed schemes in weakly or strongly correlated environments, and  $\chi_{k,n}$  is drawn from the Gaussian distribution with unit variance. The variance of the noise, at each node, equals to  $\sigma_k = 0.01 \times \xi_k$ , where  $\xi_k \in (0.5, 1]$ , under the uniform distribution. Furthermore, the combination coefficients are chosen with respect to the Metropolis rule. Finally, the adopted performance metric, which will be used, is the average Mean Square Error (MSE), given by  $\text{MSE}(n) = 1/N \sum_{k \in \mathcal{N}} (d_{k,n} - \mathbf{u}_{k,n}^T \mathbf{w}_{k,n})^2$ , and the curves are the result of averaging 100 realizations for smoothing purposes.

The number of hyperslabs used per time update equals to  $q = 4$ , the step-size is chosen  $\mu_{k,n} = 1/2 \times \mathcal{M}_{k,n}$  and the width of the hyperslabs equals to  $\epsilon_k = 1.3 \times \sigma_k$ . The weights are set  $\omega_{k,n} = 1/q$ . The step-size in the A-C LMS equals to  $3 \times 10^{-3}$ , so that the algorithm converges to a similar error floor with the full rank APSM. In the first experiment, we

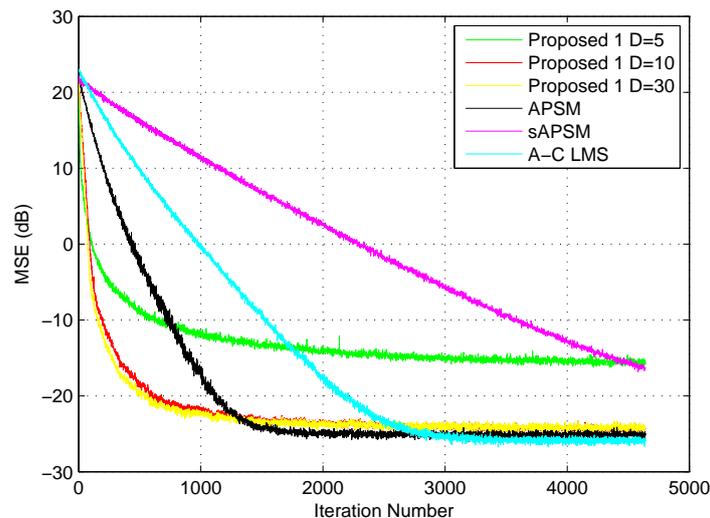


Figure 6.4: Average MSE for the first experiment.

study the performance of the proposed scheme (denoted as Proposed 1), with respect to the dimension of the subspace, within which we seek for a solution. For this reason, we consider a weakly correlated environment, so the parameter  $\theta_k \in (0, 0.5)$ ,  $\forall k \in \mathcal{N}$ , with respect to the Uniform distribution. Moreover, since the unknown vector does not undergo changes, the forgetting factor is chosen  $\zeta = 1$ . Finally, we assume that  $L = 1$ , i.e., the subspace is updated at each time instant and for the sAPSM  $D = 30$ . From Fig. 6.4 it can be seen that even if the dimension of the subspace takes small values, compared to  $m$ , the Proposed 1 performs significantly well. Analytically, the Proposed 1, converges fast and for the specific choices  $D = 10$ ,  $D = 30$  the steady state error floor is only slightly increased compared to the full rank APSM and the A-C LMS. If  $D = 5$ , then the steady state error floor increases significantly. Moreover, the Krylov-based algorithms outperform the sAPSM. Finally, it should be pointed out that the complexity of the LMS is of order  $O(m)$  and the complexity of the APSM is of order  $O(qm)$ .

In Table 6.1 we present the steady state Mean Square Deviation, *i.e.*,  $\|\mathbf{w}_{\text{av}} - \mathbf{w}^*\|^2$ , as well as the distance of the steady state estimate from  $\mathbf{w}_{\text{WF}}^{(D)}$ , *i.e.*,  $\|\mathbf{w}_{\text{av}} - \mathbf{w}_{\text{WF}}^{(D)}\|^2$ , where  $\mathbf{w}_{\text{av}} = 1/N \sum_{k \in \mathcal{N}} \mathbf{w}_{k,n}$ , for a large  $n$ . It can be observed, that the smaller the dimension of the Krylov subspace, the smaller the distance of the estimate from  $\mathbf{w}_{\text{WF}}^{(D)}$ , whereas the mean square deviation is larger. This is a direct consequence of (6.4) since, as one can see in this equation, a smaller  $D$  leads to a larger upper bound of the distance between  $\mathbf{w}_{\text{WF}}^{(D)}$  and

## Dimensionality Reduction in Distributed Adaptive Learning via Krylov Subspaces

---

$D$	$\ \mathbf{w}_{\text{av}} - \mathbf{w}_{\text{WF}}^{(D)}\ ^2$	$\ \mathbf{w}_{\text{av}} - \mathbf{w}^*\ ^2$
5	$1.21 * 10^{-4}$	$3.49 * 10^{-4}$
10	$1.87 * 10^{-4}$	$1.26 * 10^{-5}$
20	$2.28 * 10^{-4}$	$1.23 * 10^{-5}$
30	$2.40 * 10^{-4}$	$1.22 * 10^{-5}$

Table 6.1: Steady State distances.

$D$	SDCS
5	$2.6 * 10^{-4}$
10	$2.1 * 10^{-4}$
20	$1.9 * 10^{-4}$
30	$1.9 * 10^{-4}$

Table 6.2: Squared Distance from the Consensus Subspace.

$\mathbf{w}_*$ . Finally, in Table 6.2, we present the steady state squared distance from the consensus subspace (SDCS), *i.e.*,  $\|(\mathbf{I}_m - \mathbf{B}\mathbf{B}^T)\underline{\mathbf{w}}_n\|^2$ ,  $n \rightarrow \infty$ , where the definition of the matrix  $\mathbf{B}$  is provided in Chapter 4. It can be readily seen, that in the steady state every choice of  $D$  leads to a small distance from the consensus subspace.

In the second experiment, Fig. 6.5, the parameters remain the same as in the previous one. Nevertheless, here we examine the Average Excess Mean Square Error (EMSE) instead of the MSE. The Average EMSE is given by  $\text{EMSE} := 1/N \sum_{k \in \mathcal{N}} (\mathbf{u}_{k,n}^T \mathbf{w}_* - \mathbf{u}_{k,n}^T \mathbf{w}_{k,n})^2$ . From Fig. 6.5 it can be seen that the full rank LMS and the APSM converge to a lower steady state error floor, compared to the algorithms built around the Krylov subspace rationale. This fact is expected since in the Krylov based algorithms we seek for a vector lying in a subspace of lower dimension, and not the unknown one. However, the Krylov based algorithms converge significantly faster and, moreover, compared to the full rank algorithms, the difference in the steady state error is relatively small.

In the third experiment, we consider that the parameters remain the same as in the previous experiment, albeit a fixed dimension for the subspace, namely  $D = 10$ , is chosen. Our goal is to study the sensitivity of the algorithm, to the parameter  $L$ . To this end, we set different values to  $L$ , or in other words, to the frequency with which the subspace is updated. From Fig. 6.6 it can be readily observed that the smaller the update window, the faster the convergence, due to the fact that for a small window we update the estimate of the subspace more often, and we reach sooner a good approximation of it, compared to the case

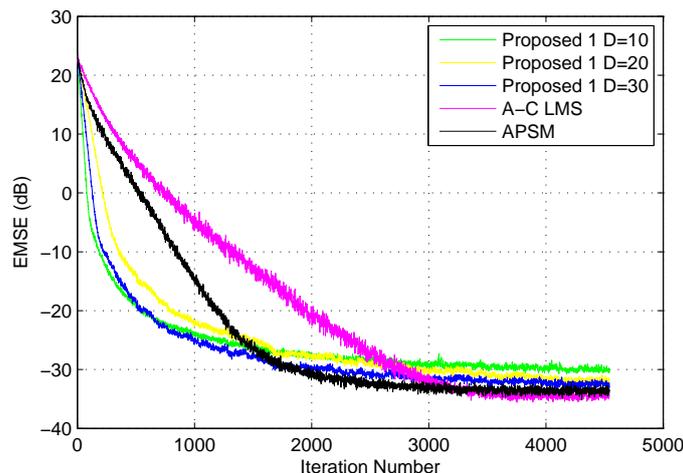


Figure 6.5: Average EMSE for the second experiment.

of a larger window. Moreover, as in the previous experiment, the Krylov-based algorithms outperform the sAPSM. Finally, we should note that the Proposed 1 performs well even for large values of  $L$ , which makes it appropriate to be adopted in distributed learning.

In the fourth experiment, we consider a non-stationary environment, since, as it is by now well established, a fast convergence speed does not necessarily coincide with a good tracking ability [64]. To be more specific, we consider that a sudden change in the unknown parameter vector takes place. So, in this experiment, we fix  $L = 1$  and  $D = 10$  and we alter the forgetting factor. From Fig. 6.7 it can be seen that until the system undergoes the change, the best performance is achieved for  $\zeta = 1$ , whereas for smaller  $\zeta$  the steady state error floor is increased. Nevertheless, if  $\zeta = 1$ , the algorithm has a long memory of the old statistics, through which the subspace is constructed, that have to change and its tracking ability is not good. On the contrary, the other choices of  $\zeta$  provide a good tracking ability. Obviously, for large  $L$  the tracking ability may be affected, since apart from the forgetting factor, one has to take into consideration the fact that at time instant  $n$  the quantities sensed at a past time instant are delivered through the network; this is a direct consequence of the strategy adopted in subsection 6.2.1, in order to enhance the information flow. In this case, we consider that the algorithm is able to monitor abrupt changes of the orbit  $(\mathbf{w}_{k,n})_{n \in \mathbb{Z}_{\geq 0}}$ , in order to restart transmitting the input and the desire response. In order to "sense" the previously mentioned abrupt changes, we employ the following metric:  $\|\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}\| / \|\mathbf{w}_{k,n} - \mathbf{w}_{k,n-1}\|$ ,  $\forall k \in \mathcal{N}$ , or, more specifically, we restart the transmission

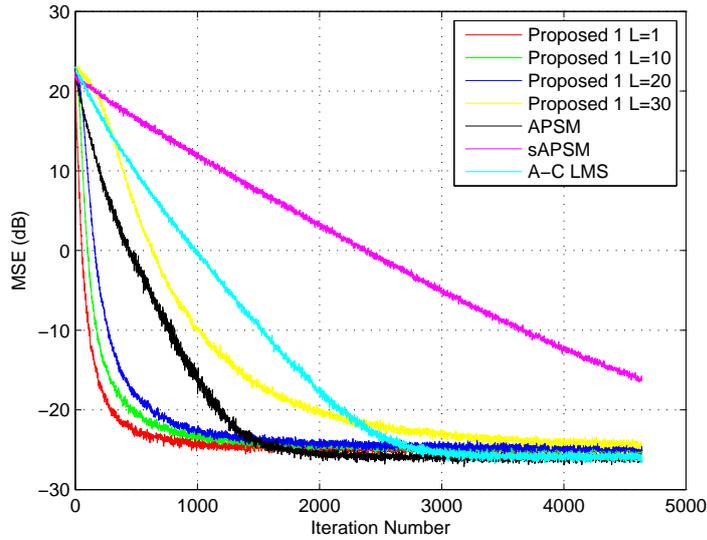


Figure 6.6: Average MSE for the third experiment.

of the input coefficients and the desired responses, if this ratio is greater than a threshold, which is chosen, here, to be equal to 10.

In the fifth experiment, we validate the performance of the whitening version (denoted as Proposed 2), in a strongly correlated environment. To this end, the parameter  $\theta_k$  takes values inside the interval  $(0.8, 1)$ . We compare the Proposed 1 for  $D = 10$ , the Proposed 2, for the following choices  $D = 10$ ,  $D = 20$ , the sAPSM, the full rank APSM and the A-C LMS. In the A-C LMS, we choose the largest step-size for which the algorithm converges, and it equals to  $10^{-3}$ . The rest of the parameters remain the same as in the previous experiments, and the forgetting factor which corresponds to the computation of  $\hat{\sigma}'_{i,n}$  equals to  $\gamma = 1$ . Fig. 6.8 illustrates that the performance of the Proposed 1 is degraded due to the highly correlated input. However, by employing the transformation, which whitens the input (Proposed 2), the performance is significantly enhanced, even if the dimension is relatively low, compared to the case where we employ the original input.

Finally, in the sixth experiment, we examine how the performance of the Proposed 1 is affected when the Krylov subspace is constructed based on information coming from a single node (see also subsection 6.2.1). To this end, we compare the Proposed 1 in the case where the subspace is constructed using information from every node, with the same algorithm in the cases where: a) the optimum node, b) the worst node and c) an arbitrary node, provide information in order to construct the subspace.  $D$  equals to 20 and the rest parameters are

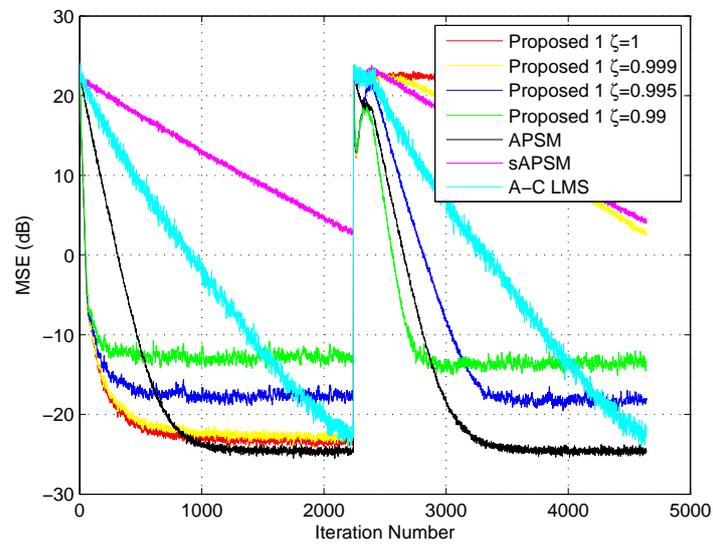


Figure 6.7: Average MSE for the fourth experiment.

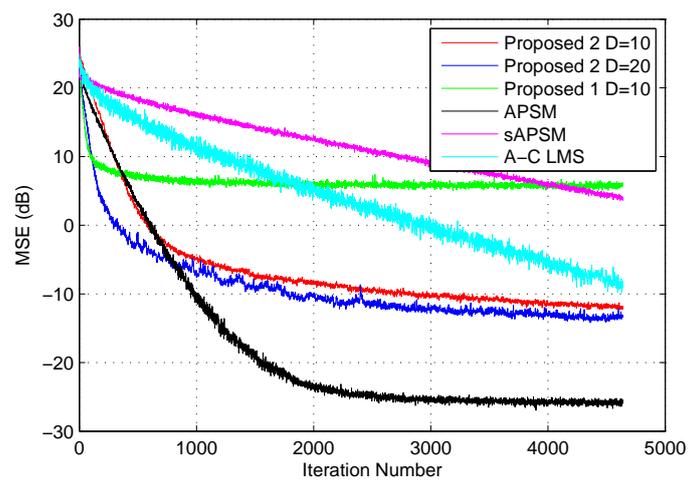


Figure 6.8: Average MSE for the fifth experiment.

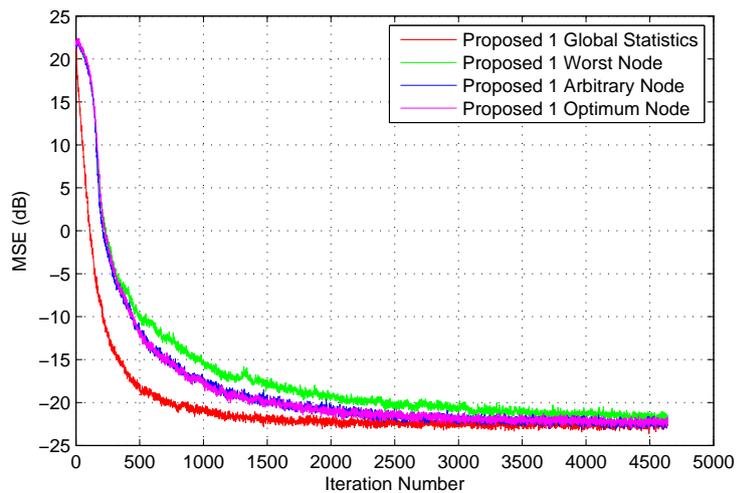


Figure 6.9: Average MSE for the sixth experiment.

the same as in the first experiment. The optimum node is the one with the less correlated input and the worst node is the one with the most correlated input. Fig. 6.9 shows that by using global information the algorithm converges faster. Nevertheless, the proposed scheme performs well even in the worst case scenario, where the node with the most correlated input is used in order to compute the subspaces. This results is very useful, in large networks, where using global information may be prohibited.

# Appendix F

## Projection Operators Onto Subspaces

The projection onto a closed convex set  $\mathcal{C}$ , in the  $\mathbf{W}$ -norm sense, where  $\mathbf{W}$  is an  $m \times m$  positive definite matrix, is given via the following optimization  $P_{\mathcal{C}}^{(\mathbf{W})}(\mathbf{w}) := \operatorname{argmin}_{\mathbf{x} \in \mathcal{C}} \|\mathbf{w} - \mathbf{x}\|_{\mathbf{W}}$ . Moreover, the projection of a point, say  $\mathbf{w}$ , onto a subspace, say  $V$ , is given by  $P_V(\mathbf{w}) = \mathbf{Q}\mathbf{Q}^T\mathbf{w}$ , where  $\mathbf{Q}$  is a matrix whose columns form a basis for  $V$ , whereas the projection of  $\mathbf{w}$  onto  $V$  in the  $\mathbf{W}$ -norm sense equals to  $P_V^{(\mathbf{W})}(\mathbf{w}) = \mathbf{Q}(\mathbf{Q}^T\mathbf{W}\mathbf{Q})\mathbf{Q}^T\mathbf{W}\mathbf{w}$ .

# Appendix G

## Proof of Claim 1

By basic linear algebraic arguments [130], it can be verified that the subspace  $K_n$  is isomorphic and isometric to  $\mathbb{R}^D$  via the mapping  $\hat{\mathbf{T}}_n : \mathbb{R}^D \rightarrow K_n$  where  $\hat{\mathbf{T}}_n$  is the matrix whose columns form an orthonormal basis for  $K_n$ .

Take the previously mentioned argument into consideration and fix a  $\phi \in K_n$ . Then notice that

$$\begin{aligned} \|\phi - P_{S_n \cap K_n}(\phi)\| &= \min_{\mathbf{w} \in S_n \cap K_n} \|\phi - \mathbf{w}\| \\ &= \min_{\tilde{\mathbf{w}} \in \tilde{S}_n} \|\tilde{\phi} - \tilde{\mathbf{w}}\| \\ &= \|\tilde{\phi} - P_{\tilde{S}_n}(\tilde{\phi})\| \\ &= \|\phi - \hat{\mathbf{T}}_n P_{\tilde{S}_n}(\tilde{\phi})\|, \end{aligned} \tag{G.1}$$

where  $\tilde{S}_n = \{\tilde{\mathbf{w}} \in \mathbb{R}^D : |d_n - \mathbf{u}^T \hat{\mathbf{T}}_n \tilde{\mathbf{w}}| \leq \epsilon\}$ .

By (G.1) and the uniqueness of the projection, we obtain

$$P_{S_n \cap K_n}(\phi) = \hat{\mathbf{T}}_n P_{\tilde{S}_n}(\tilde{\phi}) = \hat{\mathbf{T}}_n P_{\tilde{S}_n}(\hat{\mathbf{T}}_n^T \phi), \tag{G.2}$$

which completes our proof.

# Appendix H

## Proof of Claim 2

Recalling the arguments of Claim 1, it can be verified that

$$\tilde{\mathcal{M}}_{k,n} = \mathcal{M}_{k,n} \tag{H.1}$$

Moreover, it holds that  $\mathbf{w}_{k,n+1} = \hat{\mathbf{T}}_{n+1} \tilde{\mathbf{w}}_{k,n+1}$ . Going back to Eq. (6.8) and substituting  $\tilde{\boldsymbol{\phi}}_{k,n} = \hat{\mathbf{T}}_n^T \boldsymbol{\phi}_{k,n}$ ,  $P_{S_{k,n} \cap K_n}(\boldsymbol{\phi}_{k,n}) = \hat{\mathbf{T}}_n P_{\tilde{S}_{k,n}}(\tilde{\boldsymbol{\phi}}_{k,n}) \Rightarrow P_{\tilde{S}_{k,n}}(\tilde{\boldsymbol{\phi}}_{k,n}) = \hat{\mathbf{T}}_n^T P_{S_{k,n} \cap K_n}(\boldsymbol{\phi}_{k,n})$ , and if left multiply with  $\hat{\mathbf{T}}_{n+1}$  and equation (H.1) we obtain the desired result.

# Appendix I

## Proof of Theorems 1 and 2

We will prove Theorem 2, since Theorem 1 is a special case of it. To be more specific, the properties which will be proved in this appendix hold also for Theorem 1, if we substitute the matrix  $\mathbf{Z}'_{n_2}{}^{-\frac{1}{2}}\mathbf{Y}$  by  $\mathbf{I}_m$ .

### I.1 Monotonicity

First of all let us define  $\underline{\mathbf{h}}_* = \begin{bmatrix} \mathbf{h}_* \\ \vdots \\ \mathbf{h}_* \end{bmatrix} \in \mathbb{R}^{Nm}$ ,  $\forall \mathbf{h}_* \in \hat{\Omega}$  and  $\underline{\mathbf{h}}_n = \begin{bmatrix} \mathbf{h}_{1,n} \\ \vdots \\ \mathbf{h}_{N,n} \end{bmatrix}$ . Since  $\forall n \geq n_1$ , we have that  $\hat{\mathbf{T}}'_n = \hat{\mathbf{T}}'_{n+1} = \hat{\mathbf{T}}'_{n_1}$ , it holds that (see also [154])  $\hat{\mathbf{T}}'_{n+1}\hat{\mathbf{T}}_n'^T = P_{\hat{K}_{n_1}}$ . Fix a node, say  $k \in \mathcal{N}$ . We have that,  $\forall n \geq n'_0$

$$\|\mathbf{h}_{k,n+1} - \mathbf{h}_*\| = \left\| P_{\hat{K}_{n_1}} \left( \varphi_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{k,j} \cap \hat{K}_{n_1}}(\varphi_{k,n}) - \varphi_{k,n} \right) \right) - \mathbf{h}_* \right\|$$

However, from the definition of  $\hat{\Omega}$  and since  $\mathbf{h}_* \in \hat{\Omega} \Rightarrow \mathbf{h}_* \in \hat{K}_{n_1} \Leftrightarrow \mathbf{h}_* = P_{\hat{K}_{n_1}}(\mathbf{h}_*)$ ,  $\forall n \geq n'_0$  by definition. Hence

$$\|\mathbf{h}_{k,n+1} - \mathbf{h}_*\| = \left\| P_{\hat{K}_{n_1}} \left( \varphi_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{k,j} \cap \hat{K}_{n_1}}(\varphi_{k,n}) - \varphi_{k,n} \right) \right) - P_{\hat{K}_{n_1}}(\mathbf{h}_*) \right\|. \quad (\text{I.1})$$

A well known property of the projection operator, *e.g.*, [149], is the non-expansivity, *i.e.*, given a non-empty convex set, say  $\mathcal{C}$ ,  $\|P_{\mathcal{C}}(\mathbf{w}_1) - P_{\mathcal{C}}(\mathbf{w}_2)\| \leq \|\mathbf{w}_1 - \mathbf{w}_2\|, \forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^m$ .

Combining the previous with (I.1) we obtain

$$\|\mathbf{h}_{k,n+1} - \mathbf{h}_*\| \leq \left\| \varphi_{k,n} + \mu_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{k,j} \cap \hat{K}_{n_1}}(\varphi_{k,n}) - \varphi_{k,n} \right) - \mathbf{h}_* \right\|$$

Gathering the inequalities for every node, we obtain that

$$\left\| \begin{bmatrix} \mathbf{h}_{1,n+1} \\ \vdots \\ \mathbf{h}_{N,n+1} \end{bmatrix} - \underline{\mathbf{h}}_* \right\| \leq \left\| \begin{bmatrix} \varphi_{1,n} + \mu_{1,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{1,j} P_{S'_{1,j} \cap \hat{K}_{n_1}}(\varphi_{1,n}) - \varphi_{1,n} \right) \\ \vdots \\ \varphi_{N,n} + \mu_{N,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{N,j} P_{S'_{N,j} \cap \hat{K}_{n_1}}(\varphi_{N,n}) - \varphi_{N,n} \right) \end{bmatrix} - \underline{\mathbf{h}}_* \right\| \quad (\text{I.2})$$

Following similar steps as in [41, Theorem 1] and under assumptions (a')-(d') it can be proved that

$$\left\| \begin{bmatrix} \varphi_{1,n} + \mu_{1,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{1,j} \cap \hat{K}_{n_1}}(\varphi_{1,n}) - \varphi_{1,n} \right) \\ \vdots \\ \varphi_{N,n} + \mu_{N,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{S'_{N,j} \cap \hat{K}_{n_1}}(\varphi_{N,n}) - \varphi_{N,n} \right) \end{bmatrix} - \underline{\mathbf{h}}_* \right\| \leq \left\| \begin{bmatrix} \mathbf{h}_{1,n} \\ \vdots \\ \mathbf{h}_{N,n} \end{bmatrix} - \underline{\mathbf{h}}_* \right\| \quad (\text{I.3})$$

Combining (I.2), (I.3) we obtain

$$\|\underline{\mathbf{h}}_{n+1} - \underline{\mathbf{h}}_*\| \leq \|\underline{\mathbf{h}}_n - \underline{\mathbf{h}}_*\|, \quad \forall n \geq n'_0 \quad (\text{I.4})$$

From (I.4) we have

$$\left\| \begin{bmatrix} \mathbf{h}_{1,n+1} - \mathbf{h}_* \\ \vdots \\ \mathbf{h}_{N,n+1} - \mathbf{h}_* \end{bmatrix} \right\| \leq \left\| \begin{bmatrix} \mathbf{h}_{1,n} - \mathbf{h}_* \\ \vdots \\ \mathbf{h}_{N,n} - \mathbf{h}_* \end{bmatrix} \right\|, \quad \forall n \geq n'_0 \quad (\text{I.5})$$

However, if we take into consideration (6.17) and  $\forall n \geq n'_0$

$$\begin{bmatrix} \mathbf{h}_{1,n+1} - \mathbf{h}_* \\ \vdots \\ \mathbf{h}_{N,n+1} - \mathbf{h}_* \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_{n_2}^{-\frac{1}{2}'} \mathbf{Y}(\mathbf{w}_{1,n+1} - \hat{\mathbf{w}}'_*) \\ \vdots \\ \mathbf{Z}_{n_2}^{-\frac{1}{2}'} \mathbf{Y}(\mathbf{w}_{N,n+1} - \hat{\mathbf{w}}'_*) \end{bmatrix}.$$

Having as kick off point the previous equation, it is not difficult to obtain that  $\forall n \geq n'_0$

$$\left\| \begin{bmatrix} \mathbf{h}_{1,n+1} - \mathbf{h}_* \\ \vdots \\ \mathbf{h}_{N,n+1} - \mathbf{h}_* \end{bmatrix} \right\| = \left\| \begin{bmatrix} \mathbf{w}_{1,n+1} - \hat{\mathbf{w}}'_* \\ \vdots \\ \mathbf{w}_{N,n+1} - \hat{\mathbf{w}}'_* \end{bmatrix} \right\|_{\mathbf{G}}. \quad (\text{I.6})$$

Since  $\mathbf{A}$  is a positive definite matrix ([154]), it is not difficult to obtain that  $\mathbf{G}$  is also positive definite. Let us take a closer look on  $\hat{\mathbf{w}}'_*$ . Since by assumption  $\mathbf{h}_* \in \hat{K}_n \cap S'_{j,n}, \forall k \in \mathcal{N}, \forall j \in \mathcal{J}_n, \forall n \geq n'_0$  we have that there exists  $\tilde{\mathbf{h}}_* \in \mathbb{R}^D$  such that  $\mathbf{h}_* = \hat{\mathbf{T}}'_{n_1} \tilde{\mathbf{h}}_*$  and  $|\mathbf{h}_{k,j}^T \mathbf{h}_* - d_{k,j}| \leq \epsilon_k, \forall k \in \mathcal{N}, j \in \mathcal{J}_n, n \geq n'_0$ . The previous equations yield that  $\hat{\mathbf{w}}'_* = \mathbf{Y}^T \mathbf{Z}_{n_2}^{-\frac{1}{2}'} \hat{\mathbf{T}}'_{n_1} \tilde{\mathbf{h}}_* \Rightarrow \hat{\mathbf{w}}'_* \in \overline{M}$ , and since  $\mathbf{h}_{k,n}^T \mathbf{h}_* = \mathbf{u}_{k,n}^T \hat{\mathbf{w}}'_*$ , it holds that  $\hat{\mathbf{w}}'_* \in S_{k,j}, \forall k \in \mathcal{N}, n \geq n'_0$ .

Now, combining (I.5) and (I.6) implies

$$\|\underline{\mathbf{w}}_{n+1} - \underline{\mathbf{w}}'_*\|_{\mathbf{G}} \leq \|\underline{\mathbf{w}}_n - \underline{\mathbf{w}}'_*\|_{\mathbf{G}}, \forall n \geq n'_0, \quad (\text{I.7})$$

## I.2 Asymptotic Optimality

Let us define the following non-negative cost function  $\forall k \in \mathcal{N}$

$$\Theta_{k,n}(\mathbf{h}) = \begin{cases} \frac{1}{L_{k,n}} \sum_{j \in \mathcal{I}_{k,n}} \omega_{k,j} d_{(\mathbf{G})}(\boldsymbol{\varphi}_{k,n}, S'_{k,j} \cap \hat{K}_n) d_{(\mathbf{G})}(\mathbf{h}, S'_{k,j} \cap \hat{K}_n), & \text{if } \mathcal{I}_{k,n} \neq \emptyset \\ 0, & \text{if } \mathcal{I}_{k,n} = \emptyset, \end{cases} \quad (\text{I.8})$$

where  $\mathcal{I}_{k,n} := \{j \in \mathcal{J}_n : \boldsymbol{\varphi}_{k,n} \notin S_{k,j}\}$  and  $L_{k,n} = \sum_{j \in \mathcal{J}_n} \omega_{k,j} d_{(\mathbf{G})}(\boldsymbol{\varphi}_{k,n}, S'_{k,j} \cap \hat{K}_n)$ , It can be readily seen that since  $\mathbf{h}_* \in \Omega$ ,  $\Theta_{k,n}(\mathbf{h}_*) = 0, \forall k \in \mathcal{N}, \forall n \geq n_0$ . Following similar steps as in [41] and [82], we have that (6.10) can be equivalently written

$$\mathbf{h}_{k,n+1} = \begin{cases} P_{\hat{K}_n} \left( \boldsymbol{\varphi}_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\boldsymbol{\varphi}_{k,n})}{\|\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})\|^2} \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) \right), & \text{if } \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) \neq 0, \\ P_{\hat{K}_n}(\boldsymbol{\varphi}_{k,n}), & \text{if } \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) = 0, \end{cases} \quad (\text{I.9})$$

where  $\lambda_{k,n} = \frac{\mu_{k,n}}{\mathcal{M}'_{k,n}} \in (\varepsilon_1, 2 - \varepsilon_1)$  and with  $\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})$  we denote the subgradient of  $\Theta_{k,n}(\cdot)$  at the point  $\boldsymbol{\varphi}_{k,n}$ . Following similar steps as in [41, 45] and if assumptions (a')-(d') hold true, it can be proved that

$$\lim_{n \rightarrow \infty} \Theta_{k,n}(\boldsymbol{\varphi}_{k,n}) = 0, \quad \forall k \in \mathcal{N} \quad (\text{I.10})$$

which in turn implies that [41]:

$$\lim_{n \rightarrow \infty} d_{(\mathcal{G})}(\mathbf{h}_{k,n+1}, \hat{\Omega}_n) = 0, \quad \forall k \in \mathcal{N}. \quad (\text{I.11})$$

This yields that

$$\lim_{n \rightarrow \infty} \|\mathbf{h}_{k,n+1} - P_{\hat{\Omega}_n}(\mathbf{h}_{k,n+1})\| = 0. \quad (\text{I.12})$$

However, using similar arguments as before with  $\hat{\mathbf{w}}'_*$ , for every point  $\hat{\mathbf{h}}$  that lies in  $\hat{\Omega}_n$  it holds that  $\hat{\mathbf{w}} := \mathbf{Y} \mathbf{Z}_{n_2}^{-\frac{1}{2}'} \hat{\mathbf{h}} \in \bar{\Omega}_n$ . Hence, if we define  $\check{\mathbf{w}} = \mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}'} P_{\hat{\Omega}_n}(\mathbf{h}_{k,n+1})$ , we have

$$\begin{aligned} \|\mathbf{w}_{k,n+1} - P_{\bar{\Omega}_n}(\mathbf{w}_{k,n+1})\| &\leq \|\mathbf{w}_{k,n+1} - \check{\mathbf{w}}\| \\ &= \|\mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}'} \mathbf{h}_{k,n+1} - \mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}'} P_{\hat{\Omega}_n}(\mathbf{h}_{k,n+1})\| \\ &\leq \|\mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}'}\| \|\mathbf{h}_{k,n+1} - P_{\hat{\Omega}_n}(\mathbf{h}_{k,n+1})\| \end{aligned} \quad (\text{I.13})$$

where the first inequality holds from the definition of the distance function, as the vectors  $\check{\mathbf{w}}, P_{\bar{\Omega}_n}(\mathbf{w}_{k,n+1}) \in \bar{\Omega}_n$ . Taking limits in (I.13) and recalling (I.12), we conclude that  $\lim_{n \rightarrow \infty} d(\mathbf{w}_{k,n+1}, \bar{\Omega}_n) = 0$ .

## I.3 Asymptotic Consensus

Under assumptions (a')-(d'), the algorithmic scheme achieves asymptotic consensus, *i.e.*, [31]:

$$\lim_{n \rightarrow \infty} \|\mathbf{h}_{k,n} - \mathbf{h}_{l,n}\| = 0, \quad \forall k, l \in \mathcal{N}.$$

It has been proved [31], that the algorithmic scheme achieves asymptotic consensus, *i.e.*,  $\|\mathbf{h}_{k,n} - \mathbf{h}_{l,n}\| \rightarrow 0, n \rightarrow \infty, \forall k, l \in \mathcal{N}$  if and only if

$$\lim_{n \rightarrow \infty} \|\underline{\mathbf{h}}_n - P_{\mathcal{O}}(\underline{\mathbf{h}}_n)\| = 0. \quad (\text{I.14})$$

First of all, notice that  $\mathbf{h}_{k,n} \in \hat{K}_n, \forall n \in \mathbb{Z}_{\geq 0}$ . Since  $\boldsymbol{\varphi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l} \mathbf{h}_{l,n}$  is a convex combination of vectors which belong to  $\hat{K}_n$ , which is a convex set [17], then  $\boldsymbol{\varphi}_{k,n} \in \hat{K}_n$ . Hence  $P_{\hat{K}_{n_1}}(\boldsymbol{\varphi}_{k,n}) = \boldsymbol{\varphi}_{k,n}, \forall k \in \mathcal{N}, \forall n \geq n'_0$ . So,

$$\begin{aligned} \|\mathbf{h}_{k,n+1} - \boldsymbol{\varphi}_{k,n}\| &= \left\| P_{\hat{K}_{n_1}} \left( \boldsymbol{\varphi}_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\boldsymbol{\varphi}_{k,n})}{\|\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})\|^2} \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) \right) - \boldsymbol{\varphi}_{k,n} \right\| \\ &= \left\| P_{\hat{K}_{n_1}} \left( \boldsymbol{\varphi}_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\boldsymbol{\varphi}_{k,n})}{\|\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})\|^2} \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) \right) - P_{\hat{K}_{n_1}}(\boldsymbol{\varphi}_{k,n}) \right\| \\ &\leq \left\| \boldsymbol{\varphi}_{k,n} - \lambda_{k,n} \frac{\Theta_{k,n}(\boldsymbol{\varphi}_{k,n})}{\|\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})\|^2} \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) - \boldsymbol{\varphi}_{k,n} \right\| \\ &= \left\| \lambda_{k,n} \frac{\Theta_{k,n}(\boldsymbol{\varphi}_{k,n})}{\|\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})\|^2} \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) \right\| \rightarrow 0, \end{aligned}$$

where in the inequality we have used the nonexpansivity of the projection operator onto a closed convex set and the limit on the last equality holds true as  $\left\| \lambda_{k,n} \frac{\Theta_{k,n}(\boldsymbol{\varphi}_{k,n})}{\|\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})\|^2} \Theta'_{k,n}(\boldsymbol{\varphi}_{k,n}) \right\| \leq (2 - \varepsilon_1) \frac{\Theta_{k,n}(\boldsymbol{\varphi}_{k,n})}{\|\Theta'_{k,n}(\boldsymbol{\varphi}_{k,n})\|} \rightarrow 0$  which holds from (I.10) and [45]. Now, it can be readily seen that

$$\lim_{n \rightarrow \infty} \|\mathbf{h}_{k,n+1} - \boldsymbol{\varphi}_{k,n}\| = 0, \forall k \in \mathcal{N} \quad (\text{I.15})$$

If we generalize (I.15) for the whole network, we have

$$\lim_{n \rightarrow \infty} \|\underline{\mathbf{h}}_{n+1} - \mathbf{P}\underline{\mathbf{h}}_n\| = 0, \quad (\text{I.16})$$

where  $\mathbf{P}$  is the consensus matrix defined in Chapters 4, 5. Having as kick off point (I.16) and if we follow similar steps as in [45] it can be verified that  $\lim_{n \rightarrow \infty} (\mathbf{I}_{N_m} - \mathbf{B}\mathbf{B}^T)\underline{\mathbf{h}}_{n+1} = 0$ .

The previous relation implies that

$$\lim_{n \rightarrow \infty} \|\mathbf{h}_{k,n} - \mathbf{h}_{l,n}\| = 0, \forall k, l \in \mathcal{N}. \quad (\text{I.17})$$

Hence,  $\forall n \geq n_1, \|\mathbf{w}_{k,n} - \mathbf{w}_{l,n}\| \leq \|\mathbf{Y}^T \hat{\mathbf{Z}}_{n_1}^{\frac{1}{2}}\| \|\mathbf{h}_{k,n} - \mathbf{h}_{l,n}\|$ . Taking limits and recalling (I.17) completes our proof.

## I.4 Strong Convergence

Following similar steps as in Claim 2, it can be proved that the algorithm in (6.18) can be equivalently written:

$$\tilde{\mathbf{h}}_{k,n+1} = \tilde{\varphi}_{k,n} + \tilde{\mu}_{k,n} \left( \sum_{j \in \mathcal{J}_n} \omega_{k,j} P_{\tilde{S}'_{k,j}}(\tilde{\varphi}_{k,n}) - \tilde{\varphi}_{k,n} \right), \quad (\text{I.18})$$

where  $\tilde{S}'_{k,j} := \{\tilde{\mathbf{w}} \in \mathbb{R}^D : |d_{k,n} - \boldsymbol{\psi}_{k,n}^T \hat{\mathbf{T}}'_n \tilde{\mathbf{w}}| \leq \epsilon_k\}$ . Notice that the algorithm in (I.18) is a special case of the algorithm proposed in [41]. The difference is that in the latter, the combined information coming from the nodes of the neighbourhood is projected onto a convex set, before the adaptation step. So, the convergence analysis, which took place in [41] holds for the scheme presented in (6.8). In order to verify this, we have to examine if the assumptions, under which the scheme converges, hold here too. First of all notice that under Assumption (a),  $\exists \mathbf{h}_0 \in \hat{\Omega}$ . From the previous we have that  $\exists \tilde{\mathbf{h}}_0 \in \mathbb{R}^D$  such that  $\tilde{\mathbf{h}}_0 = \hat{\mathbf{T}}_{n_1}'^T \mathbf{h}_0$ . Moreover, since  $\mathbf{h}_0$  belongs to the intersection of the hyperslabs, it satisfies  $|d_{k,n} - \mathbf{h}_{k,n}^T \mathbf{h}_0| \leq \epsilon_k, \forall k \in \mathcal{N}, \forall n \geq n'_0$ . So, we have that

$$|d_{k,j} - \mathbf{h}_{k,j}^T \mathbf{h}_0| \leq \epsilon_k \Leftrightarrow |d_{k,j} - \mathbf{h}_{k,j}^T \hat{\mathbf{T}}_{n_1}' \tilde{\mathbf{h}}_0| \leq \epsilon_k, \quad \forall k \in \mathcal{N}, \forall j \in \mathcal{J}_n, \forall n \geq n'_0. \quad (\text{I.19})$$

From the previous we have that there exists  $\tilde{\mathbf{h}}_0 \in \mathbb{R}^D$ , such that  $\tilde{\mathbf{h}}_0 \in \tilde{S}'_{k,j}, \forall k \in \mathcal{N}, \forall j \in \mathcal{J}_n, \forall n \geq n'_0$ . Thus,  $\hat{\Omega} \neq \emptyset$ . Moreover,  $\tilde{\mu}_{k,n} \in (0, 2\tilde{\mathcal{M}}'_{k,n})$ . In [41, Theorem 1.1] it has been proved, that these two facts, together with Assumption (d) are the assumptions under which the algorithm converges to a point, i.e.,

$$\lim_{n \rightarrow \infty} \tilde{\mathbf{h}}_n = \tilde{\mathbf{h}}_O, \quad (\text{I.20})$$

where  $\tilde{\mathbf{h}}_O := [\tilde{\mathbf{h}}_O^T, \dots, \tilde{\mathbf{h}}_O^T]^T \in \tilde{\mathcal{O}}$ . Taking into consideration (I.20) it follows that  $\lim_{n \rightarrow \infty} \tilde{\mathbf{h}}_{k,n} = \tilde{\mathbf{h}}_O, \forall k \in \mathcal{N}$ . Our proof is complete, since from the previous equation we have that

$$\lim_{n \rightarrow \infty} \mathbf{h}_{k,n} = \lim_{n \rightarrow \infty} \hat{\mathbf{T}}_{n_1}' \tilde{\mathbf{h}}_{k,n} = \hat{\mathbf{T}}_{n_1}' \tilde{\mathbf{h}}_O.$$

If we write the previous relation for all the nodes of the network we obtain

$$\lim_{n \rightarrow \infty} \underline{\mathbf{h}}_n = \underline{\mathbf{h}}_O,$$

where  $\underline{\mathbf{h}}_O = [(\hat{\mathbf{T}}'_{n_1} \tilde{\mathbf{h}}_O)^T, \dots, (\hat{\mathbf{T}}'_{n_1} \tilde{\mathbf{h}}_O)^T]^T$ . In words, the algorithm converges to a point, which lies in the consensus subspace.

According to the previous discussion we have that  $\underline{\mathbf{w}}_n = \overline{\mathbf{G}} \underline{\mathbf{h}}_n, \forall n \geq n'_0$ , with  $\overline{\mathbf{G}} = \text{diag} \underbrace{\{\mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}'}, \dots, \mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}'}\}}_N$ . Recall that from Theorem 1, we have  $\lim_{n \rightarrow \infty} \underline{\mathbf{h}}_n = \underline{\mathbf{h}}_O$ , where  $\underline{\mathbf{h}}_O \in \mathcal{O}$ . Hence,

$$\lim_{n \rightarrow \infty} \underline{\mathbf{w}}_n = \lim_{n \rightarrow \infty} \overline{\mathbf{G}} \underline{\mathbf{h}}_n = \overline{\mathbf{G}} \underline{\mathbf{h}}_O = \underline{\mathbf{w}}'_O. \quad (\text{I.21})$$

Since consensus holds for  $\underline{\mathbf{h}}_O$ , i.e.,  $\underline{\mathbf{h}}_O = \begin{bmatrix} \mathbf{h}_O \\ \vdots \\ \mathbf{h}_O \end{bmatrix}$ , it can be readily obtained that  $\underline{\mathbf{w}}'_O =$

$\begin{bmatrix} \mathbf{w}'_O \\ \vdots \\ \mathbf{w}'_O \end{bmatrix}$ , hence  $\underline{\mathbf{w}}'_O \in \mathcal{O}$ . Finally, since  $\mathbf{w}'_O = \mathbf{Y}^T \mathbf{Z}_{n_2}^{\frac{1}{2}'} \hat{\mathbf{T}}'_{n_1} \tilde{\mathbf{h}}_O \Rightarrow \mathbf{w}'_O \in \overline{\mathbf{M}}$ , which finishes our proof.

# Chapter 7

## Conclusions and Future Work

*This dissertation presented and studied distributed algorithms for adaptive learning in decentralized/ad-hoc networks. The proposed schemes belong to the family of the Adaptive Projected Subgradient Method and the nodes cooperate with each other by adopting the diffusion optimization rationale. This Chapter presents a summary of the work, as well as possible directions for future research.*

### 7.1 Summary

Initially, the problem of adaptive parameter estimation in ad-hoc networks was considered. In general, despite the fact that the nodes share a common goal, *i.e.*, the estimation of an unknown parameter vector, the input and the noise statistics differ from node to node. To overcome possible problems, which occur due to these diversities, a novel Combine-Project-Adapt cooperation protocol was introduced. The physical reasoning of the intermediate projection step was to “harmonize” the local information with the information coming from the neighborhood. The proposed algorithm exhibited a significant performance enhancement at the minimal expense of an extra projection.

Wireless Sensor Networks usually consist of cheap and sensitive sensors and, henceforth, various problems may occur to a number of them. To this end, we considered a scenario where a number of nodes are malfunctioning and the associated observations are very noisy. The problem was successfully dealt by employing the Huber cost function, which is robust when the data are corrupted with outliers. Projections onto halfspaces, associated with the

subgradient of the Huber function, were employed. Numerical examples showed that the proposed scheme outperformed significantly other distributed algorithms, which do not take into consideration the presence of outliers.

Many signals/systems adhere to parsimonious models. That is, the vector to be estimated is sparse. The sparse nature of the system can be exploited by embedding into the problem  $\ell_1$ -norm or weighted  $\ell_1$ -norm constraints. Furthermore, one can promote sparse solutions by modifying properly the projection operators. More analytically, sparsity can be exploited if the Euclidean projections are substituted by the variable metric projections, which weight heavier coefficients which belong to the support set of the unknown vector, compared to the rest. The proposed sparsity-promoting adaptive distributed algorithm combined both of these features and was developed according to the diffusion optimization strategy.

Finally, the problem of dimensionality reduction in ad-hoc networks was studied. The cooperation among the nodes demands that at every time instant each node will transmit a number of coefficients, which is at least equal to the dimension of the unknown vector. In scenarios where this dimension is large, the information exchange can be a burden. To this end, a reduced rank distributed adaptive scheme was introduced, where instead of seeking for the unknown vector in the original space, one searches for the projection of it onto a lower dimension subspace. The involved subspaces were the so-called Krylov subspaces, which are related to the reduced rank Wiener filter. As it was experimentally verified, the proposed algorithm served a good trade-off between the performance and the number of transmitted coefficients.

## 7.2 Future Work

In this section, we point out possible directions and issues for future research.

- Throughout this dissertation, we considered that the measurements, through which the unknown vector is estimated, are related via the linear system. Nevertheless, in several applications, such as satellite, mobile communications, high definition TV, just to name a few, a nonlinear behavior is observed. To this end, the generalization of the algorithms presented here in scenarios where the input and the output are related via an unknown nonlinear function, can be investigated.

- In sparsity-aware learning one seeks for an unknown vector, which has a small number of non-zero coefficients. Hence, low complexity schemes, whose complexity is of the order of the number of non-zero coefficients, instead of the full dimensionality, can be developed. Such techniques can also reduce the number of transmitted coefficients and, henceforth, the bandwidth of the network.
- The algorithm of Chapter 6 employed the Krylov subspace rationale for dimensionality reduction, which reduced the number of transmitted coefficients. Reduced rank techniques of lower complexity can be developed. To this direction, sophisticated techniques for subspace tracking, proposed in the literature, can be combined with adaptive learning techniques so as to seek for the unknown vector (using the adaptive algorithms) within a lower dimension subspace, which will be estimated via the subspace learning procedure. Through this methodology, fewer coefficients will be transmitted.
- A problem of great importance and interest in distributed learning is to develop decentralized classification and clustering algorithms. The proposed APSM based algorithms can be modified so as to minimize properly chosen cost functions, which will be suitable for supervised/unsupervised learning. The resulting algorithms can of relatively low complexity and suitable for online learning and general enough to cope with several learning scenarios.



# List of Abbreviations

- APSM: Adaptive Projected Subgradient Method
- dB: decibel
- DCT: Discrete Cosine Transformation
- EMSE: Excess Mean Square Error
- i.i.d.: Independent and Identically Distributed
- Lasso: Least Absolute Shrinkage and Selection Operator
- LMS: Least Mean Squares
- LS: Least Squares
- MSD: Mean Square Deviation
- MSE: Mean Square Error
- PCA: Principal Component Analysis
- POCS: Projections Onto Convex Sets
- RLS: Recursive Least Squares
- WSN: Wireless Sensor Network



# List of symbols

- $\mathbf{w}^T$  Transpose of a vector
- $\mathbf{A}^T$  Transpose of a matrix
- $\otimes$  Kronecker Product
- $\|\mathbf{w}\|$  Euclidean Norm
- $\|\mathbf{w}\|_1$   $\ell_1$  Norm
- $\|\mathbf{w}\|_0$   $\ell_0$  Norm
- $\|\mathbf{w}\|_{\mathbf{A}}$  Weighted Norm
- $|\mathcal{S}|$  Cardinality of the set  $\mathcal{S}$
- $\text{supp}(\cdot)$  Support set of a vector
- $\langle \cdot, \cdot \rangle$  Inner Product
- $\langle \cdot, \cdot \rangle_{\mathbf{A}}$  Weighted Inner Product
- $I$  Identity Mapping
- $I_m$   $m \times m$  Identity Matrix.
- $I_{Nm}$   $Nm \times Nm$  Identity Matrix.
- $\mathbf{0}_m$   $m \times 1$  Zero Vector.
- $\mathbf{0}_{Nm}$   $Nm \times 1$  Zero Vector.
- $\mathbf{1}_m$   $m \times 1$  Ones Vector.

## Conclusions and Future Work

---

- $\mathbf{1}_{Nm}$   $Nm \times 1$  Ones Vector.
- $\mathbb{R}$  Set of All Real Numbers
- $\mathbb{Z}$  Set of All Integers Numbers

# Bibliography

- [1] Proceedings of the distributed sensor nets workshop. Pittsburgh.
- [2] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.
- [3] David S Alberts, John J Garstka, and Frederick P Stein. Network centric warfare: Developing and leveraging information superiority. Technical report, DTIC Document, 2000.
- [4] D. Angelosante, J.A. Bazerque, and G.B. Giannakis. Online adaptive estimation of sparse signals: where RLS meets the  $\ell_1$ -norm. *IEEE Transactions on Signal Processing*, 58(7):3436–3447, 2010.
- [5] Behtash Babadi, Nicholas Kalouptsidis, and Vahid Tarokh. Sparls: The sparse rls algorithm. *IEEE Transactions on Signal Processing*, 58(8):4013–4025, 2010.
- [6] R. Baraniuk. Compressive sensing. *Lecture notes in IEEE Signal Processing magazine*, 24(4):118–120, 2007.
- [7] Juan Andrés Bazerque and Georgios B Giannakis. Distributed spectrum sensing for cognitive radio networks by exploiting sparsity. *IEEE Transactions on Signal Processing*, 58(3):1847–1862, 2010.
- [8] Jacob Benesty and Steven L. Gay. An improved pnllms algorithm. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1881–1884, may 2002.
- [9] Florence Benezit, Patrick Thiran, and Martin Vetterli. Interval consensus: from quantized gossip to voting. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 3661–3664. IEEE, 2009.
- [10] Dimitri P Bertsekas and John N Tsitsiklis. Parallel and distributed computation. 1999.
- [11] D.P. Bertsekas, A. Nedic, and A.E. Ozdaglar. *Convex analysis and optimization*. Athena Scientific, 2003.
- [12] Doron Blatt and Alfred O Hero. Energy-based sensor network source localization via projection onto convex sets. *Signal Processing, IEEE Transactions on*, 54(9):3614–3619, 2006.
- [13] V.D. Blondel, J.M. Hendrickx, A. Olshevsky, and J.N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05*, pages 2996–3000, 2005.
- [14] Béla Bollobás. *Modern graph theory*, volume 184. Springer Verlag, 1998.

## BIBLIOGRAPHY

---

- [15] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Analysis and optimization of randomized gossip algorithms. In *IEEE Conference on Decision and Control*, volume 5, pages 5310–5315. IEEE, 2004.
- [16] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *Information Theory, IEEE Transactions on*, 52(6):2508–2530, 2006.
- [17] Stephen Boyd and Lieven Vandenbergh. *Convex optimization*. Cambridge University Press, 2004.
- [18] LM Bregman. The method of successive projections for finding a common point of convex sets. *Soviet Math. Dokl.*, pages 688–692, 1965.
- [19] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.
- [20] E.J. Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 1433–1452. Citeseer, 2006.
- [21] E.J. Candes and M.B Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.
- [22] E.J. Candes, M.B. Wakin, and S.P. Boyd. Enhancing sparsity by reweighted  $\ell_1$  minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.
- [23] Federico S Cattivelli and Ali H Sayed. Hierarchical diffusion algorithms for distributed estimation. In *SSP*, pages 537–540. IEEE, 2009.
- [24] Federico S Cattivelli and Ali H Sayed. Multi-level diffusion adaptive networks. In *ICASSP*, pages 2789–2792. IEEE, 2009.
- [25] Federico S Cattivelli and Ali H Sayed. Distributed detection over adaptive networks using diffusion adaptation. *IEEE Transactions on Signal Processing*, 59(5):1917–1932, 2011.
- [26] Federico S Cattivelli and Ali H Sayed. Distributed detection over adaptive networks using diffusion adaptation. *Signal Processing, IEEE Transactions on*, 59(5):1917–1932, 2011.
- [27] 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*, 56(5):1865–1877, 2008.
- [28] F.S. Cattivelli and A.H. Sayed. Diffusion lms strategies for distributed estimation. *IEEE Transactions on Signal Processing*, 58(3):1035–1048, 2010.
- [29] F.S. Cattivelli and A.H. Sayed. Diffusion strategies for distributed kalman filtering and smoothing. *IEEE Transactions on Automatic Control*, 55(9):2069–2084, 2010.
- [30] RLG Cavalcante and S Stanczak. Robust set-theoretic distributed detection in diffusion networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3757–3760. IEEE, 2012.
- [31] R.L.G. Cavalcante, I. Yamada, and B. Mulgrew. An adaptive projected subgradient approach to learning in diffusion networks. *Signal Processing, IEEE Transactions on*, 57(7):2762–2774, 2009.

- 
- [32] J. Chen and A.H. Sayed. Distributed pareto optimization via diffusion strategies. *IEEE Journal of Selected Topics in Signal Processing*, 7(2):205–220, 2013.
- [33] Jianshu Chen and Ali H Sayed. Bio-inspired cooperative optimization with application to bacteria motility. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5788–5791. IEEE, 2011.
- [34] Jianshu Chen and Ali H Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. *Signal Processing, IEEE Transactions on*, 60(8):4289–4305, 2012.
- [35] Yilun Chen, Yuantao Gu, and A.O. Hero. Sparse LMS for system identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3125–3128, 2009.
- [36] Chee-Yee Chong and Srikanta P Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.
- [37] Chee-Yee Chong and Srikanta P Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.
- [38] Chee-Yee Chong, Shozo Mori, and Kuo-Chu Chang. Distributed multitarget multisensor tracking. *Multitarget-multisensor tracking: Advanced applications*, 1:247–295, 1990.
- [39] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis. A sparsity promoting adaptive algorithm for distributed learning. *IEEE Transactions on Signal Processing*, 60(10):5412–5425, oct. 2012.
- [40] S. Chouvardas, K. Slavakis, and S. Theodoridis. A novel adaptive algorithm for diffusion networks using projections onto hyperslabs. In *CIP*, pages 393–398. IEEE, 2010.
- [41] S. Chouvardas, K. Slavakis, and S. Theodoridis. Adaptive robust distributed learning in diffusion sensor networks. *IEEE Transactions on Signal Processing*, 59(10):4692–4707, 2011.
- [42] S. Chouvardas, K. Slavakis, and S. Theodoridis. Trading off communications bandwidth with accuracy in adaptive diffusion networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2048–2051. IEEE, 2011.
- [43] S. Chouvardas, K. Slavakis, and S. Theodoridis. Trading off complexity with communication costs in distributed adaptive learning via krylov subspaces for dimensionality reduction. *IEEE Journal of Selected Topics in Signal Processing*, 7(2):257–273, 2013.
- [44] Symeon Chouvardas, Gerasimos Mileounis, Nicholaos Kalouptsidis, and Sergios Theodoridis. A greedy sparsity-promoting LMS for distributed adaptive learning in diffusion networks. In *In Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013.
- [45] Symeon Chouvardas, Konstantinos Slavakis, Yannis Kopsinis, and Sergios Theodoridis. Sparsity-promoting adaptive algorithm for distributed learning in diffusion networks. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1084–1088. IEEE, 2012.
- [46] Symeon Chouvardas, Konstantinos Slavakis, Sergios Theodoridis, and Isao Yamada. Stochastic analysis of hyperslab-based adaptive projected subgradient method under bounded noise. *To appear on IEEE Signal Processing Letters*, 2013.

## BIBLIOGRAPHY

---

- [47] J Cioffi and Thomas Kailath. Fast, recursive-least-squares transversal filters for adaptive filtering. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 32(2):304–337, 1984.
- [48] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4(2):28–34, 2002.
- [49] P. Di Lorenzo and A.H. Sayed. Sparse distributed learning based on diffusion adaptation. *IEEE Transactions on Signal Processing*, 61(6):1419–1433, 2013.
- [50] Paolo Di Lorenzo, Sergio Barbarossa, and Ali H Sayed. Sparse diffusion lms for distributed adaptive estimation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3281–3284. IEEE, 2012.
- [51] Alexandros G Dimakis, Soumya Kar, Jose MF Moura, Michael G Rabbat, and Anna Scaglione. Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11):1847–1864, 2010.
- [52] Paulo SR Diniz. *Adaptive filtering: algorithms and practical implementation*. Springer, 2013.
- [53] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
- [54] D.L. Duttweiler. Proportionate normalized least-mean-squares adaptation in echo cancelers. *IEEE Transactions on Speech and Audio Processing*, 8(5):508–518, September 2000.
- [55] Deborah Estrin, Lewis Girod, Greg Pottie, and Mani Srivastava. Instrumenting the world with wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 4, pages 2033–2036. IEEE, 2001.
- [56] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the ACM/IEEE international conference on Mobile computing and networking*, pages 263–270. ACM, 1999.
- [57] J Alexander Fax and Richard M Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.
- [58] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.
- [59] M. Gerla and J.T.C. Tsai. Multicluster, mobile, multimedia radio network. *Wireless networks*, 1(3):255–265, 1995.
- [60] G-O Glentis, Kostas Berberidis, and Sergios Theodoridis. Efficient least squares adaptive algorithms for fir transversal filtering. *IEEE Signal Processing Magazine*, 16(4):13–41, 1999.
- [61] J.S. Goldstein, I.S. Reed, and L.L. Scharf. A multistage representation of the Wiener filter based on orthogonal projections. *IEEE Transactions on Information Theory*, 44(7):2943–2959, 1980.
- [62] Yuko Hatano, Arindam K Das, and Mehran Mesbahi. Agreement in presence of noise: pseudogradients on random geometric networks. In *IEEE Conference on Decision and Control, European Control Conference. CDC-ECC'05*, pages 6382–6387. IEEE, 2005.

- 
- [63] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, 4th edition, 2002.
- [64] Simon Haykin. Cognitive radio: brain-empowered wireless communications. *IEEE journal on selected areas in communications*, 23(2):201–220, 2005.
- [65] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms: Fundamentals*. Springer, 1993.
- [66] M.L. Honig and J.S. Goldstein. Adaptive reduced-rank interference suppression based on the multistage Wiener filter. *IEEE Transactions on Communications*, 50(6):986–994, 2002.
- [67] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge university press, 2005.
- [68] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge university press, 2005.
- [69] Minyi Huang and Jonathan H Manton. Stochastic lyapunov analysis for consensus algorithms with noisy measurements. In *American Control Conference, 2007. ACC'07*, pages 1419–1424. IEEE, 2007.
- [70] PJ Huber and EM Ronchetti. *Robust Statistics*. New York: Wiley, 2009.
- [71] D Jensen. Sivam: Communication, navigation and surveillance for the amazon. *Avionics Mag*, 2002.
- [72] Chunxiao Jiang, Yan Chen, and KJ Liu. Distributed adaptive networks: A graphical evolutionary game-theoretic view. *arXiv preprint arXiv:1212.1245*, 2012.
- [73] M. Joham and M. D. Zoltowski. Interpretation of the Multi-Stage Nested Wiener Filter in the Krylov Subspace Framework. *Tech. Rep. TUMLNS-TR-00-6, Munich University of Technology*, 2000.
- [74] Emil Jovanov, Aleksandar Milenkovic, Chris Otto, and Piet C De Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and rehabilitation*, 2(1):1–10, 2005.
- [75] Nicholas Kalouptsidis, Gerasimos Mileounis, Behtash Babadi, and Vahid Tarokh. Adaptive algorithms for sparse system identification. *Signal Processing*, 91(8):1910–1919, 2011.
- [76] A. Kansal, S.N. Batalama, and D.A. Pados. Adaptive maximum sinr rake filtering for ds-cdma multipath fading channels. *IEEE Journal on Selected Areas in Communications*, 16(9):1765–1773, dec 1998.
- [77] Soumya Kar, Saeed Aldosari, and José MF Moura. Topology for distributed inference on graphs. *IEEE Transactions on Signal Processing*, 56(6):2609–2613, 2008.
- [78] Soumya Kar and José MF Moura. Sensor networks with random links: Topology design for distributed consensus. *IEEE Transactions on Signal Processing*, 56(7):3315–3326, 2008.
- [79] Soumya Kar and José MF Moura. Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise. *IEEE Transactions on Signal Processing*, 57(1):355–369, 2009.
- [80] Soumya Kar and Jose MF Moura. Convergence rate analysis of distributed gossip (linear parameter) estimation: Fundamental limits and tradeoffs. *IEEE Journal of Selected Topics in Signal Processing*, 5(4):674–690, 2011.

- [81] Akshay Kashyap, Tamer Başar, and Ramakrishnan Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.
- [82] Y. Kopsinis, K. Slavakis, and S. Theodoridis. Online sparse system identification and signal reconstruction using projections onto weighted  $\ell_1$  balls. *IEEE Transactions on Signal Processing*, 59(3):936–952, 2011.
- [83] A. N. Krylov. On the numerical solution of equation by which are determined in technical problems the frequencies of small vibrations of material systems. *Mathematics of Computation*, 4:491–539, 1931.
- [84] Sen M Kuo, Yu C Huang, and Zhibing Pan. Acoustic noise and echo cancellation microphone system for videoconferencing. *IEEE Transactions on Consumer Electronics*, 41(4):1150–1158, 1995.
- [85] Sen M Kuo and Zhibing Pan. Development and analysis of distributed acoustic echo cancellation microphone system. *Signal processing*, 37(3):333–344, 1994.
- [86] Javad Lavaei and Richard M Murray. On quantized consensus by means of gossip algorithm—part ii: Convergence time. In *American Control Conference, ACC'09.*, pages 2958–2965. IEEE, 2009.
- [87] Adrian S Lewis and Michael L Overton. Eigenvalue optimization. *Acta numerica*, 5(1):149–190, 1996.
- [88] Jinchao Li and Ali H Sayed. Modeling bee swarming behavior through diffusion adaptation with asymmetric information sharing. *EURASIP Journal on Advances in Signal Processing*, 18, 2012.
- [89] L. Li and J.A. Chambers. A new incremental affine projection-based adaptive algorithm for distributed networks. *Signal Processing*, 88(10):2599–2603, 2008.
- [90] L. Li, J.A. 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*, 58(1):151–164, 2010.
- [91] M. Loève. *Probability Theory*, volume 1. Springer-Verlag, New York, fourth edition, 1977.
- [92] C.G. Lopes and A.H. Sayed. Distributed adaptive incremental strategies: formulation and performance analysis. In *In Proc. IEEE ICASSP*, volume 3, pages 584–587. IEEE, 2006.
- [93] C.G. Lopes and A.H. Sayed. Incremental adaptive strategies over distributed networks. *IEEE Transactions on Signal Processing*, 55(8):4064–4077, 2007.
- [94] C.G. Lopes and A.H. Sayed. Diffusion least-mean squares over adaptive networks: Formulation and performance analysis. *IEEE Transactions on Signal Processing*, 56(7):3122–3136, 2008.
- [95] A. Maleki and D.L. Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):330–341, 2010.
- [96] G. Mateos, J.A. Bazerque, and G.B. Giannakis. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10):5262–5276, 2010.

- 
- [97] Gonzalo Mateos and Georgios B Giannakis. Distributed recursive least-squares: Stability and performance analysis. *Signal Processing, IEEE Transactions on*, 60(7):3740–3754, 2012.
- [98] Gonzalo Mateos, Ioannis D Schizas, and Georgios B Giannakis. Distributed recursive least-squares for consensus-based in-network adaptive estimation. *IEEE Transactions on Signal Processing*, 57(11):4583–4588, 2009.
- [99] John McQuillan, Ira Richer, and Eric Rosen. The new routing algorithm for the arpanet. *IEEE Transactions on Communications*, 28(5):711–719, 1980.
- [100] Gerasimos Mileounis, Behtash Babadi, Nicholas Kalouptsidis, and Vahid Tarokh. An adaptive greedy algorithm with application to nonlinear communications. *IEEE Transactions on Signal Processing*, 58(6):2998–3007, 2010.
- [101] George V Moustakides and Sergios Theodoridis. Fast newton transversal filters—a new class of adaptive estimation algorithms. *IEEE Transactions on Signal Processing*, 39(10):2184–2193, 1991.
- [102] E.J. Msechu, S.I. Roumeliotis, A. Ribeiro, and G.B. Giannakis. Decentralized quantized kalman filtering with scalable communication cost. *IEEE Transactions on Signal Processing*, 56(8):3727–3741, 2008.
- [103] Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada. A sparse adaptive filtering using time-varying soft-thresholding techniques. In *IEEE International Conference on Acoustics Speech and Signal Processing, Dallas, USA*, pages 3734–3737. IEEE, 2010.
- [104] Cory Myers, Alan Oppenheim, Randall Davis, and Webster Dove. Knowledge based speech analysis and enhancement. In *Int. Conf.on Acoustics, Speech, and Signal Processing*, volume 9, pages 162–165. IEEE, 1984.
- [105] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.
- [106] Clyde E Nishimura and Dennis M Conlon. Iuss dual use: Monitoring whales and earthquakes using sosus. *Marine Technology Society Journal*, 27:13–21, 1994.
- [107] C.H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.
- [108] S.S. Pereira and A. Pages-Zamora. Distributed consensus in wireless sensor networks with quantized information exchange. In *Signal Processing Advances in Wireless Communications, 2008. SPAWC 2008. IEEE 9th Workshop on*, pages 241–245, july 2008.
- [109] G Pierra. Decomposition through formalization in a product space. *Mathematical Programming*, 28(1):96–115, 1984.
- [110] B.T. Polyak. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, 9(3):14–29, 1969.
- [111] Joel B Predd, SB Kulkarni, and H Vincent Poor. Distributed learning in wireless sensor networks. *Signal Processing Magazine, IEEE*, 23(4):56–69, 2006.
- [112] Zhi Quan, Shuguang Cui, and Ali H Sayed. Optimal linear cooperation for spectrum sensing in cognitive radio networks. *Selected Topics in Signal Processing, IEEE Journal of*, 2(1):28–40, 2008.

## BIBLIOGRAPHY

---

- [113] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 20–27. ACM New York, NY, USA, 2004.
- [114] Michael G Rabbat, Robert D Nowak, and James A Bucklew. Generalized consensus computation in networked systems with erasure links. In *Signal Processing Advances in Wireless Communications, 2005 IEEE 6th Workshop on*, pages 1088–1092. IEEE, 2005.
- [115] S Sundhar Ram, A Nedić, and VV Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.
- [116] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.
- [117] Yousef Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation*, 37(155):105–126, 1981.
- [118] A Sahai and D Cabric. Spectrum sensing: fundamental limits and practical challenges. In *Proc. IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, 2005.
- [119] Venkatesh Saligrama and David A Castanon. Reliable distributed estimation with intermittent communications. In *IEEE Conference on Decision and Control*, pages 6763–6768. IEEE, 2006.
- [120] A. H. Sayed. *Fundamentals of Adaptive Filtering*. John Wiley & Sons, New Jersey, 2003.
- [121] A.H. Sayed and C.G. Lopes. Adaptive processing over distributed networks. *IEICE Trans. on Fund. of Electronics, Communications and Computer Sciences*, 90(8):1504–1510, 2007.
- [122] Ali H Sayed. Diffusion adaptation over networks. *arXiv preprint arXiv:1205.4220*, 2012.
- [123] Ioannis D Schizas, Gonzalo Mateos, and Georgios B Giannakis. Distributed lms for consensus-based in-network adaptive processing. *IEEE Transactions on Signal Processing*, 57(6):2365–2382, 2009.
- [124] K. Slavakis, S. Theodoridis, and I. Yamada. Online kernel-based classification using adaptive projection algorithms. *IEEE Transactions on Signal Processing*, 56(7 Part 1):2781–2796, 2008.
- [125] K. Slavakis, I. Yamada, and N. Ogura. The adaptive projected subgradient method over the fixed point set of strongly attracting nonexpansive mappings. *Numerical Functional Analysis and Optimization*, 27, 7(8):905–930, 2006.
- [126] Konstantinos Slavakis, Pantelis Bouboulis, and Sergios Theodoridis. Adaptive multiregression in reproducing kernel hilbert spaces: The multiaccess MIMO channel case. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):260–276, 2012.
- [127] Konstantinos Slavakis, Sergios Theodoridis, and Isao Yamada. Adaptive constrained learning in reproducing kernel hilbert spaces: the robust beamforming case. *IEEE Transactions on Signal Processing*, 57(12):4744–4764, 2009.

- [128] Konstantinos Slavakis and Isao Yamada. The adaptive projected subgradient method constrained by families of quasi-nonexpansive mappings and its application to online learning. *SIAM Journal on Optimization*, 23(1):126–152, 2013.
- [129] H. Stark, Y. Yang, and Y. Yang. *Vector space projections: A numerical approach to signal and image processing, neural nets, and optics*. John Wiley & Sons, Inc. New York, NY, USA, 1998.
- [130] G. Strang. *Introduction to linear algebra*. Wellesley Cambridge Press, 2003.
- [131] N. Takahashi and I. Yamada. Steady-state mean-square performance analysis of a relaxed set-membership nlms algorithm by the energy conservation argument. *Signal Processing, IEEE Transactions on*, 57(9):3361–3372, 2009.
- [132] Noriyuki Takahashi, Isao Yamada, and Ali H Sayed. Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis. *IEEE Transactions on Signal Processing*, 58(9):4795–4810, 2010.
- [133] S. Theodoridis and K. Koutroumbas. *Pattern recognition*. Academic Press, 4th ed. edition, 2009.
- [134] Sergios Theodoridis, Konstantinos Slavakis, and Isao Yamada. Adaptive learning in a world of projections. *IEEE Signal Processing Magazine*, 28(1):97–123, 2011.
- [135] John Nikolas Tsitsiklis. Problems in decentralized decision making and computation. Ph.d. dissertation.
- [136] S Tu and A Sayed. Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks. *IEEE Transactions on Signal Processing*, 60(12):6217–6234, 2012.
- [137] S Tu and A Sayed. Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks. 2012.
- [138] Sheng-Yuan Tu and Ali H Sayed. Foraging behavior of fish schools via diffusion adaptation. In *CIP*, pages 63–68. IEEE, 2010.
- [139] Sheng-Yuan Tu and Ali H Sayed. Mobile adaptive networks. *IEEE Journal of Selected Topics in Signal Processing*, 5(4):649–664, 2011.
- [140] Pramod K Varshney. Distributed detection theory and data fusion. Technical report, DTIC Document, 1995.
- [141] Ramanarayanan Viswanathan and Pramod K Varshney. Distributed detection with multiple sensors i. fundamentals. *Proceedings of the IEEE*, 85(1):54–63, 1997.
- [142] John Von Neumann. *Functional operators: The geometry of orthogonal spaces*. Princeton University Press, 1950.
- [143] Bernard Widrow and Marcia Hoff. Adaptive switching circuits. 1960.
- [144] Chai Wah Wu. Synchronization and convergence of linear dynamics in random directed networks. *IEEE Transactions on Automatic Control*, 51(7):1207–1210, 2006.

## BIBLIOGRAPHY

---

- [145] Jin-Jun Xiao, A. Ribeiro, Zhi-Quan Luo, and G.B. Giannakis. Distributed compression-estimation using wireless sensor networks. *IEEE Signal Processing Magazine*, 23(4):27 – 41, july 2006.
- [146] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.
- [147] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.
- [148] I. Yamada and N. Ogura. Adaptive projected subgradient method for asymptotic minimization of sequence of nonnegative convex functions. *Numerical functional analysis and optimization*, 25(7&8):593–617, 2004.
- [149] I. Yamada and N. Ogura. Adaptive projected subgradient method for asymptotic minimization of sequence of nonnegative convex functions. *Numerical functional analysis and optimization*, 25(7&8):593–617, 2004.
- [150] I. Yamada, K. Slavakis, and K. Yamada. An efficient robust adaptive filtering scheme based on parallel subgradient projection techniques. 6:3725–3728, 2001.
- [151] M. Yukawa, K. Slavakis, and I. Yamada. Adaptive parallel quadratic-metric projection algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1665 –1680, 2007.
- [152] M. Yukawa and I. Yamada. Set-theoretic adaptive filtering based on data-driven sparsification. *International Journal of Adaptive Control and Signal Processing*, 25:707–722.
- [153] M. Yukawa and I. Yamada. A unified view of adaptive variable-metric projection algorithms. *EURASIP Journal on Advances in Signal Processing*, 2009:2–2, 2009.
- [154] Masahiro Yukawa. Krylov-proportionate adaptive filtering techniques not limited to sparse systems. *IEEE Transactions on Signal Processing*, 57(3):927–943, 2009.
- [155] Masahiro Yukawa, Rodrigo C de Lamare, and Isao Yamada. Robust reduced-rank adaptive algorithm based on parallel subgradient projection and krylov subspace. *IEEE Transactions on Signal Processing*, 57(12):4660–4674, 2009.