Tamara Guerra Miller

Distributed Sparsity-Aware Signal Processing Algorithms for Sensor Networks

Dissertação de mestrado

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Programa de Pós–graduação em Engenharia Elétrica

Rio de Janeiro
March 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Tamara Guerra Miller

Distributed Sparsity-Aware Signal Processing Algorithms for Sensor Networks

Dissertação de mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétrica of the Departamento Engenharia Elétrica, PUC-Rio as partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

Advisor: Prof. Rodrigo C. de Lamare

Rio de Janeiro
March 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Tamara Guerra Miller

Distributed Sparsity-Aware Signal Processing Algorithms for Sensor Networks

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétrica of the Departamento Engenharia Elétrica, PUC-Rio as partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

Prof. Rodrigo C. de Lamare
Advisor
Centro de Estudos em Telecomunicações — PUC-Rio

Prof. Cássio Guimarães Lopes
USP

Prof. Raimundo raimundo Sampaio Neto
Centro de estudos em Telecomunicações — PUC-Rio

Dr. César Augusto Medina
Centro de estudos em Telecomunicações — PUC-Rio

Prof. José Eugenio Leal
Coordinator of the Centro Técnico
Científico da PUC-Rio

Rio de Janeiro, March 11th, 2016

Tamara Guerra Miller

The author graduated in Telecommunication and Electronics Engineering from the José Antonio Echeverría Polytechnic Institute, in Havana, Cuba, 2010.

Bibliographic data

Miller, Tamara Guerra

Distributed Sparsity-Aware Signal Processing Algorithms for Sensor Networks/ Tamara Guerra Miller; advisor: Prof. Rodrigo C. de Lamare . — Rio de Janeiro : PUC-Rio, Departamento de Engenharia Elétrica, 2016.

v., 84 f: il.(color.) ; 29,7 cm

1. Dissertação de mestrado - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica.

Inclui referências bibliográficas.

I. C. de Lamare, Rodrigo . II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Acknowledgments

To all the professors that have contributed to my academic and professional career until today, especially to the professors of the PUC-Rio.

To my advisor Rodrigo Caiado de Lamare for his support during the master's studies.

To the CAPES, CNPq and the PUC-Rio, for the financial support.

To my family, specially my mother, my father, my brother and my grandparents for always being there for me.

To Sebastián, for his unconditional support these years.

To my colleagues and friends of CETUC for the support and friendship.

Abstract

Miller, Tamara Guerra; C. de Lamare, Rodrigo . **Distributed Sparsity-Aware Signal Processing Algorithms for Sensor Networks**. Rio de Janeiro, 2016. 84p. Msc. Dissertation — Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation proposes distributed adaptive algorithms exploiting sparsity for parameter and spectrum estimation over sensor networks. Conventional and modified conjugate gradient (CG and MCG) algorithms using consensus and diffusion strategies are presented. Sparsity-aware versions of CG an MCG algorithms using l_1 and log-sum penalty functions are developed. The proposed sparsity-aware and non-sparse CG and MCG methods outperform the equivalent variants of the least-mean square (LMS) algorithms in terms of convergence rate and mean square deviation (MSD) at steady state, and have a close performance to the recursive least square (RLS) algorithm. The diffusion CG strategies have shown the best performance, specifically the adapt then combine (ATC) version. Furthermore a distributed alternating mixed discrete-continuous (DAMDC) algorithm to approach the oracle algorithm based on the diffusion strategy for parameter and spectrum estimation over sensor networks is proposed. An LMS type algorithm with the DAMDC proposed technique obtains the oracle matrix in an adaptive way and compare it with the existing sparsity-aware as well as the classical algorithms. The proposed algorithm has an improved performance in terms of MSD. Numerical results show that the AMDC-LMS algorithm is reliable and can be applied in several scenarios.

Keywords

distributed signal processing; sensor networks; adaptive algorithms; sparsity-aware algorithms.

Resumo

Miller, Tamara Guerra; C. de Lamare, Rodrigo . **Algoritmos Adaptativos com Exploração de Esparsidade em Redes de Sensores Distribuídas**. Rio de Janeiro, 2016. 84p. Dissertação de Mestrado — Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Neste trabalho de dissertação são propostos algoritmos adaptativos que exploram a esparsidade em redes distrbuídas de sensores para estimacion de parâmetros e estimação espectral. São desenvolvidos algoritmos gradiente conjugado (CG) distribuído para os protocolos consenso e difusão em versão convencional e modificada (MCG). Esses algoritmos são desenvolvidos com exploração de esparsidade usando as funções penalidades l_1 e log-sum. Os métodos propostos apresentam um melhor desempenho en termos de velocidade de convergência e desvio médio quadratico (MSD) que as já conhecidas variantes distribuídas do algoritmo least mean square (LMS) e muito próximo ao desempenho do algoritmo recursive least square (RLS). Além disso, propõe-se um algoritmo distribuído de optimização alternada de variáveis discretas e contínuas (DADMDC) baseado no LMS, com o objetivo de identificar as posições não nulas do vetor de parâmetros assim como faz o algoritmo oráculo. O algoritmo DAMDC-LMS apresenta um desempenho muito próximo ao algoritmo oráculo e tem maior velocidade de convergência que os algoritmos estudados com exploração de esparsidade. Os resultados numéricos mostram que o algoritmo DAMDC-LMS pode ser aplicado em vários cenários.

Palavras–chave

processamento distribuído de sinais; redes de sensores; algoritmos adaptativos; algoritmos com exploração de esparsidade.

Sumário

Figure list

Table list

1
Introduction

1.1
Overview

For several years, sensor networks have been applied in medicine, industry, agriculture and other areas [1]-[9],[11]-[18]. Distributed signal processing has become a very common and useful approach to extract information in a network by performing estimation of the desired parameters, spectrum estimation and other applications. The efficiency of the network depends on the communication protocol used to exchange information between the nodes, as well as the algorithm to obtain the parameters. Another important aspect is to prevent a failure in any agent that may affect the operation and the performance of the network. Similar to a single node adaptive processing, the performance of the network may vary in time. Distributed schemes can offer better estimation performance as compared with the centralized approach, based on the principle that each node communicates with the other nodes and exploits the spatial diversity in the network [1].

The main strategies for communication and exchange of information in distributed processing are incremental, consensus and diffusion protocols. Recent studies show that diffusion strategies outperform the other ones [1], [6]. Each strategy will be introduced and discussed in the next chapter.

Distributed learning procedures also offer an attractive approach to dealing with large data sets [1]-[9], but still face many challenges in this field. It is necessary to ensure a strongly link connection and large communication bandwidth between nodes to start and maintain the network operation to estimate parameters and transmit their local data. In many scenarios, the impulse responses of unknown systems can be assumed to be sparse, containing only a few large coefficients interspersed among many negligible ones [9]. Other important challenges in distributed processing are the reduction of the computational cost, and to achieve a faster convergence rate than the centralized methods and a lower error value at steady state.

1.2 Motivation

Most of the studies developed for distributed processing in general and particulary exploiting sparsity have focused on the least-mean square (LMS) and recursive least-squares (RLS) algorithms using different penalty functions [9]-[12]. These penalty functions perform a regularization that attracts to zero the coefficients of the parameter vector that are not associated with the weights of interest. The most well-known and exploited penalty functions are the l_0-norm, the l_1-norm and the log-sum [10]. With these techniques a better network performance is achieved, in the presence of sparsity in the set of parameters.

The Conjugate Gradient (CG) algorithm [14] has been studied and developed for adaptive centralized and distributed processing, mostly using the diffusion strategy [15], which often results in algorithms that are more computationally complex than consensus techniques. The faster convergence performance of CG algorithms over the LMS algorithm and its lower computational complexity and better numerical stability than the RLS algorithm makes it suitable for this task. However, prior work on distributed CG techniques is rather limited as a consensus-type algorithm and techniques that exploit possible sparsity of the signals have not been developed so far. For those reasons it is proposed in this work distributed CG algorithms for parameter estimation over sensor networks

The optimal algorithm for processing sparse signals and systems is known as the oracle algorithm. It can identify the positions of the non-zero coefficients and fully exploit the sparsity of the system under consideration [16].Prior work on distributed oracle method is rather limited and techniques that exploit possible sparsity of the signals using discrete and continuous variables have not been developed so far. In this dissertation we also propose a distributed AMDC-LMS algorithm based on alternating and mixed optimization of continuous and discrete values. Specifically, we develop distributed LMS algorithm using the diffusion protocol for distributed parameter and spectrum estimation.

1.3 Contributions

The contributions can be summarized as fallows:

. Two CG-based consensus distributed solutions and two diffusion distributed CG-based strategies are proposed for parameter estimation. In

detail, the consensus distributed conventional CG solution (Consensus-CG), consensus distributed modified CG solution (Consensus-MCG), diffusion distributed conventional CG solution (DDCG) and diffusion distributed modified CG solution (DDMCG) are developed and analysed in terms of their computational complexity. The proposed CG algorithms present a faster convergence rate than the LMS-type algorithms and a lower computational complexity than RLS-type techniques for the same communication protocols. These algorithms can be used in different applications such as environmental monitoring and medical parameters identification.

- Four CG-based consensus distributed solutions and four diffusion distributed CG-based strategies, all of them exploiting sparsity for parameter estimation. In detail, the sparsity-aware consensus distributed CCG and MCG solutions, and the sparsity-aware diffusion distributed CCG and MCG solutions using for each one the l_1 and $log-sum$ penalty functions. The proposed sparsity-aware CG methods have an improved performance in terms of mean square deviation (MSD)and convergence rate compared with the same versions of the least-mean square (LMS) algorithm and a close performance to the distributed recursive least squares (RLS) algorithm.
- A distributed adaptive mixed discrete continuous (DAMDC) algorithm for parameter estimation. Specifically, we developed the (DAMDC-LMS) algorithm using the diffusion protocol. Based on the faster convergence of the Oracle-LMS algorithm as compared with conventional and sparsity-aware versions, the DAMDC-LMS algorithm is proposed in order to obtain a very close performance in terms of convergence and minimum square deviation (MSD) to the oracle method. The application scenarios of this method are parameter estimation and spectrum estimation over sensor networks.

1.4 Dissertation Outline

This dissertation is organized as follows:

- Chapter 2 presents an overview of the theory related to this work and introduces the algorithms and strategies that are used in this dissertation. The topics of distributed signal processing, adaptive algorithms, distributed strategies and sparsity-aware techniques are covered along with a

review of previous work in these topics and important applications such as parameter estimation and spectrum estimation.

- Chapter 3 describes the system model and problem statement of adaptive algorithms for distributed networks based on CG strategies. Specifically, consensus and diffusion adaptive solutions are proposed. The computational complexity as well as convergence rate and MSD value are analysed for the distributed estimation scenario.
- In Chapter 4, adaptive CG algorithms for distributed estimation exploiting sparsity are proposed. The analysis of the proposed l_1 and log-sum CG and MCG algorithms are presented in terms of their stability, steady-state, tracking performance and computational complexity.
- Chapter 5 presents a novel distributed LMS scheme based on alternating mixed discrete-continuous adaptation (ADMC-LMS). This algorithm is developed for conventional and sparse data sets for parameter and spectrum estimation scenarios. This proposal is compared with recently reported algorithms in the literature such as the oracle and sparsity-aware techniques in terms of MSD and tracking capability.
- Chapter 6 presents the conclusions of this work, and discusses future directions that could be carried out in terms of research.

1.5 Notation

$\boldsymbol{a}$	the vector (boldface lower case letters).
$\boldsymbol{A}$	the matrix (boldface upper case letters).
$\Re$	real part.
$\boldsymbol{I}_N$	$N \times N$ identity matrix.
$(.)^*$	complex conjugate.
$(.)^T$	matrix transpose.
$(.)^H$	Hermitian transpose.
$E[.]$	expectation operator.

2
Literature Review

In this chapter we review prior art on distributed signal processing. The network connection models, the main protocols to exchange information between nodes and some adaptive algorithms usually used for distributed processing are introduced and analysed. Fundamental sparsity-aware techniques are discussed as important methods for parameter estimation of sparse systems covered in our research.

2.1
Distributed Signal Processing

With the aim of exploiting the potential of sensor networks, it is essential to develop energy and bandwidth efficient signal processing algorithms that can be implemented in a fully distributed manner. Distributed signal processing requires judicious coordination and planning as well as careful exploitation of the limited communication capability of each individual sensor [2]. It aims to provide a superior adaptation performance to non-cooperative centralized signal processing, and is able to solve optimization and adaptation problems [18].

In the non-cooperative mode of operation, agents act independently of each other. In the centralized mode of operation, agents transmit their data to a fusion center, which is capable of processing the data centrally. The fusion center then shares the results of the analysis back with the distributed agents. The centralized solutions can be powerful, but they demand high levels of computational cost and routing resources to perform the exchange of information to and from the fusion center. Another important weakness of centralized networks is the fact that if the fusion center fails the entire network collapses [1].

In contrast, in the distributed mode of operation, agents are connected by a topology and they are allowed to share information only with their immediate neighbors. This kind of cooperation overcomes the aforementioned disadvantages associated with centralized processing. Recent studies have

shown the improved adaptation performance of distributed techniques as compared with non-cooperatives agents and that they also offer an attractive approach to dealing with such large data sets and cloud computing [1]-[18].

2.1.1 Distributed Sensor Networks

With the development of sensor technology and wireless networks, the use of sensor networks has been a good solution for many applications such as environmental monitoring, medicine and surveillance. The design of a sensor network employs a geographical area where the nodes are spatially spread to obtain and exchange information for a specific application.

Some factors to be taken into account for the design of a sensor network are the fault tolerance, scalability, network topology, hardware constraints, energy consumption, the application environment and production costs [2]. The integration of these aspects is very important to decide the communication protocol and the algorithm to be implemented, and how to obtain the desired information of the network.

For distributed processing over sensor networks several algorithms have been used to process and obtain the information required by the desired application. Some of the algorithms developed are the LMS [1]-[11], RLS [12], [21] and CG [14], [15], [18]. These algorithms as well as some applications are presented in the following sections.

2.1.2 Applications

Distributed sensor networks are of interest to the most diverse fields. Some important applications are defense, education and agriculture. This section is focussed on distributed parameter estimation and distributed spectrum estimation applications, that are the applications considered in this work.

Distributed Estimation

A network consisting of N nodes distributed over a spatial domain has a neighborhood for node k that is denoted by N_k. Figure 2.1 shows a network with six nodes and the neighborhood of node $k = 6$ given by $N_6 = 2, 3, 6$.

The main task in distributed estimation is to estimate the unknown parameter vector $\boldsymbol{\omega}_0$ of dimension $M \times 1$, where M is the number of filter

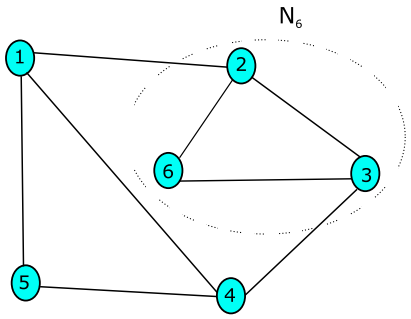

Figure 2.1 – Network model with 6 nodes and N_6 neighborhood representation.

taps [1]. At each time i each node k can take measurements $\{d_{k,i}, \boldsymbol{x}_{k,i}\}$ of random scalar $d_{k,i}$ and the $M \times 1$ input signal vector $\boldsymbol{x}_{k,i}$. The random processes $\{d_{k,i}, \boldsymbol{x}_{k,i}\}$ are related to $\boldsymbol{\omega}_0$ through the linear regression model given by

$$d_{k,i} = \boldsymbol{\omega}_0^H \boldsymbol{x}_{k,i} + n_{k,i}, \tag{2-1}$$

where $n_{k,i}$ is the measurement noise with zero mean and variance $\sigma_{n,k}^2$. To obtain the MSE optimal solution at each node k we need to minimize the cost function [1] given by

$$C(\boldsymbol{\omega}_{k,i}) = \sum_{k=1}^{N} E[|d_{k,i} - \boldsymbol{\omega}_{k,i}^H \boldsymbol{x}_{k,i}|^2], \tag{2-2}$$

where E denotes expectation and $\boldsymbol{\omega}_{k,i}$ is the estimated vector generated by node k at time instant i.

The minimization of the cost function (2-2) have been the goal of many works to estimate the optimal parameter vector $\boldsymbol{\omega}$. With this purpose some distributed adaptive algorithms such as LMS [6] [11] [29], RLS [21] [36] and CG[15] have been proposed to achieve this goal.

Distributed Spectrum Estimation

Spectral estimation has become a very important task due to the potential of a better use of the frequency spectrum. In distributed processing the goal in this application is to estimate over a sensor network with N nodes the spectrum of a transmitted signal $\boldsymbol{s}$ [17] [31] [42]. The power spectral density (PSD) of the signal $\boldsymbol{s}$ at each frequency denoted by $\boldsymbol{\Phi}_s(f)$ is given by

$$\boldsymbol{\Phi}_s(f) = \sum_{m=1}^{B} b_m(f)\omega_{0m} = \boldsymbol{b}_0^T(f)\boldsymbol{\omega}_0, \tag{2-3}$$

where $\boldsymbol{b}_0(f) = [b_1(f), ..., b_B(f)]^T$ is the vector of basis functions evalu-

ated at frequency f, $\boldsymbol{\omega}_0 = [\omega_{01}, ..., \omega_{0B}]$ is a vector of weighting coefficients representing the power that transmits the signal $\boldsymbol{s}$ over each basis, and B is the number of basis functions. For B sufficiently large, the basis expansion in (2-3) can approximate the frequency spectrum. Possible choices for the set of basis functions $b_m(f)_{m=1}^B$ include: rectangular functions, raised cosines, Gaussian bells and splines [17], [18], [31].

The channel transfer function between a transmit node conveying the signal $\boldsymbol{s}$ and receive node k at time instant i is denoted by $\boldsymbol{H}_k(f,i)$, then the PSD of the received signal observed by node k can be expressed as

$$\begin{aligned}\boldsymbol{\Phi}_k(f,i) &= |H_k(f,i)|^2\boldsymbol{\Phi}_s(f) + \upsilon_{n,k}^2, \\ &= \textstyle\sum_{m=1}^{B} |H_k(f,i)|^2 b_m(f)\omega_{0m} + \upsilon_{n,k}^2, \\ &= \boldsymbol{b}_{k,i}^T(f)\boldsymbol{\omega}_{0m} + \upsilon_{n,k}^2.\end{aligned} \tag{2-4}$$

where $\boldsymbol{b}_{k,i}^T(f) = [|H_k(f,i)|^2 b_m(f)]_{m=1}^B$ and $\upsilon_{n,k}^2$ is the receiver noise power at node k.

With the distributed model as reference, at every time instant i each node k measures the PSD $\boldsymbol{\Phi}_k(f,i)$ presented in (2-4) over N_c frequency samples $f_j = f_{min} : (f_{max} - f_{min})/N_c : f_{max}$, for $j = 1, ..., N_c$, the desired signal is given by

$$d_{k,i}(j) = \boldsymbol{b}_{k,i}^T(f_j)\boldsymbol{\omega}_0 + \upsilon_{n,k}^2 + n_{k,i}(j). \tag{2-5}$$

where the last term denotes the observation Gaussian noise with zero mean and variance $\sigma_{n,j}^2$. The receiver noise power $\upsilon_{n,k}^2$ can be estimated with high accuracy in a preliminary step using, e.g., an energy estimator over an idle band, and then subtracted from (5-25) [18]. A linear model is obtained from the measurements over N_c contiguous channels:

$$\boldsymbol{d}_{k,i} = \boldsymbol{B}_{k,i}\boldsymbol{\omega}_0 + \boldsymbol{n}_{k,i}, \tag{2-6}$$

where $\boldsymbol{B}_{k,i} = [\boldsymbol{b}_{k,i}^T(f_j)]_{j=1}^{N_c} \in \Re^{N_c \times B}$, and $\boldsymbol{n}_{k,i}$ is a zero mean random vector. Then we can establish the cost function for each agent k

$$C(\boldsymbol{\omega}_{k,i}) = E[|\boldsymbol{d}_{k,i} - \boldsymbol{B}_{k,i}\boldsymbol{\omega}_0|^2], \tag{2-7}$$

The same analysis presented for distributed estimation is applied for spectrum estimation and the global cost function is given by

$$C(\boldsymbol{\omega}) \sum_{k=1}^{N} = E[|\boldsymbol{d}_{k,i} - \boldsymbol{B}_{k,i}\boldsymbol{\omega}_0|^2], \tag{2-8}$$

Distributed algorithms also have been proposed in several works to solve this global cost function with the aim of obtaining the estimated frequency spectrum. The most commonly used algorithm for this application is the least-mean square (LMS) algorithm [17] and [31].

2.2 Distributed Network Models

In distributed connected networks there is always one path connecting two nodes [1],[2] . The nodes my be connected directly by an edge if they are neighbors, or they may be connected by a path that passes through other intermediate nodes. Generally the networks by graphs, vertices and edges. In the following sections are presented some of the most known connected network models and topologies are represented.

2.2.1 Weakly-Connected Network

A connected network model can be considered as weakly connected when the paths with nonzero weight values can link two different nodes directly or indirectly in at least one direction. When this occurs the information can flow through the network at least in one direction between two agents [1]. Figure 2.2 below represents this kind of network.

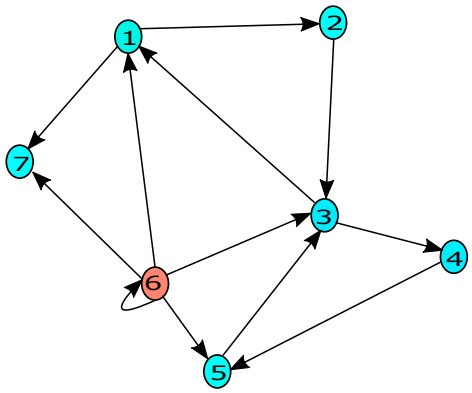

Figure 2.2 – Weakly-connected network.

It can be seen in Figure 2.2 that node 6 is incapable of receiving information from any other node. In contrast, node 6 can send information to the other nodes directly to its neighbors or through intermediate nodes.

2.2.2
Connected Network

The connected model classification is given when paths with nonzero weight values are linking any two nodes. The linking connection must be in both directions and can be directly or indirectly through another node [18].

In this model the information flows in both directions between any two different agents of the network. The forward and backward paths to the information flows can be the same or not.

2.2.3
Strongly-Connected network

A strongly-connected network is defined as a connected network previously defined and where all weight values of the self-loops of some agent are positives [1][41]. Figure 2.3 represents this kind of network.

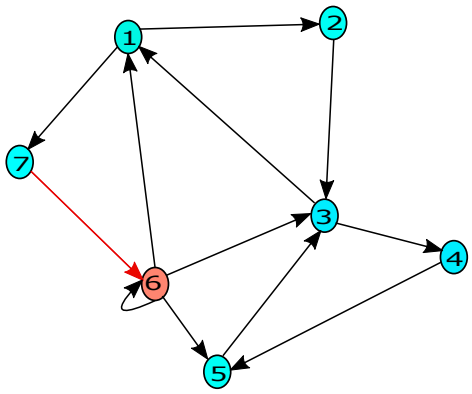

Figure 2.3 – Strongly-connected network.

For the information to flow between agents, it is not sufficient for paths to exist linking these agents. It is also necessary that the connection remains active due to zero scaling weight. The arrow represented in red for emphasis in Figure 2.3 shows de difference between weakly and strongly connected networks. The reversal connection between nodes 6 and 7 in Figure 2.3 as compared with Figure 2.2 illustrates how node 6 can receive information from any other node in the network directly or indirectly, and also can reach all other agents in the same way. Our work is focused on strongly-connected networks.

2.3 Distributed Strategies for Signal Processing

In distributed networks, besides the connection models there are several connection topologies between nodes, as well as exchange protocols that regulate the communication flow. According to the connection topology distributed networks can be fully or partially connected networks.

A fully or totally connected network is a network in which each of the nodes is connected to every other node. In this kind of topology the number of connections increase with the number of nodes [18] [41]. This feature can represent a disadvantage in terms of resources and bandwidth. Figure 2.4 shows and scheme of a fully connected network.

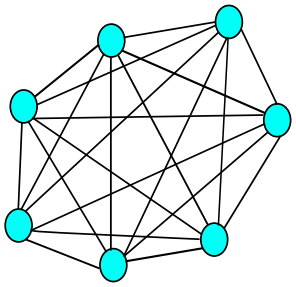

Figure 2.4 – Fully-connected network topology.

In partially connected networks the nodes only can exchange information with neighbors determined by the connectivity topology [18] [41]. This topology shows benefits in terms of physical connections as compared with fully connected networks. Our research is focused on partially connected networks.

Based on the communication strategy to exchange information between nodes there are three protocols on distributed networks, incremental, consensus and diffusion [1]. In the following sections we introduce and analyze these strategies.

2.3.1 Incremental Strategy

The incremental protocol is a strategy where the communication flows cyclically and the information is exchanged sequentially from one node to another adjacent node. For this strategy the flow of information must be preset at the beginning of the process, this means that the path to exchange information between nodes must be defined before each agent starts to share and update its status [5]. Once the path is defined the process begins, each node

uses the information from its previous neighbor, updates its own information and sends to the next node, as depicted in Figure 2.5 .

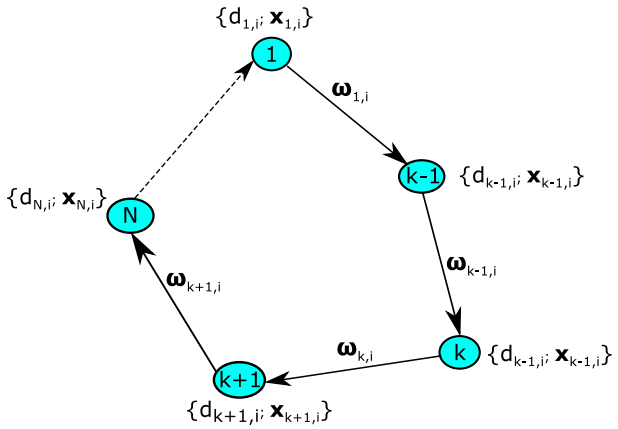

Figure 2.5 – Incremental strategy estimation.

In Figure 2.5 an example can be seen of an incremental protocol scheme with 5 nodes, where agent 1 is referring to the start of the cyclic path and agent N is associated to the end. The incremental calculations carried out by agent k are given by.

$$\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k-1,i} - \mu\widehat{\nabla_{\boldsymbol{\omega}^*} J_k}(\boldsymbol{\omega}_{k-1,i}) \tag{2-9}$$

Using the LMS algorithm the incremental update at each node is as follows:

$$\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k-1,i} + \mu \boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{x}_{k,i}^H \boldsymbol{\omega}_{k-1,i}]^* \tag{2-10}$$

The incremental solution is a simple cooperation strategy, but suffers some limitations. First, the incremental strategy is sensitive to agent or link failures. If an agent or a link on the cyclical path fails, then the flow of information through the network is interrupted. The cooperation between agents is limited due to each agent only receives the data from the previous agent and share its own only with the next one [26]. Another limitation is that for each iteration i, it is necessary to perform N incremental steps.

2.3.2 Consensus Strategy

In the consensus protocol each node has the ability to run its update simultaneously with the rest of the nodes. The parameter vector $\boldsymbol{\omega}_{k-1,i}$ on the left side of equation (2-10) that represents an incremental factor from the previous analyzed incremental strategy, is replaced by a combination of previous iterations of the neighboring nodes. The second term of the equation

is also replaced by the value of the parameter vector of previous iterations already available in agent k.The consensus strategy imposes a mathematical constraint so that all connected agents must converge to the same parameter vector [18] [32] [35]. A representation of a connected network that is used for the consensus strategy is illustrated in Figure 2.6.

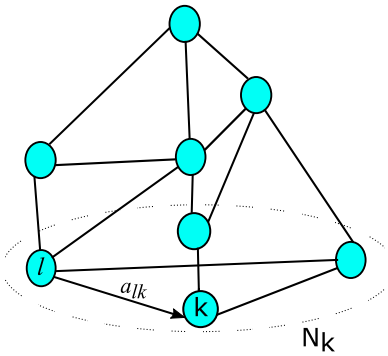

Figure 2.6 – Distributed network scheme.

The general recursion for distributed consensus strategy of the LMS algorithm is given by

$$\boldsymbol{\omega}_{k,i} = \sum_{l \in N_k} a_{lk}\boldsymbol{\omega}_{l,i-1} - \mu\widehat{\nabla_{\boldsymbol{\omega}^*} J_k}(\boldsymbol{\omega}_{k,i-1}), \tag{2-11}$$

where a_{lk} represents the combining coefficients of the data fusion which should comply with

$$a_{kl} \geq 0 \qquad \sum_{l \epsilon N_k} a_{kl} = 1,\ l \epsilon N_{k,i}, \forall k. \tag{2-12}$$

Equation (2-12) means that for every agent k, the sum of the weights a_{lk} that arrive at it from its neighbors is one. This value represents the weight that agent k assigns to $\boldsymbol{\omega}_{l,i-1}$ that receives from agent l. Matrix $\boldsymbol{A}$ is an $N \times N$ matrix that contains a_{lk} scalar values. The sum of these scalar values at each column is equal to one, which means that A is a left-stochastic matrix [1].

There are different rules to calculate the a_{lk} coefficients such as the Metropolis and the Laplacian rules [7]. In this work we adopted the Metropolis rule to compute the a_{lk} values, as given by

$$a_{lk} = \begin{cases} \frac{1}{max\{|\mathcal{N}_k|,|\mathcal{N}_l|\}} & \text{if } k \neq l \text{ are linked,} \\ 1 - \sum\limits_{l \in \mathcal{N}_k/k} a_{lk}, & \text{for } k = l. \end{cases} \tag{2-13}$$

The consensus protocol as compared with the incremental strategy does not need to define the cycle and re-number the nodes for the information to

flow, which is translated in a more efficient protocol.

2.3.3 Diffusion Strategy

In the diffusion mechanism, each node communicates with the rest of the nodes of the network. The scheme of the connection topology, as well as the consensus strategy is the same represented in Figure 2.6. There are two kinds of diffusion protocols in distributed networks, the Combine-then-Adapt (CTA) strategy and the Adapt-then-Combine (ATC) strategy [6].

Combine then Adapt (CTA)

The CTA strategy first involves a combination step, where it is first evaluated into an intermediate state variable the influence of the neighborhood data from the previous iteration to update its information. This intermediate variable is then used to perform the weight update [26]. The general recursion for this diffusion strategy variant using the LMS algorithm is described as follows:

$$CTA \begin{cases} \boldsymbol{\varphi}_{k,i-1} = \sum_{l \in N_k} a_{lk} \boldsymbol{\omega}_{l,i-1}, \quad Combination, \\ \boldsymbol{\omega}_{k,i} = \boldsymbol{\varphi}_{k,i-1} - \mu \widehat{\nabla_{\boldsymbol{\omega}^*} J_k}(\boldsymbol{\varphi}_{k,i-1}), \quad Adaptation, \end{cases} \tag{2-14}$$

where a_{lk} is the combiner coefficient obtained with the Metropolis rule using equation (2-13) previously discussed in the distributed consensus strategy.

At each time instant i all the nodes of the network are performing the combination and adaptation steps simultaneously. Using the LMS algorithm this strategy has the same computational complexity as the consensus protocol [1].

Adapt then Combine (ATC)

The ATC protocol is obtained by switching the order of the the CTA variant. The adaptation step is performed first, and the parameter vector is obtained from the previous value $\boldsymbol{\omega}_{k,i-1}$ at each node. The second step performs a combination of the previous adaptation step and the combiner coefficients of the neighbors of each agent [26]. This strategy has been implemented using

different adaptive algorithms such as RLS, affine projection (AP) and LMS [34] [27] [26]. The recursion for the LMS algorithm at each node is given by

$$ATC \begin{cases} \boldsymbol{\varphi}_{k,i} = \boldsymbol{\omega}_{k,i-1} - \mu\widehat{\nabla_{\boldsymbol{\omega}^*} J_k}(\boldsymbol{\omega}_{k,i-1}), & \textit{Adaptation}, \\ \boldsymbol{\omega}_{k,i} = \sum_{l \in N_k} a_{lk}\boldsymbol{\varphi}_{l,i}, & \textit{Combination}. \end{cases} \tag{2-15}$$

The coefficients of the combiner are also calculated with the Metropolis rule of equation (2-13).

The ATC diffusion strategy incorporates the influence of the data from the neighborhood of node k into the update of the parameter vector, which influences the performance of the algorithms. Using the LMS algorithm it has been demonstrated that the diffusion strategies in general outperform the consensus protocol [26]. Specifically the ATC mechanism converges faster and obtains lower MSD value at steady state than CTA and consensus strategies [6].

2.4 Adaptive Algorithms

Adaptive algorithms have been used for centralized and distributed signal processing applications that require learning and tracking abilities to process signals and extract information. Adaptive algorithms have the ability to learn by observing the environment and are often guided by reference signals [18]. In distributed environments each node operates as an adaptive filter that models the relationship between the input and the output signals, adjusting the parameters according to the algorithm and the application field. The adaptive algorithm and the cooperation strategy define the efficiency of the network to update and obtain the desired parameters.

There are several adaptive algorithms, the most widely used for adaptive signal processing is the least mean square (LMS) algorithm [18]. The recursive least square (RLS) algorithm is another well-known method in adaptive filtering that provides better convergence rate as compared with the LMS, but it is more complex in terms of computational operations [43]. Another algorithm that has been used for adaptive filtering is the conjugate gradient (CG) algorithm that presents a better performance than the LMS in terms of convergence rate and a lower computational complexity than the RLS, and also reaches a better stability [8]. These three algorithms are described in the following subsections.

2.4.1 The Least-Mean Square (LMS) Algorithm

The LMS algorithm is a stochastic-gradient algorithm that aims to minimize the following MSE cost function

$$C_{LMS}(\boldsymbol{\omega}_i) = E[|d_i - \boldsymbol{\omega}_i^H \boldsymbol{x}_i|^2], \tag{2-16}$$

where d_i is the desired signal, $\boldsymbol{x}_i$ is the input signal vector and the filter weight vector is represented by $\boldsymbol{\omega}_i$.The algorithm has useful tracking abilities and has been used for many applications such as parameter and spectrum estimation [1] and [17].

The optimum solution to be solved, known as the Winier filter, is described as

$$\boldsymbol{\omega}_0 = \boldsymbol{R}_x^{-1} \boldsymbol{b}_x. \tag{2-17}$$

where $\boldsymbol{R}_x$ is the correlation matrix of the input signal $\boldsymbol{x}$ and $\boldsymbol{b}_x$ is the cross-correlation vector between the desired signal and the input signal, and are calculated as $E[\boldsymbol{x}_i \boldsymbol{x}_i^H]$ and $E[d_i^* \boldsymbol{x}_i]$, respectively. These statistical variables must be estimated [3]-[4]. The LMS algorithm adopts the simplest instantaneous estimates of $\boldsymbol{R}_x$ and $\boldsymbol{b}_x$, given by

$$\boldsymbol{R}_x = \boldsymbol{x}_i \boldsymbol{x}_i^H, \tag{2-18}$$

$$\boldsymbol{b}_x = d_i^* \boldsymbol{x}_i. \tag{2-19}$$

By substituting equations (2-44) and (2-45) in the gradient vector of the cost function we get

$$\begin{aligned} \boldsymbol{\omega}_i &= \boldsymbol{\omega}_{i-1} - \mu \nabla_{\boldsymbol{\omega}^*} C(\boldsymbol{\omega}_i), \\ &= \boldsymbol{\omega}_{i-1} + \mu \boldsymbol{x}_i [d_i^* - \boldsymbol{x}_i^H \boldsymbol{\omega}_{i-1}], \end{aligned} \tag{2-20}$$

where μ is the LMS step size. This parameter is related to the convergence rate and the stability of the method, and the convergence is achieved when

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{2-21}$$

where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{R}_x$.

2.4.2 The Recursive Least Squares (RLS) Algorithm

The RLS algorithm estimates the weights of the filter using the criterion of least squares (LS). The cost function to be minimized is the quadratic error described as

$$C_{RLS}(\boldsymbol{\omega}_i) = \sum_{l=0}^{i} \lambda^{i-l} |d_i - \boldsymbol{\omega}_i^H \boldsymbol{x}_i|^2. \tag{2-22}$$

where λ is the forgetting factor that must be $0 << \lambda < 1$. Usually the value of the forgetting is very close to one [3]. The derivation process of the cost function gives as result

$$\frac{\partial C_{RLS}(\boldsymbol{\omega}_i)}{\partial \boldsymbol{\omega}_i^*} = \sum_{l=0}^{i} \lambda^{i-l} [\boldsymbol{x}_i \boldsymbol{x}_i^H \boldsymbol{\omega}_i - \boldsymbol{x}_i d_i]^* = 0. \tag{2-23}$$

Separating the terms of equation (2-23) two terms can be defined as follows:

$$\boldsymbol{\Phi}_i = \sum_{l=0}^{i} \lambda^{i-l} \boldsymbol{x}_i \boldsymbol{x}_i^H, \tag{2-24}$$

$$\boldsymbol{\Theta}_i = \sum_{l=0}^{i} \lambda^{i-l} \boldsymbol{x}_i d_i^*. \tag{2-25}$$

Then equation (2-23) can be stated as

$$\boldsymbol{\omega}_i = \boldsymbol{\Phi}_i^{-1} \boldsymbol{\Theta}_i. \tag{2-26}$$

The recursive expressions for Φ_i and Θ_i are given by

$$\boldsymbol{\Phi}_i = \lambda \boldsymbol{\Phi}_{i-1} + \boldsymbol{x}_i \boldsymbol{x}_i^H, \tag{2-27}$$

$$\boldsymbol{\Theta}_i = \lambda \boldsymbol{\Theta}_{i-1} + \boldsymbol{x}_i d_i^*. \tag{2-28}$$

As can be seen in (2-26) it is necessary to use the matrix inversion lemma for Φ_i as follows:

$$\boldsymbol{\Phi}_i^{-1} = \lambda^{-1} \boldsymbol{\Phi}_{i-1}^{-1} - \frac{\lambda - 2\boldsymbol{\Phi}_{i-1}^{-1} \boldsymbol{x}_i \boldsymbol{x}_i^H \boldsymbol{\Phi}_{i-1}^{-1}}{1 + \lambda^{-1} \boldsymbol{x}_i^H \boldsymbol{\Phi}_{i-1}^{-1} \boldsymbol{x}_i}. \tag{2-29}$$

From now on $\boldsymbol{\Phi}_i^{-1}$ is represented by $\boldsymbol{P}_i$ as follows

$$\boldsymbol{P}_i = \boldsymbol{\Phi}_i^{-1}, \tag{2-30}$$

and we express the Kalman gain vector $\boldsymbol{k}_i$ [3], given by

$$\boldsymbol{k}_i = \frac{\lambda^{-1}\boldsymbol{P}_{i-1}\boldsymbol{x}_i}{1+\lambda^{-1}\boldsymbol{x}_i^H\boldsymbol{P}_{i-1}\boldsymbol{x}_i}. \tag{2-31}$$

Employing the recursion presented in (2-32) and the Kalman gain vector equation in (2-31) we obtain

$$\boldsymbol{P}_i = \lambda^{-1}\boldsymbol{P}_{i-1} - \lambda^{-1}\boldsymbol{k}_i\boldsymbol{x}_i^H\boldsymbol{P}_{i-1}. \tag{2-32}$$

The Kalman gain vector can be reformulated as follows:

$$\begin{aligned}\boldsymbol{k}_i &= \lambda^{-1}\boldsymbol{P}_{i-1}\boldsymbol{x}_i - \lambda^{-1}\boldsymbol{k}_i\boldsymbol{x}_i^H\boldsymbol{P}_{i-1}\boldsymbol{x}_i,\\ &= (\lambda^{-1}\boldsymbol{P}_{i-1} - \lambda^{-1}\boldsymbol{k}_i\boldsymbol{x}_i^H\boldsymbol{P}_{i-1})\boldsymbol{x}_i,\\ &= \boldsymbol{P}_i\boldsymbol{x}_i.\end{aligned} \tag{2-33}$$

The general recursion for the RLS algorithm using equations (2-28) and (2-32) is given by

$$\begin{aligned}\boldsymbol{\omega}_i &= \boldsymbol{\omega}_{i-1} + \boldsymbol{P}_i\boldsymbol{\Theta}_i,\\ &= \boldsymbol{\omega}_{i-1} - \boldsymbol{k}_i\boldsymbol{x}_i^H\boldsymbol{\omega}_{i-1} + \boldsymbol{P}_i\boldsymbol{x}_i d_i^*,\\ &= \boldsymbol{\omega}_{i-1} + \boldsymbol{k}_i[d_i - \boldsymbol{\omega}_{i-1}^H\boldsymbol{x}_i]^*,\end{aligned} \tag{2-34}$$

where the term in brackets is the a priori estimation error.

2.4.3 The Conjugate Gradient (CG) Algorithm

The main task of the conjugate gradient (CG) algorithm in adaptive processing is to compute the following linear equation [8][14]

$$\boldsymbol{R}\boldsymbol{\omega} = \boldsymbol{b}, \tag{2-35}$$

where $\boldsymbol{R}$ and $\boldsymbol{b}$ as well as in previously discussed algorithms are the correlation matrix and the cross-correlation vector, respectively. The equation (2-35) is indirectly minimizing the global cost function of the CG method [33], given by

$$C_{CG}(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^H\boldsymbol{R}\boldsymbol{\omega} - \boldsymbol{b}^H\boldsymbol{\omega}. \tag{2-36}$$

To solve the cost function in (2-36) the CG algorithm initializes some parameters as follows:

$$\boldsymbol{g}_0 = \boldsymbol{b}, \tag{2-37}$$

$$\boldsymbol{p}_0 = \boldsymbol{g}_0, \tag{2-38}$$

where $\boldsymbol{g}$ is the residual of the gradient of the cost function, and $\boldsymbol{p}$ is the search direction vector.

At each iteration j of the CG algorithm the residual vector must be calculated in the negative direction of the gradient computing the Kyrlov subspace methods [23], described as

$$\boldsymbol{g}(j) = \boldsymbol{b} - \boldsymbol{R}\boldsymbol{\omega}(j) = -\nabla C_{CG}(\boldsymbol{\omega}), \tag{2-39}$$

and the direction vector is given by

$$\boldsymbol{p}(j) = \boldsymbol{g}(j) + \beta(j)\boldsymbol{p}(j), \tag{2-40}$$

where β is computed following the Gram-Schmidt orthogonalization procedure [23] described as

$$\beta(j) = \frac{\boldsymbol{g}^H(j)\boldsymbol{g}(j)}{\boldsymbol{g}(j-1)\boldsymbol{g}(j-1)}. \tag{2-41}$$

The update equation of the CG algorithm is obtained through different operations as follows

$$\boldsymbol{\omega}(j) = \boldsymbol{\omega}(j-1) - \alpha(j)\boldsymbol{p}(j), \tag{2-42}$$

where α is the step size that minimizes the CG cost function presented in (2-36), and is given by

$$\alpha(j) = \frac{\boldsymbol{g}^H(j-1)\boldsymbol{g}(j-1)}{\boldsymbol{p}^H(j)\boldsymbol{R}(j)\boldsymbol{p}(j)}. \tag{2-43}$$

There are different ways to estimate $\boldsymbol{R}$ and $\boldsymbol{b}$. In this work, we adopted the exponentially decaying data window, where these parameters are calculated as follows:

$$\boldsymbol{R}_i = \lambda\boldsymbol{R}_{i-1} + \boldsymbol{x}_i\boldsymbol{x}_i^H, \tag{2-44}$$

$$\boldsymbol{b}_i = \lambda\boldsymbol{b}_{i-1} + d_i^*\boldsymbol{x}_i. \tag{2-45}$$

The modified CG algorithms (MCG) for adaptive signal processing have been proposed in some works [8], [47] and [48]. The MCG algorithms have been implemented to improve the performance of the conventional CG algorithm from several applications leading to different results in terms of convergence rate and error computation.

2.5 Sparsity-Aware Techniques

In the last years with the increasing amount of data that needs to be processed a lot of research works have been focussed on sparse systems. Sparse signals are characterized by the presence of many zero or negligible coefficients [9]-[12]. In the presence of sparse signals, conventional adaptive algorithms may exhibit poor performance as compared with its typical operation with non-sparse systems. To solve this problem sparsity-aware techniques have been proposed in the literature incorporating in the cost function of the adaptive algorithm a penalty term to exploit sparsity.

Another method proposed for sparse signal processing is the oracle algorithm, which can identify the position of the non-zero coefficients. This algorithm is considered the optimal method for sparse systems [16].

The following subsections introduce and discuss three sparsity-aware techniques, the Zero-Attracting (ZA) strategy, the Reweighted Zero-Attracting (RZA) strategy and the l_0-norm strategy. The update equation of each one of these strategies is described as

$$\boldsymbol{\omega}_{new} = \boldsymbol{\omega}_{previous} + \text{gradient recursion} + \text{zero atraction}. \tag{2-46}$$

To finalize the literature review the oracle algorithm is presented along with its features.

2.5.1 The Zero-Attracting Strategy

The ZA technique incorporates into the cost function of the adaptive algorithm a regularization term with the l_1-norm. The goal of of this method is to attract the filter coefficients to zero using a scalar factor ρ. The new cost function for the ZA-LMS algorithm [10] is given by

$$C_{ZA\text{-}LMS}(\boldsymbol{\omega}_i) = E[|d_i - \boldsymbol{\omega}_i^H \boldsymbol{x}_i|^2] + \rho f_1(\boldsymbol{\omega}_i), \tag{2-47}$$

where f_1 represents the $\|\boldsymbol{\omega}_i\|_1$ penalty function.

The derivation process is the same as the standard LMS algorithm. The derivative of the last term in (2-47) with respect to $\boldsymbol{\omega}_i$ is obtained as follows:

$$\frac{\partial(f_1)}{\partial(\boldsymbol{\omega}_i^*)} = \text{sgn}(\boldsymbol{\omega}_i) = \begin{cases} \frac{[\boldsymbol{\omega}_i]_m}{|[\boldsymbol{\omega}_i]_m|}, & \text{if } [\boldsymbol{\omega}_i]_m \neq 0 \\ 0, & \text{if } \boldsymbol{\omega}_i = 0. \end{cases} \tag{2-48}$$

Using the equation (2-48), into the gradient updating of the LMS, we obtain the recursion of the ZA-LMS algorithm given by

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \mu \boldsymbol{x}_i[d_i^* - \boldsymbol{x}_i^H \boldsymbol{\omega}_{i-1}] - \rho \text{sgn}(\boldsymbol{\omega}_{i-1}). \tag{2-49}$$

In many works it has been demonstrated how the ZA technique outperforms the typical adaptive algorithms in presence of sparse signals [10]-[12], [16] [17] [19] [21]. The zero attractor accelerates the convergence of the algorithm when the majority of the coefficents are zero. With the increase in the number of non-zero coefficients the algorithm tends to deteriorate its perfomance.

2.5.2 The Reweighted Zero-Attracting Strategy

By replacing the f_1 penalty with the log-sum penalty in (2-47) it is obtained the cost function for RZA-LMS algorithm as follows

$$C_{RZA\text{-}LMS}(\boldsymbol{\omega}_i) = E[|d_i - \boldsymbol{\omega}_i^H \boldsymbol{x}_i|^2] + \rho \sum_{i=1}^{M} log(1 + |\boldsymbol{\omega}_{k,i}|/\varepsilon). \tag{2-50}$$

To obtain the recursion of the RZA sparsity-aware LMS algorithm it is necessary to differentiate the regularization term added in (2-50) with respect to $\boldsymbol{\omega}_i^*$, as shown below

$$f_2 = \rho \sum_{m=1}^{M} \log(1 + \frac{|[\boldsymbol{\omega}_{k,i}]_m|}{\varepsilon}), \tag{2-51}$$

$$\frac{\partial(f_2)}{\partial(\boldsymbol{\omega}_i^*)} = \frac{\text{sgn}(\boldsymbol{\omega}_i)}{1 + \varepsilon \|\boldsymbol{\omega}_i\|_1}. \tag{2-52}$$

The reweighted zero attractor term presented in (2-52) takes effect only in coefficients whose magnitudes are comparable to $1/\varepsilon$ and there is a shrinkage exerted on the coefficients larger than $1/\varepsilon$ [10].

The update equation for this technique is then calculated as

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \mu \boldsymbol{x}_i[d_i^* - \boldsymbol{x}_i^H \boldsymbol{\omega}_{i-1}] - \rho \frac{\text{sgn}(\boldsymbol{\omega}_i)}{1 + \varepsilon \|\boldsymbol{\omega}_{i-1}\|_1}. \tag{2-53}$$

The RZA strategy has improved the performance as compared with the ZA mechanism, due to the reweighted zero attractor and the selective shrinkage. According to [10]-[12] a robust performance for non-sparse scenarios is also obtained with the RZA strategy.

2.5.3 The l_0-Norm Strategy

The l_0 Norm constraint LMS algorithm has been proposed for adaptive processing of sparse signals in non-cooperative environments and also in distributed networks [17] and [40]. Similarly to the ZA and RZA techniques, the l_0 Norm has been implemented to accelerate the sparse system identification task. In this case the penalty function is given by

$$f_3 = \rho \|\boldsymbol{\omega}_i\|_0, \tag{2-54}$$

where $\|.\|$ is the l_0 norm as indicates the name of the strategy. An approximation of the l_o according to [40] is

$$\|\boldsymbol{\omega}_i\|_0 = \sum_{i=0}^{L-1} (1 - e^{-\beta(\boldsymbol{\omega}_i)}), \tag{2-55}$$

where β is the attractor parameter that equalizes both terms of the equation as it tends to infinity. The cost function with the l_0 norm incorporated is given by

$$C_{l_0\text{-}LMS}(\boldsymbol{\omega}_i) = E[|d_i - \boldsymbol{\omega}_i^H \boldsymbol{x}_i|^2] + \rho \sum_{i=0}^{L-1} (1 - e^{-\beta(\boldsymbol{\omega}_i)}). \tag{2-56}$$

By minimizing the cost function the gradient recursion of (2-56) that represents the update equation of this sparsity-aware technique is described as

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \mu \boldsymbol{x}_i [d_i^* - \boldsymbol{x}_i^H \boldsymbol{\omega}_{i-1}] - \mu\rho\beta \mathrm{sgn}(\boldsymbol{\omega}_{i-1}) e^{-\beta(\boldsymbol{\omega}_{i-1})}. \tag{2-57}$$

2.5.4 The Oracle Algorithm

The oracle algorithm, also developed to exploit sparse systems is able to identify the non-zero positions of the parameter vector. The main goal of the method is to minimize the cost function described by

$$C_{or\text{-}LMS}(\boldsymbol{P}_i, \boldsymbol{\omega}_i) = E[|d_i + \boldsymbol{\omega}_i^H \boldsymbol{P}_i \boldsymbol{x}_i|^2], \tag{2-58}$$

where $\boldsymbol{P}_i$ is an $M \times M$ diagonal matrix that contains the positions of the non-zero taps [16]. The output estimate $\widehat{d_i}$ is given by

$$\widehat{d_i} = \boldsymbol{\omega}_i^H \boldsymbol{P}_i \boldsymbol{x}_i. \tag{2-59}$$

The oracle technique has been proposed using the LMS algorithm as presented in (2-58) and the RLS algorithm [13]. This method significantly outperforms the classical and the standard sparsity-aware adaptive algorithms in terms of convergence rate and error computation. As part of our research in chapter 5 a distributed method is presented to approach the oracle algorithm for distributed parameter and spectrum estimation.

3
Distributed Conjugate Gradient Algorithms

3.1
Introduction

In the last few years with the deployment of distributed networks for parameter sensing, many adaptive algorithms have been used to obtain the desired information. Least-mean square (LMS) and recursive least-squares (RLS) type algorithms have been applied so far [1]-[9], [35]-[37]. These algorithms have been reported using incremental, consensus and diffusion communication strategies [9]-[11],[35],[36]. The diffusion strategy has been more exploited due to its better response in terms of convergence and mean square deviation (MSD) and error (MSE) values at steady state.

The Conjugate Gradient (CG) algorithm is based on the CG method [22] and also has been successfully applied for distributed processing [15], [18]. Although the LMS-based algorithms are computationally simpler when compared with conjugate gradient (CG) algorithms, the adaptation speed is often slow, especially for the conventional LMS algorithm. Second, with the increase of the adaptation speed, the system stability may decrease significantly. The RLS-based algorithms usually have a high computational complexity and are prone to numerical instability [18]. The CG algorithm, often results in an algorithm of faster convergence than the LMS and with a very close performance as compared to the RLS, that can also achieves better numerical stability [14]. There are two well-known variants of CG algorithms, the conventional CG (CCG) and the modified CG (MCG) algorithms [15].

In this chapter we propose distributed CG algorithms based on consensus and two variants of diffusion strategies for parameter estimation over sensor networks. Specifically, we develop a distributed standard and modified CG algorithms using the consensus , ATC and CTA diffusion protocols.The proposed algorithms are compared with recently reported algorithms in the literature. The particular application presented in this paper is parameter estimation over sensor networks, which can be found in many scenarios of practical interest such as environmental monitoring and military defense.

3.2 System Model and Problem Statement

In this section, we describe the system model of the distributed estimation scheme and introduce the problem statement.

The system model of the network consists of N nodes that exchange information between them, where each node represents an adaptive parameter vector with neighborhood described by the set N_k. An strongly connected network with partially topology is assume, where all nodes receive sensing information (directly or indirectly) from the other agents depending on the communication protocol [1], [18], [41]. The main task of the parameter estimation problem is to adjust the $M \times 1$ weight vector $\boldsymbol{\omega}_k$ of each node, where M is the length of the filter [1]. The desired signal $d_{k,i}$ at each time i is a scalar random process given by

$$d_{k,i} = \boldsymbol{\omega}_0^H \boldsymbol{x}_{k,i} + n_{k,i}, \tag{3-1}$$

where $\boldsymbol{\omega}_0$ is the $M \times 1$ system weight vector, $\boldsymbol{x}_{k,i}$ is the $M \times 1$ input signal vector and $n_{k,i}$ is the measurement noise. The output estimate is given by

$$y_{k,i} = \boldsymbol{\omega}_{k,i}^H \boldsymbol{x}_{k,i}. \tag{3-2}$$

The main goal of the network is to minimize the following cost function:

$$C(\boldsymbol{\omega}) = \sum_{k=1}^{N} E[|d_{k,i} - \boldsymbol{\omega}_{k,i}^H \boldsymbol{x}_{k,i}|^2]. \tag{3-3}$$

By solving this minimization problem it is possible to obtain the optimum solution of the weight vector at each node. The optimum solution for the cost function is given by

$$\boldsymbol{\omega}_{k,i} = \boldsymbol{R}_{k,i}^{-1} \boldsymbol{b}_{k,i}, \tag{3-4}$$

where $\boldsymbol{R}_{k,i} = E[\boldsymbol{x}_{k,i} \boldsymbol{x}_{k,i}^H]$ is the $M \times M$ correlation matrix of the input data vector $\boldsymbol{x}_{k,i}$, and $\boldsymbol{b}_{k,i} = E[d_{k,i}^* \boldsymbol{x}_{k,i}]$ is the $M \times 1$ cross-correlation vector between the input data and $d_{k,i}$. As usual in distributed estimation, we assume that the optimum solutions $\boldsymbol{\omega}_{k,i}$ are the same across the the network (i.e., for all k) [1]. In the following sections we focus on distributed CG algorithms to minimize 3-3.

3.3 Proposed Distributed Consensus CG Algorithm

In this section, we present the proposed distributed CG algorithm using the consensus strategy. We first derive the CG algorithm and then consider the consensus protocol.

3.3.1 Derivation of the CG algorithm

The CG method can be applied to adaptive filtering problems [14]. The main objective in this task is to solve (3-4). The cost function for one agent, say agent k is given by

$$C_{CG}(\boldsymbol{\omega}_k) = \frac{1}{2}\boldsymbol{\omega}^H \boldsymbol{R}\boldsymbol{\omega} - \boldsymbol{b}^H\boldsymbol{\omega}. \tag{3-5}$$

For distributed processing over sensor networks, we present the following derivation. The CG algorithm does not need to compute an update to the inverse of $\boldsymbol{R}$, which is an advantage as compared with RLS-based algorithms. Using the data window with an exponential decay, the resulting autocorrelation matrix and cross-correlation vector use the forgetting factor λ, which is the same as the forgetting factor of the RLS algorithm. They are defined as

$$\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H, \tag{3-6}$$

$$\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^*\boldsymbol{x}_{k,i}. \tag{3-7}$$

For each time instant i, the CG computes the weights $\boldsymbol{\omega}_{k,i}$ for each iteration j until convergence, i.e, $\boldsymbol{\omega}_{k,i}(j)$ [18]. The gradient of the method in the negative direction is obtained as follows

$$\boldsymbol{g}_{k,i}(j) = \boldsymbol{b}_{k,i} - \boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i}(j) = -\nabla C_{CG}(\boldsymbol{\omega}_{k,i}), \tag{3-8}$$

Calculating the Krylov subspace [22] through different operations, the recursion is given by

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j), \tag{3-9}$$

where $\boldsymbol{p}$ is the conjugate direction vector of $\boldsymbol{g}$ and α is the step size that minimizes the cost function in (3-5) by replacing (3-8) in (3-4). Both parameters are calculated as follows:

$$\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}, \tag{3-10}$$

$$\boldsymbol{p}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j). \tag{3-11}$$

The parameter β is calculated using the Gram-Schmidt orthogonalization procedure [23] as given by

$$\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}. \tag{3-12}$$

Applying the CG method to a distributed network the cost function is expressed based on the information exchanged between all nodes $k = 1, 2..., N$. Each of the equations presented so far takes place at each agent during the iterations of the CG algorithm. Therefore, we have the cost function:

$$C_{CG}(\boldsymbol{\omega}_{k,i}) = \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^H \boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^H\boldsymbol{\omega}_{k,i}. \tag{3-13}$$

3.3.2
Consensus Conjugate Gradient.

In the consensus strategy, all nodes interact with their neighbors sharing and reaching agreement about the system parameter vector. Each node k is able to run its update simultaneously with the other agents [1][6]. Figure 4.1 illustrates the distributed network scheme for consensus and diffusion strategies.

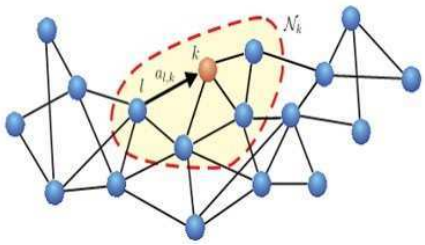

Figure 3.1 – Distributed connected network processing.

There is a cooperation factor in consensus that is a convex combination of the iterations available at the neighborhood of agent k. This combination is then updated with the previous value of the node. This mechanism performs the adaptation and learning at the same time [1].

The consensus cooperation strategy imposes a mathematical constraint so that all connected agents converge same weight $\boldsymbol{\omega}_{k,i}$ parameter vector. It means that this protocol demands the equality of the estimates within the network [28]. The optimization problem that involves the cost function of the

distributed CG algorithm with the consensus strategy is given by

$$\begin{aligned} \min\, C_{CG}(\boldsymbol{\omega}_{k,i}) &= \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^{H}\boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^{H}\boldsymbol{\omega}_{k,i} \\ \text{s.t } \boldsymbol{\omega}_k &= \boldsymbol{\omega}_k,\ l = 1,2,...,k,\ l \in N_k, \end{aligned} \tag{3-14}$$

Based on this constraint it is necessary to apply the method of Lagrange Multipliers for the network wise equality [28].

$$\mathcal{L}(\boldsymbol{\omega}_k, \lambda) = \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^{H}\boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^{H}\boldsymbol{\omega}_{k,i} + \lambda(\boldsymbol{\omega}_k - \boldsymbol{\omega}_k),\ l \in N_k \tag{3-15}$$

To obtain the recursion for updating the parameter vector and the associated Lagrange Multipliers it is necessary to apply the derivation process to the cost function. The derivative of the equations after some operations are given by

$$\begin{aligned} \nabla_{\omega^*}\mathcal{L}(\boldsymbol{\omega}_k, \lambda) &= \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^{H}\boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^{H}\boldsymbol{\omega}_{k,i} + \lambda(\boldsymbol{\omega}_k - \boldsymbol{\omega}_l) = 0,\ l \in N_k, \\ &= \sum_{k=1}^{N} \boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i} + \lambda, \end{aligned} \tag{3-16}$$

$$\begin{aligned} \nabla_{\lambda^*}\mathcal{L}(\boldsymbol{\omega}_k, \lambda) &= \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^{H}\boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^{H}\boldsymbol{\omega}_{k,i} + \lambda(\boldsymbol{\omega}_{k,i} - \boldsymbol{\omega}_{l,i}) = 0,\ l \in N_k, \\ &= \sum_{k=1}^{N} \boldsymbol{\omega}_{k,i} - \boldsymbol{\omega}_{l,i},\ l \in N_k. \end{aligned} \tag{3-17}$$

According to the discrete CG algorithm in (3-9) and to the distributed consensus algorithm [37], we have

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) + \lambda \sum_{l \in N_k} (\boldsymbol{\omega}_{k,i-1} - \boldsymbol{\omega}_{l,i-1}). \tag{3-18}$$

Following the steps of conjugate gradient presented from (3-5) to (3-12) and taking into account the constraint in (3-14), we obtain the recursion given by

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) + \alpha(j)[\boldsymbol{p}_{k,i}(j) + \lambda \sum_{l \in N_k} (\boldsymbol{\omega}_{k,i-1} - \boldsymbol{\omega}_{l,i-1})], \tag{3-19}$$

where $0 < \lambda < 1$ and the last terms in the bracket are set to enforce equality.

A number of methods have been proposed in the literature [28][38][39], based on the Alternating Direction Method of Multipliers (ADMM) for consensus strategy. For consensus-distributed algorithms the combination step is based on the connectivity among nodes, where the local estimation is given by

$$\boldsymbol{\varphi}_{k,i-1} = \sum_{l \epsilon N_k} a_{kl} \boldsymbol{\omega}_{l,i-1}, \tag{3-20}$$

where a_{kl} represents the link among the nodes and should comply with

$$\sum_{l \epsilon N_k} a_{kl} = 1, \ l \epsilon N_{k,i}, \forall k. \tag{3-21}$$

In this work the strategy adopted for the a_{kl} combiner is the Metropolis rule [1] given by

$$a_{kl} = \begin{cases} \frac{1}{max\{|\mathcal{N}_k|,|\mathcal{N}_l|\}} & \text{if } k \neq l \text{ are linked,} \\ 1 - \sum\limits_{l \in \mathcal{N}_k / k} a_{kl}, & \text{for } k = l. \end{cases} \tag{3-22}$$

where the $N \times N$ matrix $\mathbf{A}$ that contains the a_{kl} values must be left stochastic, with the sum of the terms of each column equal to one ($A^T \mathbf{1} = 1$).

The proposed distributed Consensus CG algorithm based on the derivation steps obtains the updated weight substituting (3-20) in (3-9), resulting in

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i-1}(j) + \alpha_{k,i}(j) \boldsymbol{p}_{k,i}(j). \tag{3-23}$$

The pseudo code of the proposed distributed conventional consensus CG algorithm is presented bellow in Table 3.1.

3.3.3 Distributed Consensus MCG Algorithm

The Modified CG (MCG) algorithm comes from the conventional CG algorithm previously presented. This version is proposed to have just one iteration per coefficient update. For this version the residual is calculated using (3-6), (3-8) and (3-9) [14]:

$$\boldsymbol{g}_{k,i} = \boldsymbol{b}_{k,i} - \boldsymbol{R}_{k,i} \boldsymbol{\varphi}_{k,i} = \lambda \boldsymbol{g}_{k,i-1} - \alpha_{k,i} \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i-1} + \boldsymbol{x}_{k,i} [d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H \boldsymbol{x}_{k,i}]. \tag{3-24}$$

Table 3.1 – Consensus CG Algorithm

Parameter initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta \boldsymbol{I}$, $\boldsymbol{b}_{k,0} = 0$,...
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{\varphi}_{k,i-1} = \sum_{l \epsilon N_k} a_{lk} \omega_{l,i-1}$
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i} \boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{g}_{k,i}(0) = \boldsymbol{b}_{k,i} - \boldsymbol{R}_{k,i} \boldsymbol{\omega}_{k,i-1}$
    $\boldsymbol{p}_{k,i}(0) = \boldsymbol{g}_{k,i}(0)$
    For each CG iteration $j = 1$ until convergence
      $\alpha_{k,i}(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
      $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i-1}(j) - \alpha_{k,i}(j)\boldsymbol{p}_{k,i}(j)$
      $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j-1) - \alpha_{k,i}(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
      $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
      $\boldsymbol{p}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j-1)$
    End For
    $\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i}(j_{last})$
  End for
End for

The previous equation (3-24) is multiplied by the search direction vector:

$$\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i} = \lambda \boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i-1} - \alpha_{k,i} \boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i-1} + \boldsymbol{p}_{k,i}^H \boldsymbol{x}_{k,i} [d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H \boldsymbol{x}_{k,i}], \quad (3\text{-}25)$$

Applying the expected value, considering $\boldsymbol{p}_{k,i-1}$ uncorrelated with $\boldsymbol{x}_{k,i}$, $d_{k,i}$ and $\boldsymbol{\varphi}_{k,i}$, and that the algorithm converges, the last term of (3-25) can be neglected. The line search to compute α has to satisfy the convergence bound [14] given by

$$E[\alpha_{k,i}] \in \frac{E[\boldsymbol{p}_{k,i-1}^H \boldsymbol{g}_{k,i}]}{E[\boldsymbol{p}_{k,i-1}^H \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i-1}]} [\lambda - 0.5, 1], \quad (3\text{-}26)$$

$$\alpha_{k,i} = \eta \frac{\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i}}, \quad (3\text{-}27)$$

where $(\lambda - 0.5) \leq \eta \leq \lambda$. The Polak-Ribiere method [14] for the computation of β is given by

$$\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H \boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i}^H \boldsymbol{g}_{k,i}}, \quad (3\text{-}28)$$

and should be used for better performance. Table 3.2 shows the details of the MCG algorithm for consensus strategy taking into account the considerations previously discussed.

Table 3.2 – Consensus MCG Algorithm

Parameter initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta \boldsymbol{I}$, $\boldsymbol{g}_{k,0} = \boldsymbol{b}$, $\boldsymbol{p}_{k,1} = \boldsymbol{g}_{k,0}$,...
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{\varphi}_{l,i-1} = \sum_{l \epsilon N_k} a_{lk} \omega_{l,i-1}$
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i} \boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{g}k, 1 = \boldsymbol{b}_{k,0}$
    $\boldsymbol{p}k, 1 = \boldsymbol{g}_{k,1}$
    $\alpha_{k,i} = \eta \frac{\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i}}, (\lambda - 0.5) \leq \eta \leq \lambda$
    $\boldsymbol{\omega}_{k,i} = \boldsymbol{\varphi}_{l,i-1} - \alpha_{k,i} \boldsymbol{p}_{k,i}$
    $\boldsymbol{g}_{k,i} = \lambda \boldsymbol{g}_{k,i} - \alpha_{k,i} \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i-1} + \boldsymbol{x}_{k,i} [d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H \boldsymbol{x}_{k,i}]$
    $\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H \boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i-1}^H \boldsymbol{g}_{k,i-1}}$
    $\boldsymbol{p}_{k,i} = \boldsymbol{g}_{k,i} + \beta_{k,i} \boldsymbol{p}_{k,i-1}$
  End For
End for

3.4 Proposed Distributed Diffusion Conjugate Gradient.

In diffusion protocols there are two well-known variants that differ in the choice of the output variable (which can be either the output of the combination or of the adaptation steps). The variants are named Combine then-Adapt (CTA) and Adapt-then-Combine (ATC). Both perform adaptation and learning at the same time [1] [6] and the main difference between them is the point in the recursions where the error is measured. These two variants are presented in this section. We first introduce the diffusion conventional and then the modified CG versions.

3.4.1 CTA Distributed CG Algorithm

In the CTA variant of diffusion strategy we first evaluate the convex combination term into an intermediate state variable $\boldsymbol{\varphi}_{k,i-1}$ and subsequently use it to perform the weight update [1] as follows:

$$\boldsymbol{\varphi}_{k,i-i} = \sum_{l \epsilon N_k} a_{lk} \boldsymbol{\omega}_{l,i-1}, \tag{3-29}$$

This combination is employed then to estimate the weight at each node k:

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) + \alpha_{k,i}(j)\boldsymbol{p}k, i(j); \tag{3-30}$$

where $\boldsymbol{\omega}_{k,i}(0) = \boldsymbol{\varphi}_{k,i}$ [15]. The rest of the derivation is similar to the Consensus-CG solution and the pseudo-code is detailed in Table 3.3.

Table 3.3 – CTA CG Algorithm

Parameter initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta\boldsymbol{I}$, $\boldsymbol{b}_{k,0} = 0$,...
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{R}_{k,i} = \lambda\boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda\boldsymbol{b}_{k,i-1} + d_{k,i}^*\boldsymbol{x}_{k,i}$
    $\boldsymbol{\varphi}_{k,i} = \sum_{l\epsilon N_k} a_{lk}\omega_{l,i-1}$
    $\boldsymbol{\omega}_{k,i}(0) = \boldsymbol{\varphi}_{k,i}$
    $\boldsymbol{g}_{k,i}(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0)\boldsymbol{\omega}_{k,i-1}$
    $\boldsymbol{p}_{k,i}(0) = \boldsymbol{g}_{k,i}(0)$
    For each CG iteration $j = 1$ until convergence
      $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
      $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j)$
      $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j-1) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
      $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
      $\boldsymbol{p}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j-1)$
    End For
    $\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i}(j_{last})$
  End for
End for

The MCG version for the CTA strategy is very similar to the ATC strategy proposed, but follows slightly different steps. It takes into account the parameters in (3-25),(3-27) and (3-28) presented in the previous section. In this case, we substitute the term $\boldsymbol{\omega}_{k,i}^H$ by $\boldsymbol{\varphi}_{k,i-1}^H$ in (3-25). The implementation of the strategy is shown below in Table 3.4.

As its name indicates and the details in Table 3.4, the CTA strategy first involves a combination step and then an adaptation step [6].

3.4.2 ATC Distributed CG Algorithm

In the ATC protocol, the adaptation step is the fisrt one, that is obtained from the previous value $\boldsymbol{\omega}_{k,i-1}$ at each node [1][6]. All other agents in the network perform a similar step simultaneously and update their parameters

Table 3.4 – CTA MCG Algorithm

Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta \boldsymbol{I}$, $\boldsymbol{g}_{k,0} = \boldsymbol{b}$, $\boldsymbol{p}_{k,1} = \boldsymbol{g}_{k,0}$,...
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{\varphi}_{k,i-1} = \sum_{l \epsilon N_k} a_{lk} \omega_{l,i-1}$
    $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i}$
    $\alpha_{k,i} = \eta \frac{\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i}}$, $(\lambda - 0.5) \leq \eta \leq \lambda$
    $\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i-1} - \alpha_{k,i} \boldsymbol{p}_{k,i}$
    $\boldsymbol{g}_{k,i} = \lambda \boldsymbol{g}_{k,i} - \alpha_{k,i} \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i-1} + \boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\varphi}_{k,i-1}^H \boldsymbol{x}_{k,i}]$
    $\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H \boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i-1}^H \boldsymbol{g}_{k,i-1}}$
    $\boldsymbol{p}_{k,i} = \boldsymbol{g}_{k,i} + \beta_{k,i} \boldsymbol{p}_{k,i-1}$
  End for
End for

through the CG iterations j.

$$\begin{aligned} \boldsymbol{\omega}_{k,i}(j) &= \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j) \boldsymbol{p}_{k,i}(j), \\ \boldsymbol{\varphi}_{k,i} &= \boldsymbol{\omega}_{k,i}(j_{last}). \end{aligned} \tag{3-31}$$

The second step is the convex combination of the adaptation from the neighbors of each agent after all j iterations, is given by

$$\boldsymbol{\omega}_{k,i} = \sum_{l \epsilon N_k} a_{lk} \boldsymbol{\varphi}_{l,i}. \tag{3-32}$$

Table 3.5 shows the pseudo code for the ATC strategy.

The MCG version for the ATC strategy is very similar to the CTA version presented, but performs its particular tasks. Table 3.6 shows the details of the ATC MCG algorithm.

Table 3.5 – ATC CG Algorithm

Parameter initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta I$, $\boldsymbol{b}_{k,0} = 0$
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{g}_{k,i}(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0)\boldsymbol{\omega}_{k,i-1}(0)$
    $\boldsymbol{p}_{k,i}(0) = \boldsymbol{g}_{k,i}(0)$
    For each CG iteration $j = 1$ until convergence
      $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
      $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j)$
      $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j-1) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
      $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
      $\boldsymbol{p}_{k,i}(j+1) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j)$
    End For
    $\boldsymbol{\varphi}_{k,i} = \boldsymbol{\omega}_{k,i}(j_{last})$
    $\boldsymbol{\omega}_{k,i} = \sum_{l \epsilon N_k} a_{lk}\boldsymbol{\varphi}_{l,i}$
  End for
End for

Table 3.6 – ATC MCG Algorithm

Parameter initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta \boldsymbol{I}$, $\boldsymbol{g}_{k,0} = \boldsymbol{b}$, $\boldsymbol{p}_{k,1} = \boldsymbol{g}_{k,0}$,...
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\alpha_{k,i} = \eta \frac{\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i}}, (\lambda - 0.5) \leq \eta \leq \lambda$
    $\boldsymbol{\varphi}_{k,i} = \boldsymbol{\omega}_{k,i-1} - \alpha_{k,i}\boldsymbol{p}_{k,i}$
    $\boldsymbol{g}_{k,i} = \lambda \boldsymbol{g}_{k,i-1} - \alpha_{k,i}\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1}$
      $+\boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H \boldsymbol{x}_{k,i}]$
    $\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H \boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i-1}^H \boldsymbol{g}_{k,i-1}}$
    $\boldsymbol{p}_{k,i} = \boldsymbol{g}_{k,i} + \beta_{k,i}\boldsymbol{p}_{k,i-1}$
  End for
  $\boldsymbol{\omega}_{k,i} = \sum_{l \epsilon N_k} a_{lk}\boldsymbol{\varphi}_{l,i}$
End for

3.5 Computational Complexity

The proposed consensus as well as diffusion CG methods have a quadratic computational cost and depend on the number of nodes connected and the CG

iterations.

Table 3.7 shows the computational complexity of all presented methods in terms of additions and multiplications. J is the maximum number of CG iterations and L is the number of linked nodes.

Table 3.7 – Computational Complexity of Distributed CG Algorithms

Method	Additions	Multiplications
Consensus-CG	$3LM + LJ(M^2 + 4M - 2)$	$L(2M^2 + 2M)$ $+LJ(3M^2 + 2M)$
CTA-CG	$L(M^2 + 2M)$ $+LJ(2M^2 + 6M - 3)$	$L(2M^2 + 4M)$ $+LJ(3M^2 + 4M - 1)$
ATC-CG	$L(M^2 + 3M - 1)$ $+LJ(M^2 + 6M - 3)$	$L(2M^2 + 3M)$ $+LJ(3M^2 + 4M - 1)$
Consensus-MCG	$L(M^2 + 7M - 2)$	$L(4M^2 + 4M)$
CTA-MCG	$L(3M^2 + 9M - 4)$	$L(4M^2 + 9M - 1)$
ATC-MCG	$L(4M^2 + 9M - 3)$	$L(6M^2 + 8M - 1)$

It can be seen that the complexity of the modified versions is lower than the conventional methods for $J > 1$. The conjugate gradient method has a lower computational complexity as compared with the RLS algorithm.

3.6 Simulations Results

In this section, we evaluated the proposed distributed consensus and diffusion CG algorithms and compare them with existing algorithms. The results are based on the mean square deviation (MSD) of the network. We consider a network with 20 nodes and 1000 iterations per run. Each iteration corresponds to a time instant. The results are averaged over 100 experiments. The length of the filter is 10 and the input signal, a complex Gaussian noise, has unit variance. The SNR is 30 dB.

3.6.1 Comparison between consensus and diffusion distributed CG algorithms

The system parameter vector was randomly set. After all the iterations, the performance of each algorithm in terms of MSD is shown in Figure 4.2.

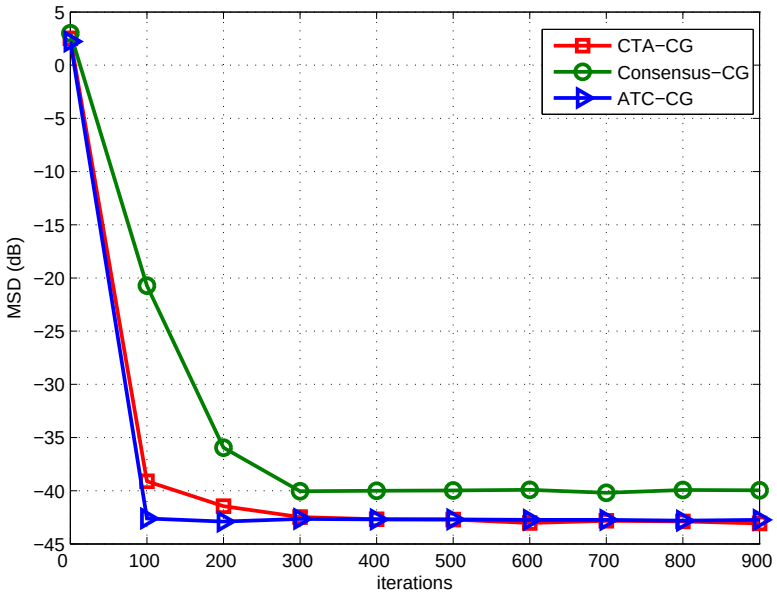

Figure 3.2 – MSD of the network for distributed Consensus-CG,CTA-CG and ATC-CG algorithms with $\lambda = 0.99$, $\delta = 10^{-2}$, $J_{CCG} = 5$.

The results show that the diffusion strategies outperform the consensus version. Specifically, the best results are obtained for the ATC variant in terms of convergence and MSD value at steady state.

3.6.2 Comparison between consensus and diffusion distributed MCG algorithms

Generally the MCG algorithms have a better performance than the standard ones because they have the ability to perform the estimation at each time instant, without the CG internal iterations of the conventional method. Figure 4.3 presents the MSD curve for modified variants of distributed CG algorithm. We also illustrate the MSD curve for comparison between Consensus-CG, Consensus-MCG, ATC-CG and ATC-MCG in figure 4.4.

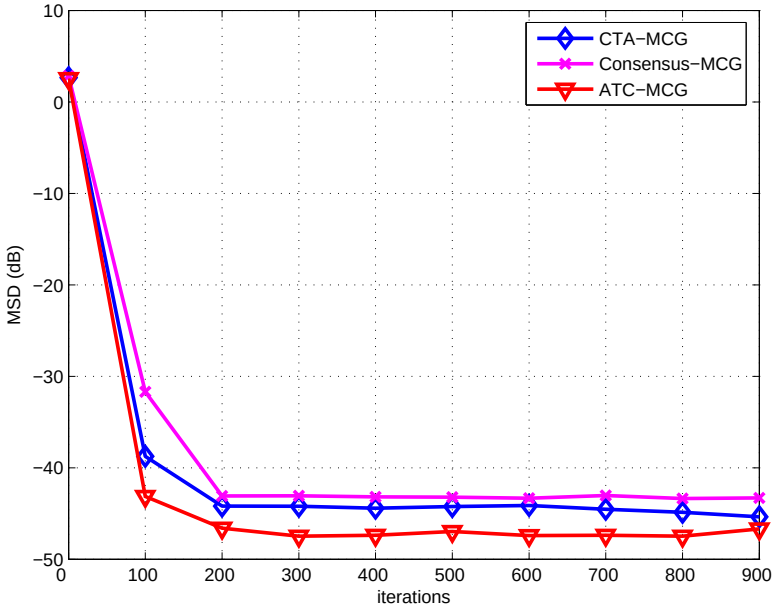

Figure 3.3 – MSD of the network for distributed Consensus-MCG, CTA-MCG and ATC-MCG algorithms with $\lambda = 0.99$, $\delta = 10^{-2}$.

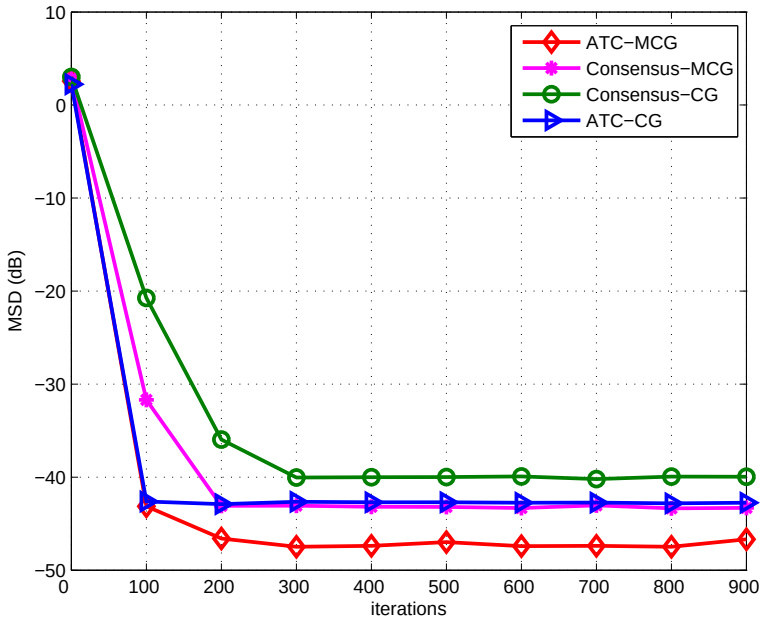

Figure 3.4 – MSD of the network for distributed Consensus-CG, Consensus-MCG, ATC-MCG and ATC-MCG algorithms with $\lambda = 0.99$, $\delta = 10^{-2}$, $J_{CCG} = 5$.

From figures 3.3 and 3.4 the MCG algorithm performs better than the conventional CG (CCG). Specifically, the ATC strategy achieves the lowest MSD value and has a faster convergence as compared with consensus and

CTA protocols. Figure 3.5 bellow shows a comparison between the proposed distributed CG and MCG algorithms and the well-known LMS and RLS algorithms, all of them using the ATC strategy. In terms of stability the MCG also outperform the the RLS algorithm.

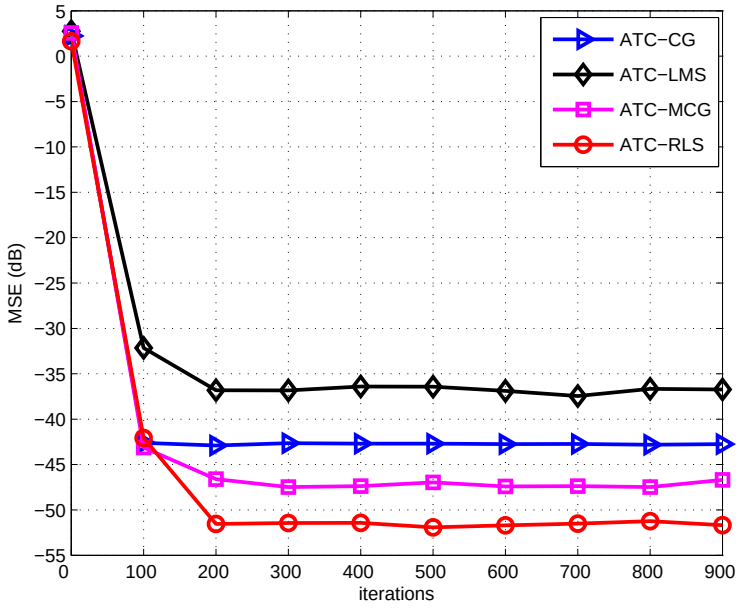

Figure 3.5 – MSD of the network for distributed ATC-CG, ATC-MCG, ATC-LMS and ATC-RLS algorithms with $\lambda = 0.99$, $\delta = 10^{-3}$, $J_{CCG} = 5$, $\mu_{LMS} = 0.035$.

3.7 Summary

In this chapter we have proposed distributed consensus and diffsuion CG algorithms for parameter estimation over sensor networks. The proposed distributed CG algorithms have a faster convergence than the LMS and a very similar performance to the RLS, specifically the diffusion ATC MCG algorithm. Simulation results have shown that the developed methods can be used for adaptive parameter estimation and can be employed in other applications. Due to the conditions of sparse parameter vectors, will be presented in the next chapter the development of these methods exploiting sparsity.

4
Distributed Sparsity-Aware Conjugate Gradient Algorithms

4.1
Introduction

A sparse system is defined as a system whose impulse response contains many zero or near-zero coefficients and only a few large ones. This kind of system exists in many applications, such as Digital TV transmission where channels can be considered sparse because of the presence of only a few dominant multipaths [40] [45]. Sparse systems also can be found in image processing, especially in images that only have a few nonzero pixel values [46]. Sensor measurements in wireless sensor networks and industrial applications also exhibit certain levels of sparsity.

In the last years many algorithms have been developed to exploit the sparse nature of systems to improve their performance [10]-[12]. Distributed strategies also have been developed in the literature for sparse systems [9],[16]-[19],[40], most of them using LMS-based algorithms. In distributed approaches, the nodes exchange information locally and cooperate with each other without the need for a central processor. The information is processed by all nodes and the data diffuse across the network by means of a real-time sharing mechanism. These sparsity-aware algorithms locate and track non-zero coefficients introducing a convex penalty term into the cost function to favor sparsity [9]. Some of the penalty functions that have been been used for sparse systems include an approximation of the l_0-norm [12],[40], the l_1- norm penalty [12],[16] and the log-sum penalty [9],[11],[16].

The l_1 penalty function results in a modified update step with a zero attractor for all the coefficients, for that reason it is called the Zero-Attracting (ZA) strategy. Then, the Reweighted Zero-Attracting (RZA), which uses the log-sum penalty function, employs reweighted step sizes of the zero attractor for different coefficients, inducing the attractor to selectively promote zero coefficients rather than uniformly promote zeros on all the coefficients[18].

In order to improve the performance of distributed CG-based methods, typical metrics of sparse systems are presented and integrated into the cost

function of the CG algorithm. Building on the work described in the previous chapter we propose distributed sparsity-aware CG algorithms based on the consensus and diffusion strategies for parameter estimation over sensor networks. Specifically, we develop sparsity-aware CG algorithms using l_1 and log-sum penalty functions. The proposed algorithms are compared with recently reported sparsity-aware algorithms in the literature. The application scenario is parameter estimation over sensor networks.

4.2 Sparse Distributed Estimation Model and Problem Statement

We consider diffusion and consensus strategies for a network where each agent k has access at each time instant to a realization of zero-mean spatial data $\{d_{k,i}, \boldsymbol{x}_{k,i}\}$[19]-[25], as described by

$$d_{k,i} = \boldsymbol{\omega}_0^H \boldsymbol{x}_{k,i} + n_{k,i}, \tag{4-1}$$

where $\boldsymbol{\omega}_0$ is the $M \times 1$ system weight vector, $\boldsymbol{x}_{k,i}$ is the $M \times 1$ input signal vector and $n_{k,i}$ is the measurement noise.

For a network with possibly sparse parameter vectors, the cost function also involves a penalty function which exploits sparsity. In this case the network needs to solve the following optimization problem:

$$\min C(\boldsymbol{\omega}_{k,i}) = \sum\nolimits_{k=1}^{N} E[|d_{k,i} - \boldsymbol{\omega}_{k,i}^H \boldsymbol{x}_{k,i}|^2] + f(\boldsymbol{\omega}_{k,i}), \tag{4-2}$$

where $f(\boldsymbol{\omega}_{k,i})$ is a penalty function that exploits the sparsity in the parameter vector $\boldsymbol{\omega}_{k,i}$. In the following sections we focus on distributed diffusion CG algorithms to solve (4-2).

4.3 Proposed Sparsity-Aware Distributed Consensus CG

Based on the previous study of distributed CG algorithm presented in chapter 3, the following description presents the general strategy of distributed sparsity-aware consensus CG algorithms using l_1 (ZA) and log-sum (RZA) norm penalty functions.

These two techniques were chosen to evaluated its performance using the CG algorithm, as it have been done with other adaptive algorithms like LMS and RLS reported in [10]-[14].

4.3.1 ZA and RZA Consensus CG algorithms

The cost function in this case is given by

$$\begin{gathered} C_{CG}(\boldsymbol{\omega}_{k,i}) = \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^{H}\boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^{H}\boldsymbol{\omega}_{k,i} + f_1(\boldsymbol{\omega}_{k,i}) \\ \text{subject to. } \boldsymbol{\omega}_k = \boldsymbol{\omega}_k,\ l = 1, 2, ..., k,\ l \in N_k, \end{gathered} \tag{4-3}$$

where f_1 denotes the l_1 penalty function (ZA) and is defined by

$$f_1 = \rho\|\boldsymbol{\omega}_{k,i}(j)\|_1. \tag{4-4}$$

Applying the partial derivative of the penalty function with respect to the parameter vector $\boldsymbol{\omega}_{k,i}^{*}$ gives

$$\frac{\partial(f_1)}{\partial(\boldsymbol{\omega}_{k,i}^{*})} = \text{sgn}(\boldsymbol{\omega}_{k,i}) = \begin{cases} \frac{[\boldsymbol{\omega}_{k,i}]_m}{|[\boldsymbol{\omega}_{k,i}]_m|}, & \text{if } [\boldsymbol{\omega}_{k,i}]_m \neq 0 \\ 0, & \text{if } \boldsymbol{\omega}_{k,i} = 0, \end{cases} \tag{4-5}$$

where $[\boldsymbol{x}]_m$ is the m-th entry of $\boldsymbol{x}$.

When instead of the f_1 function, the logarithmic penalty function f_2 is used in the cost function, we obtain

$$C_{CG}(\boldsymbol{\omega}_{k,i}) = \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^{H}\boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^{H}\boldsymbol{\omega}_{k,i} + \rho\sum_{i=1}^{M} log(1 + |\boldsymbol{\omega}_{k,i}|/\varepsilon), \tag{4-6}$$

The partial derivative of the penalty function with respect to $\boldsymbol{\omega}_{k,i}^{*}$ is given by

$$f_2 = \rho\sum_{m=1}^{M} \log(1 + \frac{|[\boldsymbol{\omega}_{k,i}]_m|}{\varepsilon}), \tag{4-7}$$

$$\frac{\partial(f_2)}{\partial(\boldsymbol{\omega}_{k,i}^{*})} = \frac{\text{sgn}(\boldsymbol{\omega}_{k,i})}{1 + \varepsilon\|\boldsymbol{\omega}_{k,i}\|_1}. \tag{4-8}$$

In both cases the sparsity-aware algorithms attract to zero the values of the parameter vector which are very small or are not useful. This results in algorithms with faster convergence and lower MSD values as can be seen in the following sections. Using equations (4-5) and (4-8), we obtain sparsity-aware algorithms with distributed strategies. Table 4.1 shows the pseudocode of the sparsity-aware algorithms for the consensus protocol. The same steps are applied for ZA and RZA methods, where the main difference lies in the chosen penalty function.

Table 4.1 – Sparsity-Aware Consensus CG Algorithm

Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta I$, $\boldsymbol{b}_{k,0} = 0$
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{\varphi}_{k,i-1} = \sum_{l \epsilon N_k} a_{lk} \omega_{l,i-1}$
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i} \boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{g}k, i(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0) \boldsymbol{\omega}_{k,i-1}(0)$
    $\boldsymbol{p}k, i(0) = \boldsymbol{g}_{k,i}(0)$
    For each CG iteration $j = 1$ until convergence
      $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
      $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i} - \alpha(j)\boldsymbol{p}_{k,i}(j)$
      $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
      $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
      $\boldsymbol{p}_{k,i}(j+1) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j)$
    End For
    $\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i}(j_{last}) - \rho \frac{\partial(f_{1,2})}{\partial \boldsymbol{\omega}_{k,i}^*}$
  End for
End for

4.3.2 ZA and RZA Consensus MCG algorithm

The sparsity-aware versions for the Consensus MCG algorithm follow the steps explained of the Consensus MCG algorithm in subsection 3.3.3, including the penalty functions (4-4) or (4-7) in the cost function given by

$$\mathcal{L}(\boldsymbol{\omega}_{k,i}, \lambda) = \frac{1}{2} \sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^H \boldsymbol{R}_{k,i} \boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^H \boldsymbol{\omega}_{k,i} + \lambda(\boldsymbol{\omega}_k - \boldsymbol{\omega}_k) + f_{1,2}(\boldsymbol{\omega}_{k,i}),\ l \in N_k, \tag{4-9}$$

The derivatives of the cost functions of the ZA and RZA consensus algorithms with respect to $\boldsymbol{\omega}_{k,i}$ are obtained as follows

$$\nabla_{\boldsymbol{\omega}^*,\lambda} \mathcal{L}(\boldsymbol{\omega}_{k,i}, \lambda) = \sum_{k=1}^{N} \boldsymbol{R}_{k,i} \boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i} + \lambda(\boldsymbol{\omega}_{k,i} - \boldsymbol{\omega}_{l,i}) - \rho \mathrm{sgn}(\boldsymbol{\omega}_{k,i}), \tag{4-10}$$

$$\nabla_{\boldsymbol{\omega}^*,\lambda} \mathcal{L}(\boldsymbol{\omega}_{k,i}, \lambda) = \sum_{k=1}^{N} \boldsymbol{R}_{k,i} \boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i} + \lambda(\boldsymbol{\omega}_{k,i} - \boldsymbol{\omega}_{l,i}) - \rho \frac{\mathrm{sgn}(\boldsymbol{\omega}_{k,i})}{1 + \varepsilon \|\boldsymbol{\omega}_{k,i}\|_1}. \tag{4-11}$$

As a result of the derivation process the penalty functions of the ZA and RZA methods are incorporated in the general recursions of consensus the CG

algorithms given by

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i-1}(j) + \alpha_{k,i}(j)\boldsymbol{p}_{k,i}(j) - \rho \mathrm{sgn}(\boldsymbol{\omega}_{k,i}), \tag{4-12}$$

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i-1}(j) + \alpha_{k,i}(j)\boldsymbol{p}_{k,i}(j) - \rho \frac{\mathrm{sgn}(\boldsymbol{\omega}_{k,i})}{1 + \varepsilon \|\boldsymbol{\omega}_{k,i}\|_1}, \tag{4-13}$$

where $\boldsymbol{\varphi}_{k,i-1}$ is the local estimation that represents the connection between the nodes of the networks and their weights, described as

$$\boldsymbol{\varphi}_{k,i-1} = \sum_{l \epsilon N_k} a_{kl}\boldsymbol{\omega}_{l,i-1}, \tag{4-14}$$

Equations (4-12) and (4-13) correspond to ZA and RZA parameter vector $\boldsymbol{\omega}_{k,i}$ updates, respectively. The residual and the direction vectors are obtained as follows:

$$\boldsymbol{g}_{k,i} = \boldsymbol{b}_{k,i} - \boldsymbol{R}_{k,i}\boldsymbol{\varphi}_{k,i} = \lambda \boldsymbol{g}_{k,i-1} - \alpha_{k,i}\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1} + \boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H \boldsymbol{x}_{k,i}]. \tag{4-15}$$

$$\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i} = \lambda \boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i-1} - \alpha_{k,i}\boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1} + \boldsymbol{p}_{k,i}^H \boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H \boldsymbol{x}_{k,i}]. \tag{4-16}$$

Applying the expected value to equation (4-16), the direction vector $\boldsymbol{p}_{k,i-1}$ is considered uncorrelated with $\boldsymbol{x}_{k,i}$, $d_{k,i}$ and $\boldsymbol{\varphi}_{k,i}$, then the last term of (4-16) can be neglected. The line search to compute α has to satisfy the convergence bound [14] given by

$$E[\alpha_{k,i}] \in \frac{E[\boldsymbol{p}_{k,i-1}^H \boldsymbol{g}_{k,i}]}{E[\boldsymbol{p}_{k,i-1}^H \boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1}]} \left[\lambda - 0.5, 1\right], \tag{4-17}$$

$$\alpha_{k,i} = \eta \frac{\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i}}, (\lambda - 0.5) \leq \eta \leq \lambda, \tag{4-18}$$

The Polak-Ribiere method [14] for the computation of β is given by

$$\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H \boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i}^H \boldsymbol{g}_{k,i}}, \tag{4-19}$$

The main idea of the sparsity-aware consensus MCG algorithm is to obtain better convergence performance and lower MSD value at steady state as compared with previously reported consensus MCG. The pseudo-code is shown below in Table 4.2.

Table 4.2 – ZA and RZA Consensus MCG Algorithm

Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta \boldsymbol{I}$, $\boldsymbol{g}_{k,0} = \boldsymbol{b}$, $\boldsymbol{p}_{k,1} = \boldsymbol{g}_{k,0}$,...
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{\varphi}_{l,i-1} = \sum_{l \epsilon N_k} a_{lk} \omega_{l,i-1}$
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i} \boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{g}k, 1 = \boldsymbol{b}_{k,0}$
    $\boldsymbol{p}k, 1 = \boldsymbol{g}_{k,1}$
    $\alpha_{k,i} = \eta \frac{\boldsymbol{p}_{k,i}^H \boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i}}, (\lambda - 0.5) \leq \eta \leq \lambda$
    $\boldsymbol{\omega}_{k,i} = \boldsymbol{\varphi}_{l,i-1} - \alpha_{k,i} \boldsymbol{p}_{k,i} - \rho \frac{\partial (f_{1,2})}{\partial \boldsymbol{\omega}_{k,i}^*}$
    $\boldsymbol{g}_{k,i} = \lambda \boldsymbol{g}_{k,i} - \alpha_{k,i} \boldsymbol{R}_{k,i} \boldsymbol{p}_{k,i-1} + \boldsymbol{x}_{k,i} [d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H \boldsymbol{x}_{k,i}]$
    $\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H \boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i-1}^H \boldsymbol{g}_{k,i-1}}$
    $\boldsymbol{p}_{k,i} = \boldsymbol{g}_{k,i} + \beta_{k,i} \boldsymbol{p}_{k,i-1}$
  End For
End for

4.4 Proposed Sparsity-Aware Distributed Diffusion CG

Similar to the equation (4-3), but without the consensus constraint we can obtain the cost function for the sparsity-aware CG algorithms using the diffusion strategy, given by

$$C_{CG}(\boldsymbol{\omega}_{k,i}) = \frac{1}{2} \sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^H \boldsymbol{R}_{k,i} \boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^H \boldsymbol{\omega}_{k,i} + f_{1,2}(\boldsymbol{\omega}_{k,i}). \tag{4-20}$$

The gradient of the method in the negative direction includes a new term due to de addition f_1 and f_2, respectively, and is obtained as follows:

$$\boldsymbol{g}_{k,i}(j) = \boldsymbol{b}_{k,i} - \boldsymbol{R}_{k,i} \boldsymbol{\omega}_{k,i}(j) - \rho \mathrm{sgn}(\boldsymbol{\omega}_{k,i}) = -\nabla C_{CG}(\boldsymbol{\omega}_{k,i}), \tag{4-21}$$

$$\boldsymbol{g}_{k,i}(j) = \boldsymbol{b}_{k,i} - \boldsymbol{R}_{k,i} \boldsymbol{\omega}_{k,i}(j) - \rho \frac{\mathrm{sgn}(\boldsymbol{\omega}_{k,i})}{1 + \varepsilon \|\boldsymbol{\omega}_{k,i}\|_1} = -\nabla C_{CG}(\boldsymbol{\omega}_{k,i}), \tag{4-22}$$

Calculating the Krylov subspace [22] through different operations and using the penalty functions of equations (4-4) and (4-7), the recursion of the

parameter vector is given by

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j) - \rho\frac{\partial(f_{1,2})}{\partial\boldsymbol{\omega}^*_{k,i}}, \tag{4-23}$$

The step size α, the direction vector $\boldsymbol{p}_{k,i}$ and β parameter still remain with the MCG computations [22] [23] as follows:

$$\alpha(j) = \frac{\boldsymbol{g}^H_{k,i}(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}^H_{k,i}(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}, \tag{4-24}$$

$$\boldsymbol{p}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j), \tag{4-25}$$

$$\beta(j) = \frac{\boldsymbol{g}^H_{k,i}(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}^H_{k,i}(j-1)\boldsymbol{g}_{k,i}(j-1)}. \tag{4-26}$$

The update equations of CTA and ATC sparsity-aware CG as result of the derivation process are described as follows

$$\textit{CTA diffusion}\begin{cases} \boldsymbol{\varphi}_{k,i-1} = \sum_{l\epsilon N_k} a_{lk}\omega_{l,i-1} \\ \boldsymbol{\omega}_{k,i}(0) = \boldsymbol{\varphi}_{k,i-1} \\ \boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j) - \rho\frac{\partial(f_{1,2})}{\partial\boldsymbol{\omega}^*_{k,i}} \end{cases} \tag{4-27}$$

$$\textit{ATC diffusion}\begin{cases} \boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j) \\ \boldsymbol{\varphi}_{k,i} = \sum_{l\epsilon N_k} a_{lk}\omega_{l,i}(j_{last}) \\ \boldsymbol{\omega}_{k,i} = \boldsymbol{\varphi}_{k,i} - \rho\frac{\partial(f_{1,2})}{\partial\boldsymbol{\omega}^*_{k,i}}. \end{cases} \tag{4-28}$$

These sparsity-aware methods result in algorithms with faster convergence and lower MSD values as can be seen in the simulations in the following sections.

4.4.1 ZA and RZA Diffusion CG Algorithms

In both CTA and ATC variants the same steps presented in Section 4.3.1 for f_1 and f_2 are carried out, including their derivatives on the typical steps of each strategy. Tables 4.3 and 4.4 shows the sparsity-aware method for CTA and ATC strategies, respectively.

Table 4.3 – Sparsity-Aware CTA CG Algorithm

Parameter initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta \boldsymbol{I}$, $\boldsymbol{b}_{k,0} = 0$,...
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{\varphi}_{k,i} = \sum_{l \epsilon N_k} a_{lk} \omega_{l,i-1}$
    $\boldsymbol{\omega}_{k,i}(0) = \boldsymbol{\varphi}_{k,i}$
    $\boldsymbol{g}_{k,i}(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0)\boldsymbol{\omega}_{k,i-1}$
    $\boldsymbol{p}_{k,i}(0) = \boldsymbol{g}_{k,i}(0)$
    For each CG iteration $j = 1$ until convergence
      $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
      $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j)$
      $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j-1) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
      $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
      $\boldsymbol{p}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j-1)$
    End For
    $\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i}(j_{last}) - \rho \frac{\partial(f_{1,2})}{\partial \boldsymbol{\omega}_{k,i}^*}$
  End for
End for

Table 4.4 – Sparsity-Aware ATC CG Algorithm

Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$, $\boldsymbol{R}_{k,0} = \delta \boldsymbol{I}$, $\boldsymbol{b}_{k,0} = 0$
For each time instant $i > 0$
  For each agent k=1,2, ..., N
    $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
    $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
    $\boldsymbol{g}k,i(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0)\boldsymbol{\omega}_{k,i-1}(0)$
    $\boldsymbol{p}k,i(0) = \boldsymbol{g}_{k,i}(0)$
    For each CG iteration $j = 1$ until convergence
      $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
      $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j)$
      $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
      $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
      $\boldsymbol{p}_{k,i}(j+1) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j)$
    End For
    $\boldsymbol{\omega}_{k,i} = \sum_{l \epsilon N_k} a_{lk}\boldsymbol{\omega}_{l,i}(j_{last}) - \rho \frac{\partial(f_{1,2})}{\partial \boldsymbol{\omega}_{k,i}^*}$
  End for
End for

4.4.2
ZA and RZA Diffusion MCG Algorithms

For scenarios with several negligible weight values among a few large coefficients we developed the modified version of the CG algorithm for CTA and ATC protocols. Tables 4.5 and 4.6 show the main steps for both strategies. The ATC and CTA MCG algorithms are very similar as presented in the previous section, including the penalty functions. In the ATC strategy, the generation of the first state resulting from the adaptation step is used in the final update.

Table 4.5 – Sparsity-Aware CTA MCG Algorithm

$$
\begin{aligned}
&\boldsymbol{\varphi}_{k,i-1} = \textstyle\sum_{l\epsilon N_k} a_{lk}\boldsymbol{\omega}_{l,i-1} \\
&\boldsymbol{\omega}_{k,1} = \boldsymbol{\varphi}_{k,0} \\
&\alpha_{k,i} = \eta\frac{\boldsymbol{p}_{k,i}^H\boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i}}, \lambda - 0.5 \leq \eta \leq \lambda \\
&\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i-1} - \alpha_{k,i}\boldsymbol{p}_{k,i} - \rho\frac{\partial(f_{1,2})}{\partial\boldsymbol{\omega}_{k,i}^*} \\
&\boldsymbol{g}_{k,i} = \lambda\boldsymbol{g}_{k,i} - \alpha_{k,i}\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1} + \boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\varphi}_{k,i-1}^H\boldsymbol{x}_{k,i}] \\
&\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i}-\boldsymbol{g}_{k,i-1})^H\boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i-1}^H\boldsymbol{g}_{k,i-1}} \\
&\boldsymbol{p}_{k,i} = \boldsymbol{g}_{k,i} + \beta_{k,i}\boldsymbol{p}_{k,i-1}
\end{aligned}
$$

Table 4.6 – Sparsity-Aware ATC MCG Algorithm

$$
\begin{aligned}
&\alpha_{k,i} = \eta\frac{\boldsymbol{p}_{k,i}^H\boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i}}, \lambda - 0.5 \leq \eta \leq \lambda \\
&\boldsymbol{\varphi}_{k,i} = \boldsymbol{\omega}_{k,i-1} - \alpha_{k,i}\boldsymbol{p}_{k,i} \\
&\boldsymbol{g}_{k,i} = \lambda\boldsymbol{g}_{k,i} - \alpha_{k,i}\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1} + \boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H\boldsymbol{x}_{k,i}] \\
&\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i}-\boldsymbol{g}_{k,i-1})^H\boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i-1}^H\boldsymbol{g}_{k,i-1}} \\
&\boldsymbol{p}_{k,i} = \boldsymbol{g}_{k,i} + \beta_{k,i}\boldsymbol{p}_{k,i-1} \\
&\boldsymbol{\omega}_{k,i} = \textstyle\sum_{l\epsilon N_k} a_{lk}\boldsymbol{\varphi}_{l,i} - \rho\frac{\partial(f_{1,2})}{\partial\boldsymbol{\omega}_{k,i}^*}
\end{aligned}
$$

4.5
Computational Complexity

The computational complexity of the sparsity-aware proposed algorithms is shown on Table 4.7 in terms of additions and multiplications. Sparsity-aware CG versions have a similar computational complexity in terms of multiplications. The same situation applies for sparsity-aware MCG versions. The number of arithmetic operations of the modified versions are considerably lower than conventional variants. This occurs mainly due to the fact that the modified methods do not contain the CG internal iterations.

Table 4.7 – Computational Complexity of Sparsity-Aware Distributed CG Algorithms

Method	Additions	Multiplications
ZA-Consensus-CG	$3LM + LJ(M^2 + 4M - 2)$	$L(M^2 + 2M)$ $+LJ(3M^2 + 3M)$
ZA-CTA-CG	$L(M^2 + 3M)$ $+LJ(2M^2 + 6M - 3)$	$L(2M^2 + 5M)$ $+LJ(3M^2 + 4M - 1)$
ZA-ATC-CG	$L(M^2 + 4M - 1)$ $+LJ(M^2 + 6M - 3)$	$L(2M^2 + 4M)$ $+LJ(3M^2 + 4M - 1)$
RZA-Consensus-CG	$3LM + LJ(M^2 + 3M - 1)$	$L(M^2 + 2M)$ $+LJ(3M^2 + 3M)$
RZA-CTA-CG	$L(M^2 + 2M)$ $+LJ(2M^2 + 8M - 3)$	$L(2M^2 + 4M)$ $+LJ(3M^2 + 6M - 1)$
RZA-ATC-CG	$L(M^2 + 3M - 1)$ $+LJ(M^2 + 8M - 3)$	$L(2M^2 + 3M)$ $+LJ(3M^2 + 6M - 1)$
ZA-Consensus-MCG	$L(M^2 + 7M) - 2$	$L(4M^2 + 5M)$
ZA-CTA-MCG	$L(3M^2 + 10M - 4)$	$L(4M^2 + 10M - 1)$
ZA-ATC-MCG	$L(4M^2 + 10M - 3)$	$(6M^2 + 9M - 1)$
RZA-Consensus-MCG	$L(M^2 + 6M - 1)$	$L(4M^2 + 5M)$
RZA-CTA-MCG	$L(3M^2 + 11M - 4)$	$L(4M^2 + 11M - 1)$
RZA-ATC-MCG	$L(4M^2 + 11M - 3)$	$L(6M^2 + 10M - 1)$

Establishing a comparison between sparsity-aware CG algorithms and non-sparse CG algorithms as proposed in the previous chapter, it can be appreciated that sparsity-aware versions add a few terms on the operations, but the level of the complexity is still in the same order $O(M^2)$ for all protocols. Specifically the RZA Consensus MCG represents the simplest method with a few less terms, but remains with a quadratic order as the other variants, and requires the storage of the same amount of data.

4.6 Simulations Results

In this section we evaluate the proposed standard sparsity-aware distributed conventional and modified CG algorithms. The results are based on the mean square deviation (MSD) of the network. We also present the results compared with sparsity-aware distributed LMS and RLS algorithms. We consider a network with 20 nodes and 1000 iterations per run. Each iteration corresponds to a time instant. The results are averaged over 50 experiments. The length of the filter is 10 and the variance of the input signal 1, which has been modeled as a complex Gaussian noise with a variance of 0.001.

4.6.1
Comparison between sparsity-aware distributed CG algorithms

Two entries are set to one and the remaining values are set to zero, giving a sparsity level of 2/10. After all the iterations, the performance of all consensus and diffusion sparsity-aware methods in terms of MSD is shown in the following figures.

The results in figures 4.1, 4.2 and 4.3 show that the sparsity-aware versions outperform the standard versions and the best results are obtained for the RZA versions. At the same time the MCG algorithms have a better performance than the standard ones.

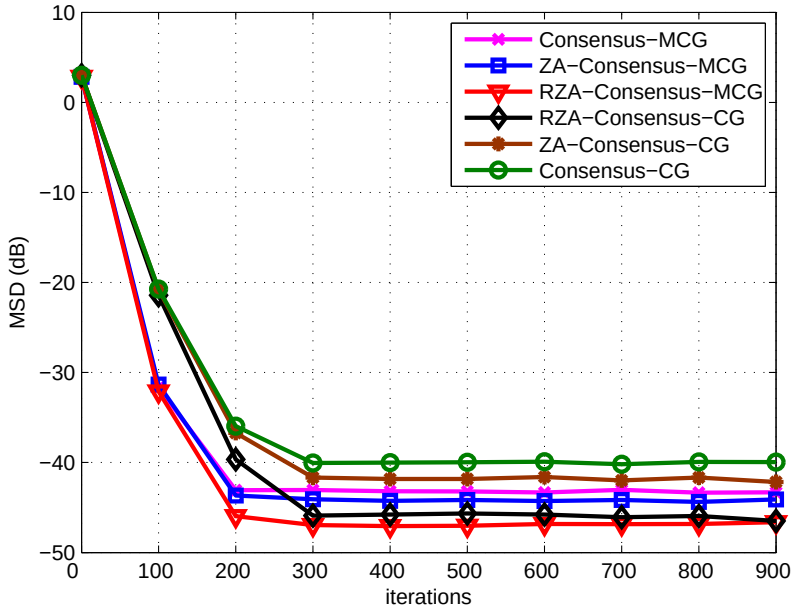

Figure 4.1 – MSD of the network for distributed consensus standard and sparsity-aware CG versions with $\lambda = 0.99$, $\rho_{ZA} = 0.5 \times 10^{-4}$, $\varepsilon = 0.1$, $\rho_{RZA} = 0.2 \times 10^{-3}$, $\delta = 10^{-2}$, $J_{CCG} = 5$, $S = 2/10$. For modified versions $\lambda = 0.95$, $\eta = 0.55$, $\rho_{ZA} = 0.5 \times 10^{-4}$, $\varepsilon = 0.1$, $\rho_{RZA} = 0.2 \times 10^{-3}$.

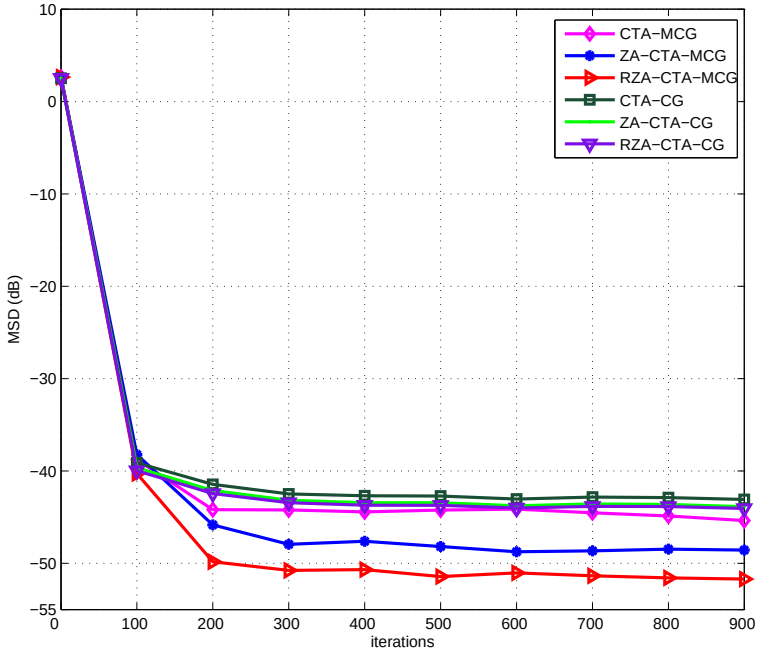

Figure 4.2 – MSD of the network for distributed CTA standard and sparsity-aware CG versions with $\lambda = 0.99$, $\rho_{ZA} = 0.5 \times 10^{-4}$, $\varepsilon = 0.1$, $\rho_{RZA} = 10^{-3}$, $\delta = 10^{-2}$, $J_{CCG} = 5$, $S = 2/10$. For modified versions $\lambda = 0.95$, $\eta = 0.55$, $\rho_{ZA} = 0.7 \times 10^{-4}$, $\varepsilon = 0.1$, $\rho_{RZA} = 0.2 \times 10^{-3}$.

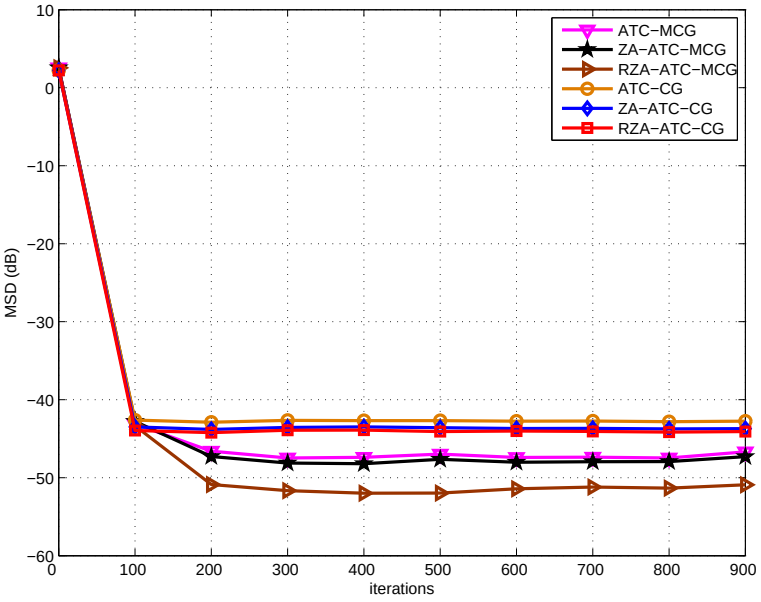

Figure 4.3 – MSD of the network for distributed ATC standard and sparsity-aware CG versions with $\lambda = 0.99$, $\rho_{ZA} = 0.5 \times 10^{-4}$, $\varepsilon = 0.1$, $\rho_{RZA} = 0.9 \times 10^{-3}$, $\delta = 10^{-2}$, $J_{CCG} = 5$, $S = 2/10$. For modified versions $\lambda = 0.95$, $\eta = 0.55$, $\rho_{ZA} = 0.2 \times 10^{-4}$, $\varepsilon = 0.1$, $\rho_{RZA} = 0.2 \times 10^{-3}$.

It can be observed that the RZA-ATC-MCG algorithm has a faster convergence and a lower MSD value as compared to ZA-ATC-MCG, CTA-MCG and consensus sparsity-aware strategies. Due to the reweighted regularization the RZA-ATC-MCG outperforms the ZA-ATC-MCG. In general, diffusion strategies outperform consensus approaches because they are able to include additional information into their processing steps generating an intermediate variable, that latter is used in the final update. Regarding the comparison between consensus and diffusion, it is also known that diffusion gives freedom to the adaptation and consensus forces the algorithms to converge to a common parameter vector. In CTA and ATC variants the order of the steps plays an important roll on the performance of the algorithm.

4.6.2
Comparison between different sparsity levels

Different sparsity levels S were considered for the proposed algorithms. Figure 4.4 shows the MSD behavior of the RZA-ATC-MCG algorithm for different values of S.

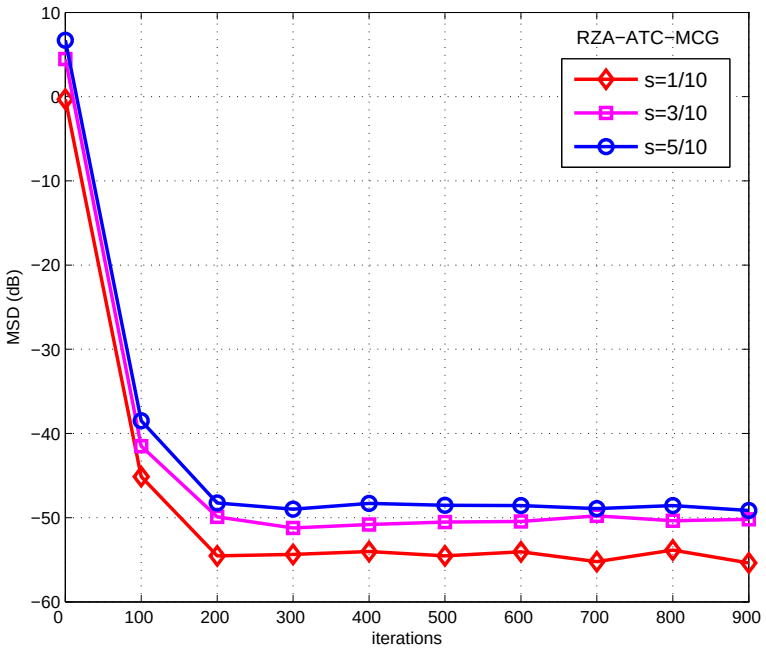

Figure 4.4 – MSD of the network for distributed RZA-ATC-MCG with $\lambda = 0.95$, $\eta = 0.55$, $\rho_{ZA} = 0.2 \times 10^{-4}$, $\varepsilon = 0.1$, $\rho_{RZA} = 0.2 \times 10^{-3}$.

It can be noticed in Figure 4.4 that if the number of nonzero values is increased the algorithm will take longer to converge and the deviation will be larger as compared to a sparse system with a lower number of nonzero values.

4.6.3
Comparison between sparsity-aware distributed MCG and RLS algorithms

The proposed algorithms were also compared with the distributed versions of the RLS algorithm. Figure 4.5 shows the MSD of a network where the RZA diffusion RLS [12] and MCG version were tested.

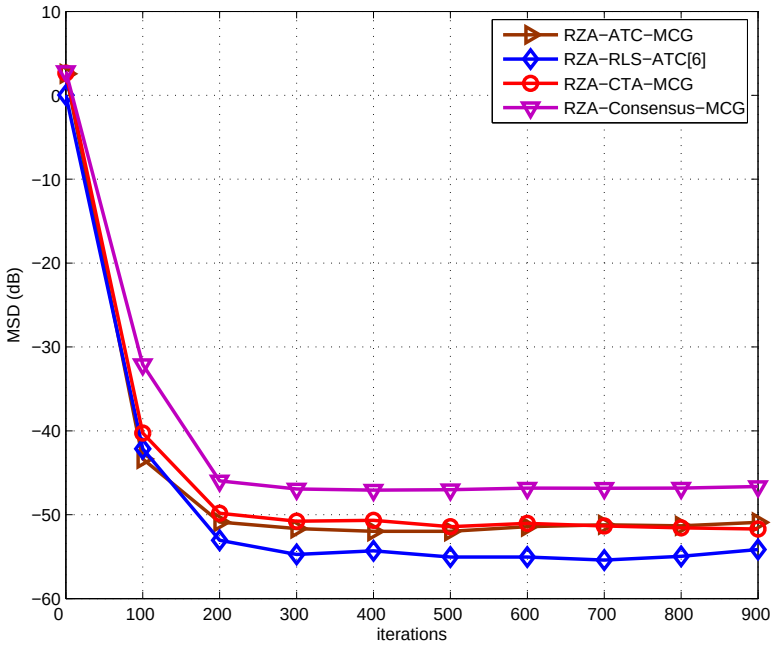

Figure 4.5 – MSD of the network number for consensus and distributed diffusion RZA CG versions with $\lambda_{Cons} = 0.99$, $\rho_{Consensus} = 0.5 \times 10^{-3}$, $\varepsilon = 0.1$, $\delta = 10^{-1}$, $\lambda_{CTA} = 0.99$, $\rho_{CTA} = 0.2 \times 10^{-3}$, $\varepsilon = 0.1$, $\gamma = 10^{-2}$, $\lambda_{ATC} = 0.99$, $\rho_{ATC} = 0.3 \times 10^{-3}$, $\varepsilon = 0.1$, $\delta = 10^{-1}$, $\eta = 0.55$, $s = 2/10$, $\lambda_{RLS} = 0.98$, $\rho_{RLS} = 0.5 \times 10^{-4}$, $\varepsilon = 0.1$, $\delta = 10^{-2}$.

4.7
Summary

In this chapter we have proposed sparsity-aware distributed CG algorithms for parameter estimation. As well as in chapter 3 the diffusion CG algorithms, in this case exploiting sparsity, outperform consensus strategy. In all cases, the modified versions obtained the lowest MSD values and fastest convergence rates. The MCG algorithm with diffusion protocols has a very close performance to the RLS algorithm in terms of convergence rate. Simulations have shown that the proposed distributed CG algorithms are suitable techniques for adaptive parameter estimation in presence of sparse systems.

5
Distributed Estimation Based on Alternating Mixed Discrete-Continuous Adaptation

5.1
Introduction

Many studies have shown that exploiting the sparsity of a system is beneficial to enhancing the performance of a signal processing algorithm [1]. Most of the studies developed for distributed processing exploiting sparsity focus on the least-mean square (LMS), recursive least-squares (RLS) and conjugate gradient (CG) algorithms using different penalty functions [9]-[12], [16]-[19]. These penalty functions perform a regularization that attracts to zero the coefficients of the parameter vector that are close to zero. The most well-known and popular penalty functions are the l_0-norm, the l_1-norm and the log-sum [9], [10].

The optimal algorithm for processing sparse signals and systems is known as the oracle algorithm[16]. It can identify the positions of the non-zero coefficients and fully exploit the sparsity of the system under consideration. The oracle algorithm also requires the computation of the optimal filter, which is a continuous optimization problem [19].

With the development and increasing deployment of mobile networks the frequency spectrum for the transmission of information has become a resource that should be exploited in a wise way to avoid signal interference. By spectral estimation in sensor networks this resource can be planned and properly exploited.

Diffusion adaptation strategies incorporating sparsity constraints have been used to solve distributed spectrum estimation problems in [17][18]. Prior work on distributed oracle method is rather limited and techniques that exploit possible sparsity of the signals using discrete and continuous variables have not been developed so far.

In this chapter we propose a DAMDC-LMS algorithm based on alternating and mixed optimization of continuous and discrete values. Specifically,we develop distributed LMS algorithms using the diffusion ATC protocol. The

proposed method is compared with recently reported algorithms in the literature. The application scenarios in this work are parameter estimation and spectrum estimation over sensor networks, which are fields of practical interest nowadays.

This chapter is organized as follows. Section 5.2 describes the system model and the problem statement. Section 5.3 presents the proposed DAMDC-LMS algorithm. Section 5.4 details the proposed algorithm for an application to spectrum estimation. Section 5.5 presents and discusses the simulation results. Finally, Section 5.6 gives the conclusions of this chapter.

5.2 System Model and Problem Statement

In this section, is presented the distributed estimation model that we followed. The network scheme and the problem statement are established.

5.2.1 System Model

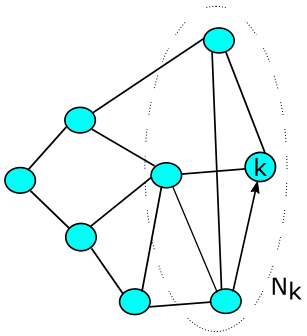

Figure 5.1 – Network topology with N nodes.

The network consists of N nodes that exchange information between them, where each node represents an adaptive parameter vector with neighborhood described by the set N_k, as shown in Figure 5.1. The main task of parameter estimation is to adjust the unknown $M \times 1$ weight vector $\boldsymbol{\omega}_k$ of each node, where M is the length of the parameter vector [1]. The desired signal $d_{k,i}$ at each time i at node k is drawn from a random process and given by

$$d_{k,i} = \boldsymbol{\omega}_0^H \boldsymbol{x}_{k,i} + n_{k,i}, \tag{5-1}$$

where $\boldsymbol{\omega}_0$ is the $M \times 1$ system weight vector, $\boldsymbol{x}_{k,i}$ is the $M \times 1$ input signal vector and $n_{k,i}$ is the measurement noise, which is modelled by a complex Gaussian

random variable with zero mean and covariance σ_n^2. The output estimate is given by

$$\begin{aligned} \hat{d}_{k,i} &= \boldsymbol{\omega}_{k,i}^H \boldsymbol{P}_{k,i} \boldsymbol{x}_{k,i} = \boldsymbol{p}_{k,i}^T \boldsymbol{W}_{k,i}^* \boldsymbol{x}_{k,i} \\ &= \boldsymbol{x}_{k,i}^T \boldsymbol{W}_{k,i}^* \boldsymbol{p}_{k,i} = \boldsymbol{x}_{k,i}^T \boldsymbol{P}_{k,i} \boldsymbol{\omega}_{k,i}^*, \end{aligned} \tag{5-2}$$

where $\boldsymbol{W}_{k,i} = diag(\boldsymbol{\omega}_{k,i})$. $\boldsymbol{P}_{k,i}$ is a square diagonal matrix of M elements that is applied to the input vector $\boldsymbol{x}_{k,i}$ and is supposed to perform the oracle algorithm by identifying the null positions of the parameter vector.

The main goal of the network is to minimize the following cost function:

$$C(\boldsymbol{P}_{k,i}, \boldsymbol{\omega}_{k,i}) = \sum_{k=1}^{N} E[|d_{k,i} - \hat{d}_{k,i}|^2]. \tag{5-3}$$

By solving this minimization problem it is possible to obtain the optimum solution of the weight vector at each node.

5.2.2 Problem Statement

We consider a diffusion ATC algorithm for a network, where each agent k has access at each time instant to a realization of the zero-mean spatial data $\{d_{k,i}, \boldsymbol{x}_{k,i}\}$[9]. For a network with possibly sparse parameter vectors, the cost function also involves a penalty function which exploits sparsity. In this case the network needs to solve the following optimization problem:

$$\min C(\boldsymbol{P}_{k,i}, \boldsymbol{\omega}_{k,i}) = \sum_{k=1}^{N} E[|d_{k,i} - \boldsymbol{\omega}_{k,i}^H \boldsymbol{P}_{k,i} \boldsymbol{x}_{k,i}|^2]. \tag{5-4}$$

In the following sections we focus on a novel distributed diffusion approach to the oracle algorithm to solve (5-4).

5.3 Proposed DAMDC-LMS Algorithm

In this section, we present the proposed distributed scheme algorithm using the diffusion strategy. We first detail how the proposed algorithm and then consider its combination with the diffusion protocol.

5.3.1 Derivation of the DAMDC-LMS algorithm

The proposed scheme for each agent k of the network previously represented is shown below in Figure 5.2. To obtain the recursion for $\boldsymbol{P}_i$ and $\boldsymbol{\omega}_i$ it is necessary to compute the stochastic gradient of both parameters, where the optimization of $\boldsymbol{P}_i$ involves discrete variables and $\boldsymbol{\omega}_i$ deals with continuous variables. Our goal is to update the system parameter vector $\boldsymbol{\omega}_i$ regardless of the adaptive algorithm. In this work we develop an alternating optimization approach using an LMS type algorithm that consists of a recursion for $\boldsymbol{P}_i$ and another recursion for $\boldsymbol{\omega}_i$.

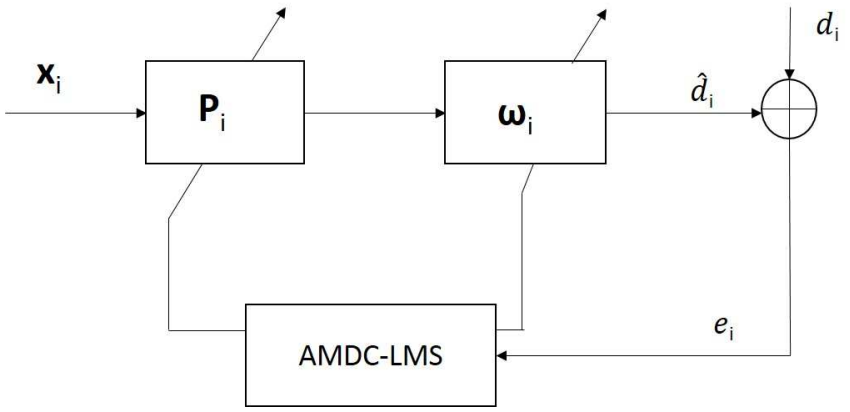

Figure 5.2 – Proposed scheme and adapting algorithm.

The first adaptive parameter matrix $\boldsymbol{P}_i$ is a quadratic matrix that is applied to $\boldsymbol{x}_i$ input data vector and is a diagonal matrix of the column vector $\boldsymbol{p}_i$. This matrix aims to approach the oracle algorithm by identifying the null positions of the parameter vector. The second adaptive parameter vector $\boldsymbol{\omega}_i$ is a column vector of M coefficients that typically performs the system identification.

In order to compute the parameter matrix $\boldsymbol{P}_i$ we must compute the gradient of the MSE cost function given by

$$C(\boldsymbol{P}_i, \boldsymbol{\omega}_i) = \sum_{k=1}^{N} E[|d_i - \boldsymbol{\omega}_i^H \boldsymbol{P}_i \boldsymbol{x}_i|^2]. \tag{5-5}$$

that is equivalent to

$$C(\boldsymbol{p}_i, \boldsymbol{\omega}_i) = \sum_{k=1}^{N} E[|d_i - \boldsymbol{p}_i^T \boldsymbol{W}_i^H \boldsymbol{x}_i|^2]. \tag{5-6}$$

where the parameter vector $\boldsymbol{p}_i$ is assumed to be real-valued parameter vector.

The gradient of the cost function is then given by

$$\begin{aligned}\nabla_{\boldsymbol{p}_i} C(\boldsymbol{p}_i, \boldsymbol{\omega}_i) = \tfrac{\partial}{\partial \boldsymbol{p}_i}(E|d_i|^2 - (\boldsymbol{p}_i^T \boldsymbol{W}_i^* E[d_i^* \boldsymbol{x}_i]) \\ -E[d_i \boldsymbol{x}_i^H]\boldsymbol{W}_i \boldsymbol{p}_i + \boldsymbol{p}^T i \boldsymbol{W}_i^* E[\boldsymbol{x}_i \boldsymbol{x}_i^H \boldsymbol{W}_i \boldsymbol{p}_i]).\end{aligned} \tag{5-7}$$

Replacing the expected values with instantaneous values, we obtain

$$\begin{aligned}\hat{\nabla}_{\boldsymbol{p}_i} C(\boldsymbol{p}_i, \boldsymbol{\omega}_i) = \tfrac{\partial}{\partial \boldsymbol{p}_i}(|d_i|^2 - (\boldsymbol{p}_i^T \boldsymbol{W}_i^* [d_i^* \boldsymbol{x}_i]) \\ -[d_i \boldsymbol{x}_i^H]\boldsymbol{W}_i \boldsymbol{p}_i + \boldsymbol{p}_i^T \boldsymbol{W}_i^* [\boldsymbol{x}_i \boldsymbol{x}_i^H \boldsymbol{W}_i \boldsymbol{p}_i]).\end{aligned} \tag{5-8}$$

Computing the gradient of the cost function with respect $\boldsymbol{p}_i$, we obtain

$$\begin{aligned}\hat{\nabla}_{\boldsymbol{p}_i} C(\boldsymbol{p}_i, \boldsymbol{\omega}_i) = d_i^* \boldsymbol{W}_i^* \boldsymbol{x} i - d_i \boldsymbol{W}_i^T \boldsymbol{x}_i^* \\ +\boldsymbol{W}_i^* \boldsymbol{x}_i \boldsymbol{x}_i^H \boldsymbol{W}_i \boldsymbol{p}_i + \boldsymbol{W}_i^T \boldsymbol{x}^* i \boldsymbol{x}_i^T \boldsymbol{W}_i^H \boldsymbol{p}_i.\end{aligned} \tag{5-9}$$

Grouping common terms, we arrive at

$$\begin{aligned}\hat{\nabla}_{\boldsymbol{p}_i} C(\boldsymbol{p}_i, \boldsymbol{\omega}_i) = -([d_i - \boldsymbol{x}_i^H \boldsymbol{W}(i) \boldsymbol{p}_i]\boldsymbol{W}_i \boldsymbol{x}_i \\ +[d_i - \boldsymbol{x}_i^T \boldsymbol{W}_i^H \boldsymbol{p}_i]\boldsymbol{W}_i^T \boldsymbol{x}_i^*),\end{aligned} \tag{5-10}$$

where $\boldsymbol{p}_i$ is a real parameter vector, $\boldsymbol{p}_i = \boldsymbol{p}_i^*$, $\boldsymbol{p}_i^H = [\boldsymbol{p}_i^*)]^T = \boldsymbol{p}_i^T$. The transpose of the diagonal matrix $\boldsymbol{W}_i$ is equal to the original one, i.e., $\boldsymbol{W}_i^T = \boldsymbol{W}_i$, then $\boldsymbol{W}_i^H = [\boldsymbol{W}_i^*]^T = [\boldsymbol{W}_i^T]^* = \boldsymbol{W}_i^*$. The terms in (5-10) represent the sum of one vector and its conjugate:

$$\begin{aligned}\hat{\nabla}_{\boldsymbol{p}_i} C(\boldsymbol{p}_i, \boldsymbol{\omega}_i) = -(\underbrace{[d_i - \boldsymbol{x}_i^H \boldsymbol{W}_i \boldsymbol{p}_i]\boldsymbol{W}_i^* \boldsymbol{x}_i}_{A} \\ + \underbrace{[d_i - \boldsymbol{x}_i^T \boldsymbol{W}_i^H \boldsymbol{p}_i]\boldsymbol{W}_i \boldsymbol{x}_i^*)}_{A^*}.\end{aligned} \tag{5-11}$$

Applying the property $A + A^* = 2\Re(A)$, we have

$$\hat{\nabla}_{\boldsymbol{p}_i} \mathrm{MSE}(\boldsymbol{p}_i, \boldsymbol{\omega}_i) = 2\Re(A). \tag{5-12}$$

The recursion to update the parameter vector $\boldsymbol{p}_i$ is given by

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i - \eta \hat{\nabla}_{\boldsymbol{p}_i} \mathrm{MSE}(\boldsymbol{p}_i, \boldsymbol{W}_i), \tag{5-13}$$

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i + 2\eta \Re(e_{p_i}^* \boldsymbol{x}_i^H \boldsymbol{W}_i), \tag{5-14}$$

where e_p is an error signal of the vector $\boldsymbol{p}$, given by

$$e_{p_i} = d_i - \boldsymbol{p}_i^T \boldsymbol{W}_{i-1} \boldsymbol{x}_i. \tag{5-15}$$

For the parameter vector update, we can apply well-known adaptive algorithms like LMS, conjugate gradient and recursive least-squares. By computing the gradient of the cost function with respect to $\boldsymbol{w}_i^*$, we have

$$\hat{\nabla} C_{\boldsymbol{w}_i^*}(\boldsymbol{p}_i, \boldsymbol{w}_i) = (d_i - \boldsymbol{x}_i^T \boldsymbol{P}_i \boldsymbol{\omega}_{i-1}^*)^* \boldsymbol{P}_i \boldsymbol{x}_i \tag{5-16}$$

The following LMS type recursion updates the parameter vector $\boldsymbol{\omega}_i$ as described by

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i + \mu e_i^* \boldsymbol{P}_i \boldsymbol{x}_i, \tag{5-17}$$

where the error signal is given by $e_i = d_i - \boldsymbol{x}_i^T \boldsymbol{P}_i \boldsymbol{\omega}_{i-1}^*$.

5.3.2 Diffusion DAMDC-LMS algorithm

Based on the previous derivation steps we develop the proposed algorithm for a distributed environment using a diffusion strategy, where all the agents of the network perform similar steps to update their weights [10].

In diffusion protocols there are two well-known variants that differ in the choice of the output variable (which can be either the output of the combination or of the adaptation steps). The variants are named Adapt-then-Combine (ATC) and Combine then-Adapt (CTA). Both perform adaptation and learning at the same time [1]-[6].

The recursions for $\boldsymbol{p}_{k,i}$ and $\boldsymbol{\omega}_{k,i}$ using the ATC variant are given by

$$\boldsymbol{p}_{k,i+1} = \boldsymbol{p}_{k,i} + 2\eta \Re(e_{p_{k,i}}^* \boldsymbol{x}_{k,i}^H \boldsymbol{W}_{k,i}) \tag{5-18}$$

$$\boldsymbol{\varphi}_{k,i+1} = \boldsymbol{\omega}_{k,i-1} + \mu e_{k,i}^* \boldsymbol{P}_{k,i} \boldsymbol{x}_{k,i}, \tag{5-19}$$

$$\boldsymbol{\omega}_{k,i} = \sum_{l \in N_k} a_{lk} \boldsymbol{\varphi}_{l,i}, \tag{5-20}$$

where (5-18) and (5-19) are the adaptation steps, and (5-20)the combination step of the ATC protocol. The combining coefficients of the data fusion are represented by a_{lk} and should comply with

$$\sum_{l \epsilon N_k} a_{lk} = 1, \ l \epsilon N_{k,i}, \forall k. \tag{5-21}$$

The strategy adopted, as in previous chapters, for the a_{lk} combiner, is the Metropolis rule [1]. given by

In order to devise the discrete vector $\boldsymbol{p}_{k,i}$ at each node the initial value is an all-one ($\boldsymbol{p}_{k,0} = 1$ or $\boldsymbol{P}_{k,0} = \boldsymbol{I}$). The parameter vector is initialized as all-

zero vector ($\boldsymbol{\omega}_{k,0} = 0$ or $\boldsymbol{W}_{k,0} = 0$). At each iteration the vector $\boldsymbol{p}_{k,i}$ assumes different values at each position m, $[p_{k,i}^0, p_{k,i}^1, ..., p_{k,i}^{M-1}]$, which are generally measured around 1. To obtain the discrete values to approach the oracle algorithm, we adopted the following rule:

$$p_{k,i+1}^m = \begin{cases} 1, \text{ if } p_{k,i}^m > 1 \\ 0, \text{ otherwise.} \end{cases} \tag{5-22}$$

where 1 is the treshold used to determine the position of the non zero values of the parameter vector. The goal is to obtain with this rule the oracle vector $\boldsymbol{p}_{k,i}$.

5.4 Distributed Spectrum Estimation using DAMDC-LMS Algorithm

In distributed spectrum estimation, we aim to estimate the spectrum of a transmitted signal $\boldsymbol{s}$ with N nodes using a wireless sensor network [18]. The power spectral density (PSD) of the signal $\boldsymbol{s}$ at each frequency denoted by $\boldsymbol{\Phi}_s(f)$ is given by

$$\boldsymbol{\Phi}_s(f) = \sum_{m=1}^{B} b_m(f)\omega_{0m} = \boldsymbol{b}_0^T(f)\boldsymbol{\omega}_0, \tag{5-23}$$

where $\boldsymbol{b}_0(f) = [b_1(f), ..., b_B(f)]^T$ is the vector of basis functions evaluated at frequency f, $\boldsymbol{\omega}_0 = [\omega_{01}, ..., \omega_{0B}]$ is a vector of weighting coefficients representing the power that transmits the signal $\boldsymbol{s}$ over each basis, and B is the number of basis functions. For B sufficiently large, the basis expansion in (5-23) can well approximate the frequency spectrum. Possible choices for the set of basis functions $b_m(f)_{m=1}^B$ include: rectangular functions, raised cosines, Gaussian bells and splines [17], [18].

The channel transfer function between a transmit node conveying the signal $\boldsymbol{s}$ and receive node k at time instant i is denoted by $\boldsymbol{H}_k(f, i)$, then the PSD of the received signal observed by node k can be expressed as

$$\begin{aligned} \boldsymbol{\Phi}_k(f, i) &= |H_k(f, i)|^2 \boldsymbol{\Phi}_s(f) + \upsilon_{n,k}^2, \\ &= \textstyle\sum_{m=1}^{B} |H_k(f, i)|^2 b_m(f)\omega_{0m} + \upsilon_{n,k}^2, \\ &= \boldsymbol{b}_{k,i}^T(f)\boldsymbol{\omega}_{0m} + \upsilon_{n,k}^2. \end{aligned} \tag{5-24}$$

where $\boldsymbol{b}_{k,i}^T(f) = [|H_k(f, i)|^2 b_m(f)]_{m=1}^B$ and $\upsilon_{n,k}^2$ is the receiver noise power at node k.

Following the distributed model, at every iteration i every node k measures the PSD $\boldsymbol{\Phi}_k(f,i)$ presented in (5-24) over N_c frequency samples $f_j = f_{min} : (f_{max} - f_{min})/N_c : f_{max}$, for $j = 1, ..., N_c$, the desired signal is given by

$$d_{k,i}(j) = \boldsymbol{b}_{k,i}^T(f_j)\boldsymbol{\omega}_0 + \upsilon_{n,k}^2 + n_{k,i}(j). \tag{5-25}$$

where the last term denotes the observation Gaussian noise with zero mean and variance $\sigma_{n,j}^2$. The receiver noise power $\upsilon_{n,k}^2$ can be estimated with high accuracy in a preliminary step using, e.g., an energy estimator over an idle band, and then subtracted from (5-25) [18]. A linear model is obtained from the measurements over N_c contiguous channels:

$$\boldsymbol{d}_{k,i} = \boldsymbol{B}_{k,i}\boldsymbol{\omega}_0 + \boldsymbol{n}_{k,i}, \tag{5-26}$$

where $\boldsymbol{B}_{k,i} = [\boldsymbol{b}_{k,i}^T(f_j)]_{j=1}^{N_c} \in \Re^{N_c \times B}$, and $\boldsymbol{n}_{k,i}$ is a zero mean random vector. Then we can establish the cost function for each agent k.

$$C(\boldsymbol{\omega}_{k,i}) = E[|\boldsymbol{d}_{k,i} - \boldsymbol{B}_{k,i}\boldsymbol{\omega}_0|^2], \tag{5-27}$$

Once we have the cost function, the AMDC-LMS algorithm can be developed by introducing the discrete parameter vector $\boldsymbol{p}_{k,i}$ in (5-27), which results in the following cost function:

$$C(\boldsymbol{\omega}_{k,i}) = E[|\boldsymbol{d}_{k,i} - \boldsymbol{B}_{k,i}\boldsymbol{P}_{k,i}\boldsymbol{\omega}_0|^2], \tag{5-28}$$

where $\boldsymbol{P}_{k,i}$ is the $B \times B$ diagonal matrix to exploit the sparsity for a more accurate spectrum estimation. Introducing the matrix $\boldsymbol{P}_{k,i}$ term for sparsity in recursions (5-18), (5-19), (5-20) we obtain

$$Adaptation \begin{cases} \boldsymbol{p}_{k,i+1} = \boldsymbol{p}_{k,i} + 2\eta\Re(e_{p_{k,i}}^*, B_{k,i}^H \boldsymbol{W}_{k,i-1}) \\ \boldsymbol{\varphi}_{k,i+1} = \boldsymbol{\omega}_{k,i-1} + \mu e_{k,i}^* \boldsymbol{P}_{k,i}\boldsymbol{B}_{k,i}, \end{cases} \tag{5-29}$$

$$Combination \begin{cases} \boldsymbol{\omega}_{k,i} = \sum_{l \in N_k} a_{lk}\boldsymbol{P}_{k,i}\boldsymbol{\varphi}_{l,i}. \end{cases} \tag{5-30}$$

Once $\boldsymbol{p}_{k,i}$ is obtained, the rule in (5-22) is applied to obtain de discrete values 0 and 1. The position in the vector with ones (1) indicates the information content of the signal at each node and entry of $\boldsymbol{x}_{k,i}$. With this method it is possible to achieve a similar behavior to the oracle algorithm identifying after a few iterations the positions of the non-zero coefficients of the sparse system.

5.5 Simulation Results

In this section, we evaluate the performance of the proposed DAMDC-LMS algorithm in two scenarios: distributed estimation and distributed spectrum estimation in wireless sensor networks. DAMDC-LMS is also compared with existing algorithms. The results are based on the mean square deviation (MSD) and power spectrum density estimation of the network.

5.5.1 Distributed Estimation

In this scenario, we consider a network with 20 nodes and 1000 iterations per run. Each iteration corresponds to a time instant. The results are averaged over 100 experiments. The length of the filter is 20 and the input signal, a complex white Gaussian noise, with zero mean unit variance. Different scenarios were tested to evaluate the MSD of the algorithm as compared with others. The system parameter vector was randomly set with two values different from zero.The SNR is set to $30dB$.After all the iterations, the performance of each algorithm in terms of MSD is shown in Figure 5.3. The results show that the proposed DAMDC algorithm outperforms the standard version as well as sparsity-aware RZA versions, and has a close performance to the oracle algorithm.

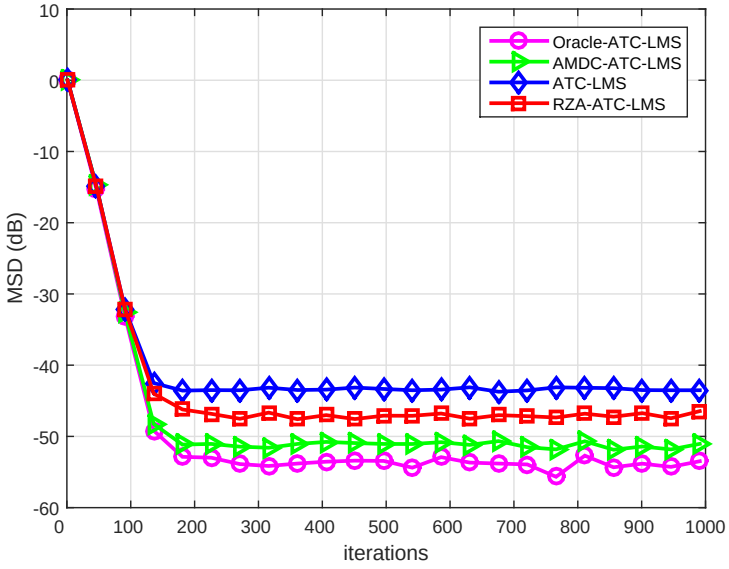

Figure 5.3 – MSD of the network for distributed ATC standard, RZA, oracle and DAMDC-LMS algorithms with $\mu = 0.45$, $\eta = 10^{-6}$, $\rho_{RZA} = 5 \times 10^{-4}$, $\varepsilon = 0.1$, $SNR = 30dB$, $S = 2/20$.

For an SNR equal to $40dB$ maintaining the parameter vector with two values different from zero all the algorithms take more time to converge. The DAMDC algorithm achieves also lower MSD than conventional and RZA versions. Figure 5.4 shows how increasing the SNR both DAMDC and oracle algorithms obtain a considerable difference compared with the standard and sparse RZA versions.

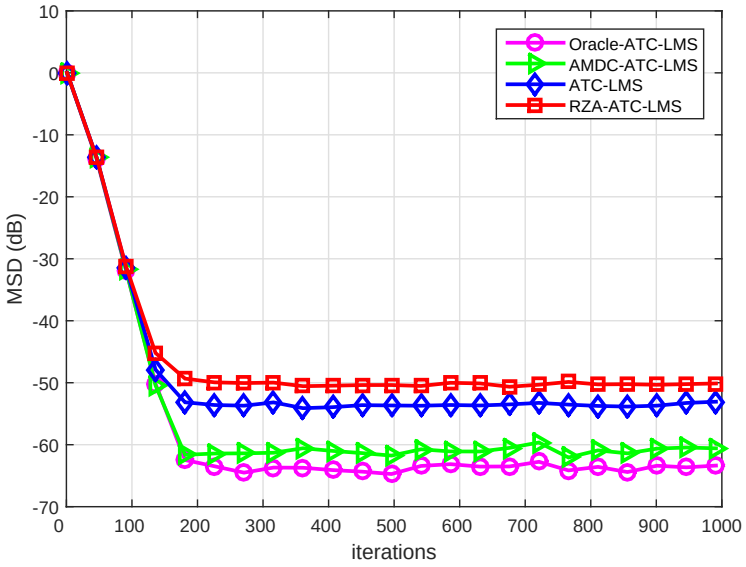

Figure 5.4 – MSD of the network for distributed ATC standard, RZA, oracle and DAMDC-LMS algorithms with $\mu = 0.45$, $\eta = 10^{-6}$, $\rho_{RZA} = 5 \times 10^{-4}$, $\varepsilon = 0.1$, $SNR = 40dB$, $S = 2/20$.

The DAMDC algorithm was evaluated modifying the sparsity of the system. For a system with 10 values different from zeros the DAMDC algorithm achieves the performance shown in Figure 5.5. In this case the DAMDC algorithm still with a very close performance to the oracle method but takes more time to converge. In general as well as the previous sparsity-aware algorithms, the DAMDC has a better performance for fewer non zero values of the parameter vector.

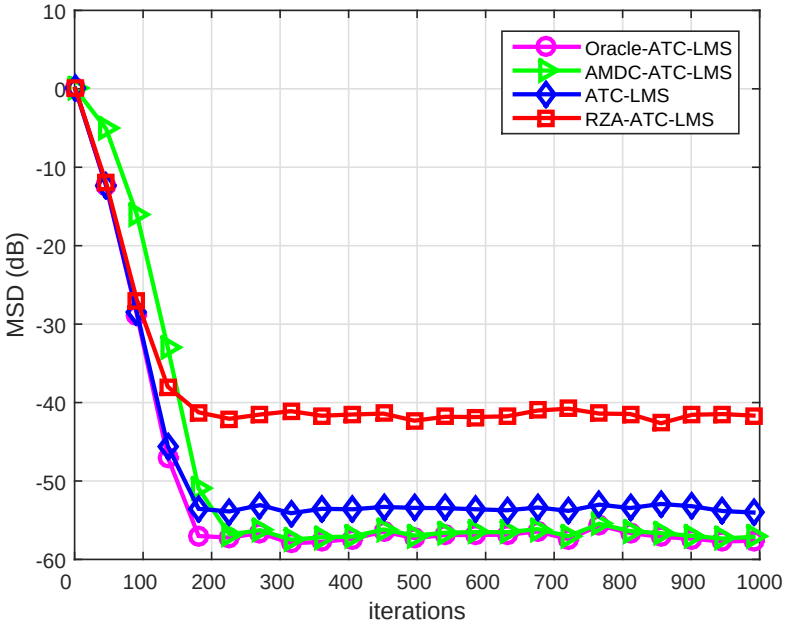

Figure 5.5 – MSD of the network for distributed ATC standard, RZA, oracle and DAMDC-LMS algorithms with $\mu = 0.45$, $\eta = 10^{-6}$, $\rho_{RZA} = 5 \times 10^{-4}$, $\varepsilon = 0.1$, $SNR = 40dB$, $S = 10/20$.

To evaluate the performance of the proposed technique in Figure 5.6 we also simulated the behavior of the steady-state network Mean Square Deviation (MSD), obtained at convergence by different algorithms, versus the signal to noise ratio (SNR). In this case the SNR was evaluated from 0 to 30 dB. Values are obtained by averaging over 10 independent experiments and over 200 time samples after convergence. The simulation considers all the parameters defined in the previous scenario. As can be seen in Figure 5.4, the estimation performance gets worse as the SNR decreases. Since the parameter vector to be estimated is sufficiently sparse (i.e., 2/20)and the SNR increases, the ADMC-LMS leads to similar performance as the oracle algorithm.

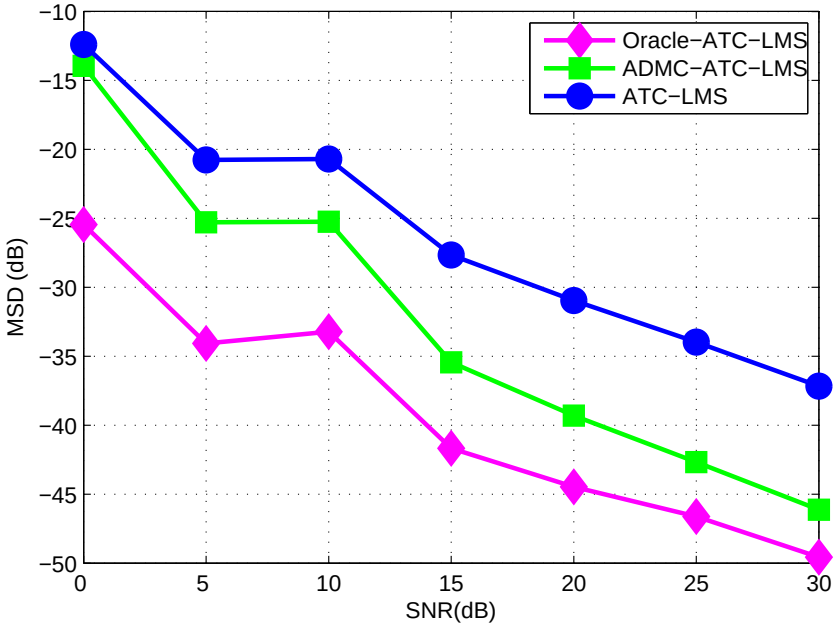

Figure 5.6 – MSD vs SNR, for different ATC-LMS algorithms.

5.5.2 Distributed Spectrum Estimation

In this scenario, we also consider a network of 20 nodes estimating the MSD and the unknown spectrum $\boldsymbol{\omega}_0$. The nodes scan 100 frequencies over the frequency axis, which is normalized between 0 and 1, and use $B = 50$ non−overlapping rectangular basis functions to model the expansion of the spectrum [29]. The basis functions have amplitude equal to one. Different values of sparsity were set to evaluate the behaviour of the MSD for spectrum estimation. For a sparsity equal to 20/50 we have obtained the performance shown below in Figure 5.7. A better performance is obtained for a low sparsity ratio with 8 non zero values. Figure 5.8 as well as Figure 5.7 have shown the faster convergence of the algorithms for spectrum estimation as compared to parameter estimation scenario.

For the spectrum estimation we assumed that the unknown spectrum $\boldsymbol{\omega}_0$ was obtained over 8 basis functions, thus leading to a sparsity ratio equal to 8/50. The power transmitted over each basis function is set equal to 0.7 mW. The variance for the observation noise is 0.001. For distributed spectrum estimation, we compared the DAMDC, the distributed oracle, l_0-ATC-LMS [12], the RZA-ATC-LMS algorithm [9] and the standard distributed ATC-LMS algorithms, all of them with the ATC diffusion strategy. Figure 5.9 shows the result of the distributed spectrum estimation of the algorithms discussed. It

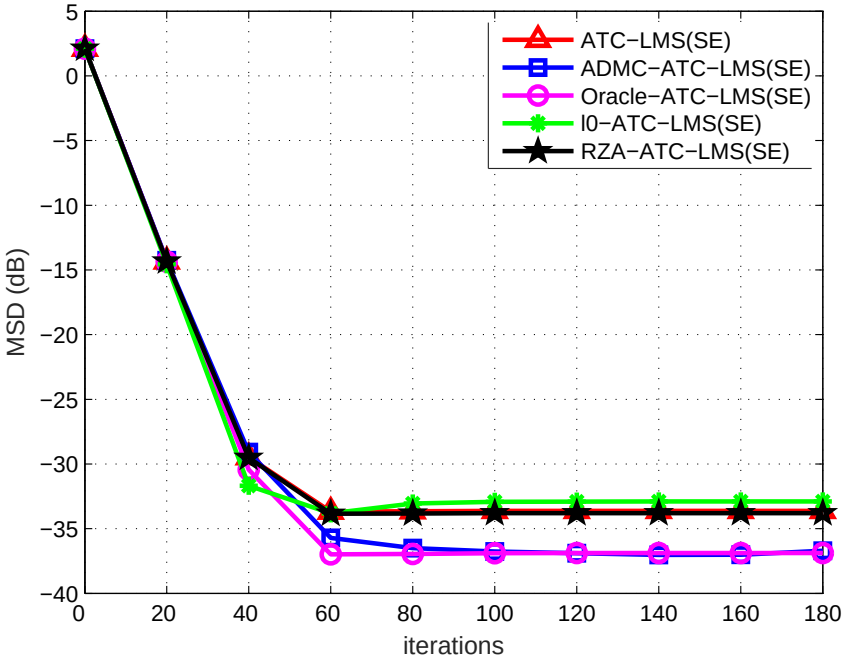

Figure 5.7 – MSD of the network for distributed spectrum estimation, $\mu = 0.45$, $\eta = 0.5 \times 10^{-3}$, $\rho_{RZA} = 3 \times 10^{-5}$, $\rho_{l0} = 3 \times 10^{-5}$, $\varepsilon = 0.1$,$SNR = 30dB$, $S = 20/50$.

can be observed that the ADMC algorithm is able to estimate the spectrum and strongly reduces the effect of the spurious terms, fitting much better the transmitted spectrum and rejecting the unused frequencies.

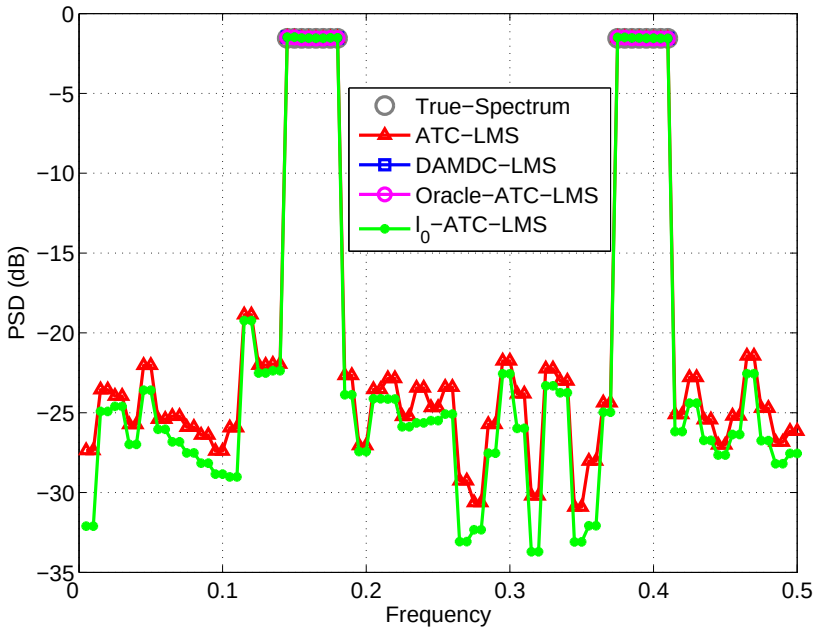

Figure 5.9 – Distributed spectrum estimation. Parameters: $\mu = 0.45$, $\eta = 0.5 \times 10^{-3}$, $\rho_{l0} = 3 \times 10^{-5}$, $\beta = 50$ and $S = 8/50$.

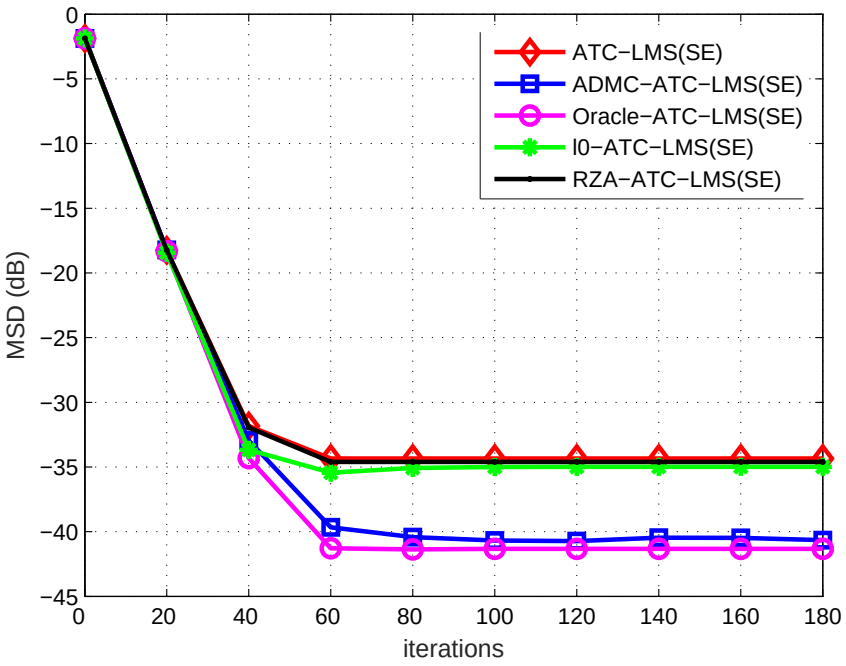

Figure 5.8 – MSD of the network for distributed spectrum estimation, $\mu = 0.45$, $\eta = 0.5 \times 10^{-3}$, $\rho_{RZA} = 3 \times 10^{-5}$, $\rho_{l0} = 3 \times 10^{-5}$ $\varepsilon = 0.1$, $SNR = 30dB$, $S = 8/50$.

To simulate the adaptation capability of the proposed technique, in Figure 5.10 we evaluated the behavior of the power estimated over an initially busy channel (the 16-th channel in this case), which ceases after iteration 500, comparing the results achieved by the DAMDC diffusion and the oracle diffusion LMS algorithms. We consider the same settings of the previous simulation. The transmitted power is set equal to 0.20 mW. We can notice how the proposed DAMDC algorithm is able to track the spectrum in an efficient and similar way compare to the oracle algorithm, due to its faster learning capability than existing algorithms.

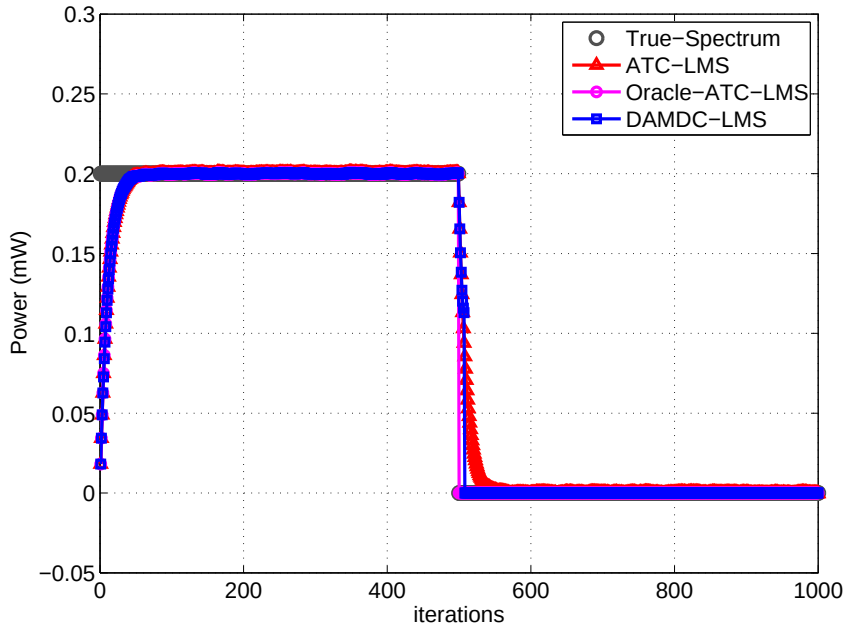

Figure 5.10 – Power spectrum tracking.

5.6 Summary

In this chapter we have proposed a distributed algorithm for parameter and spectrum estimation over sensor networks. The proposed DADMC-LMS algorithm outperforms previously sparsity-aware algorithms like RZA and ZA that use a penalty function to attract to zero the small magnitudes of the parameter vector. In all cases, the DAMDC obtained lower MSD values and faster convergence rates as compared to prior art and which is similar to the oracle algorithm. Simulations have shown that DAMDC-LMS is also suitable for spectrum estimation techniques. By exploiting sparsity and alternating optimization recursions to find the optimal parameter vector to transmit the signal, the DAMDC algorithm is able to improve the spectrum estimation performance and adaptation capability, with respect to the existing sparsity-aware and standard diffusion algorithms.

6
Conclusions and Future Work

In this work we have proposed distributed sparsity-aware algorithms for signal processing in sensor networks. Parameter estimation and spectrum estimation were the application scenarios chosen to evaluate the proposed algorithms. The proposed algorithms outperform some of the already reported algorithms in previous works in terms of MSD and convergence rate.

In chapter 3 distributed consensus and diffusion CG algorithms for parameter estimation have been presented. The proposed distributed CG algorithms have a faster convergence than the LMS and a very similar performance to the RLS. The diffusion ATC MCG algorithm have shown the best performance achieving the lowest MSD value at steady state with the fastest convergence rate.

We have demonstrated in chapter 4 that sparsity-aware techniques for distributed CG algorithms are appropriated tools for parameter estimation in sparse scenarios. The RZA diffusion ATC-MCG algorithm presents the best performance in terms of convergence rate an MSD value. All proposed sparsity-aware mechanisms outperform the distributed sparsity-aware LMS algorithms and have shown a close performance to the sparsity-aware RLS algorithms.

Chapters 3 and 4 have shown that diffusion strategies outperform the consensus protocol for CG and MCG algorithms. Specifically the diffusion ATC protocol have presented the best performance. The constraint of the consensus strategy forces the algorithm to converge to a common parameter vector. In contrast, the diffusion protocol gives freedom to the adaptation process, which influences the performance of the method. At the same time the order of the steps also has an effect over the performance of the algorithm.

In chapter 5 we have proposed a distributed algorithm, DAMDC-LMS, that approaches the oracle algorithms for parameter and spectrum estimation over sensor networks. The new proposal has obtained lower MSD values and faster convergence rates as compared to prior art sparsity-aware, classical algorithms and a similar behaviour to the oracle algorithm. The alternating process between continuous and discrete values enables the DAMDC-LMS algorithm to identify the positions with useful values in the parameter vector.

The DAMDC-LMS algorithm has also shown good capabilities for tracking and spectrum estimation.

The proposed algorithms in this thesis can be considered for future work in other fields of application. In particular the sparsity-aware CG algorithms developed can be implemented for distributed spectrum estimation scenario and can be combine with the DAMDC approach. Adaptive network topologies also represent an interesting area to apply the distributed algorithms covered in this research and to develop new strategies to exchange information between nodes.

Bibliography

[1] A. H. Sayed, *Adaptation, Learning, and Optimization over Networks*, Foundations and Trends in Machine Learning, vol. 7, no. 4 − 5, pp. 311 − 801, 2014.

[2] L. Zhi-Quan, G. Michael, L. Juan and S. Ananthram, "Distributed Signal Processing in Sensor Networks, IEEE Signal Processing Magazine", No. 15, July 2006.

[3] S. Haykin, *Adaptive Filter Theory*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[4] A. H. Sayed, Fundamentals of Adaptive Filtering. Hoboken, NJ, USA: John Wiley and Sons, 2003.

[5] G. L. Cássio and A. H. Sayed, "Incremental adaptive strategies over distributed networks", Proc. IEEE Trans. Signal Process, vol. 48, no. 8, pp.223-229, Aug 2007.

[6] S. Yuan T and A. H. Sayed, "Diffusion Strategies Outperform Consensus Strategies for Distributed Estimation over Adaptive Networks", Proc. IEEE Transactions on Signal processing, vol. 60, no. 12, December 2012.

[7] G. L. Cassio and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis", Proc.IEEE Trans. Signal Process, vol. 56, no. 7, pp. 3122-3136, July 2008.

[8] K. B. Giridhar and D. S. Mandyam, "Conjugate Gradient Techniques for Adaptive Filtering", Proc. IEEE Transactions On Circuits and Systems, vol. 39, no. 1 , January 1992

[9] P. D. Lorenzo and A. H. Sayed, "Sparse Distributed Learning Based on Diffusion Adaptation", Proc. IEEE Transactions on Signal Processing, vol. 61, no. 6, March 15, 2013.

[10] Y. Chen, Y. Gu, and A. O. Hero, "Sparse LMS for System Identification", Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, April 2009.

[11] P. D. Lorenzo and S. Barbarossa, "Distributed Least Mean Squares Strategies for Sparsity-Aware Estimation over Gaussian Markov Random Fields", Proc. IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), May 2014.

[12] E. M. Eksioglu, "RLS Adaptive Filtering with Sparsity Regularization", Proc. 10^{th} International Conference on Information Science, Signal Processing and their Applications (ISSPA), May 2010.

[13] Y. Zakharov and V. H. Nascimento, "Sparse RLS adaptive filter with diagonal loading" Proc. Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), pages. 806-810. November 2012

[14] P. S. Chang and A. N. Willson, Jr., "Analysis of Conjugate Gradient Algorithms for Adaptive Filtering", Proc. IEEE Transactions on Signal Processing, vol. 48, no. 2, February 2000

[15] S. Xu and R.C de Lamare, , "Distributed conjugate gradient strategies for distributed estimation over sensor networks", Proc. Sensor Signal Processing for Defense (SSPD), London September 2012

[16] R. C. de Lamare and R. Sampaio-Neto, "Sparsity-Aware Adaptive Algorithms Based on Alternating Optimization with shrinkage", Proc. IEEE Signal Processing Letters, vol. 21, no. 2, February 2014.

[17] P. D. Lorenzo, S. Barbosa and A. H. Sayed "Distributed Spectrum Estimation for Small Cell Networks Based on Sparse Diffusion Adaptation", Proc. IEEE Signal Processing Letters, vol. 20, no. 12, December 2013.

[18] S. Xu, *Distributed Signal Processing Algorithms for Wireless Networks*, Thesis submitted in partial fulfilment of the requirements for Doctor of Philosophy (Ph.D.), Communications and Signal Processing Research Group Electronics, University of York, May 2015.

[19] O. Taheria, S. A. Vorobyov *Reweighted l1-norm Penalized LMS for Sparse Channel Estimation and its Analysis*, Signal Processing, vol. 104, pp. 70-79, November 2014

[20] T. G. Miller, S. Xu and R.C de Lamare, "Consensus Distributed Conjugate Gradient Algorithms for Parameter Estimation over Sensor Networks", Proc. XXXIII Brazilian Telecommunications Symposium SBrT2015, September 2015.

[21] Y. Liu, C. L. and Z. Zhang, "Diffusion Sparse Least-Mean Squares Over Networks", Proc. IEEE Transactions on Signal Processing, vol. 60, no. 8, August 2012.

[22] O. Axelson, *Iterative Solution Methods*. New York: Cambridge Univ. Press,1994

[23] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd Ed. Baltimore, MD: John's Hopkins Univ. Press,1989

[24] S. Xu, R. C. de Lamare and H. V. Poor, "Distributed Compressed Estimation Based on Compressive Sensing," Proc. IEEE Signal Processing Letters, vol.22, no.9, pp.1311-1315, Sept. 2015.

[25] S. Xu, R. C. de Lamare and H. V. Poor, "Adaptive Link Selection Algorithms for Distributed Estimation", EURASIP Journal on Advances in Signal Processing, 2015.

[26] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. Towfic, "Diffusion strategies for adaptation and learning over networks", Proc. IEEE Signal Processing Magazine, vol. 30, no.3, pp. 155-171, May 2013.

[27] M.S.E. Abadi and Z. Saffari, "Distributed Estimation Over an Adaptive Diffusion Network Based on the Family of Affine Projection Algorithms", Proc. Sixth International Symposium on Telecommunications, November 2012.

[28] S. Theodoridis, *Machine Learning a Bayesian and Optimization Perspective*, Academic Press, March 2015.

[29] S. Xu, R. C. de Lamare and H. V. Poor, "Distributed Estimation Over Sensor Networks Based on Distributed Conjugate Gradient Strategies", Proc. IET Signal Processing, 2016.

[30] N. A. Lynch, *Distributed algorithms*, San Francisco, CA: Morgan Kaufmann, 1997.

[31] O, Jahromi and P Aarabi, "Distributed Spectrum Estimation in Sensor Networks", Proc.IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, pages. 849-52, May 2004.

[32] M.H. DeGroot, "Reaching a consensus", Journal of the Amercian Statistical Association vol. 69, no. 345, pp. 118-121, Mar. 1974.

[33] S. Wang, H. M. and B. Xi, D. Sun, "Conjugate Gradient-based Parameters Identification", 8th IEEE International Conference on Control and Automation, China, June 2010.

[34] G. Mateos and G. B. Giannakis, "Distributed Recursive Least-Squares: Stability and Performance Analysis", Proc. IEEE Transactions On Signal Processing, vol. 60, No. 7, July 2012.

[35] G. Mateos, I. D. Schizas and G. B. Giannakis "Distributed Recursive Least-Squares for Consensus-Based In-Network Adaptive Estimation" Proc. IEEE Transactions On Signal Processing, Vol. 57, No. 11, November 2009.

[36] A. Bertrand, M. Moonen and A. H. Sayed "Diffusion-Based Bias-Compensated RLS for Distributed Estimation over Adaptive Sensor Networks", 19th European Signal Processing Conference (EUSIPCO 2011), September 2011

[37] X.L. Feng and T.Z. Huang, "A finite-time consensus protocol of the multi-agent systems", World Academy of Science, Engineering and Technology International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering Vol:5, No:3, 2011.

[38] I. D. Schizas. G. Mateos and G. B. Giannakis, "Distributed LMS for Consensus-Based In-Network Adapting Processing", Proc. IEEE Transactions on Signal Processing, vol. 57, no. 6, June 2009

[39] E. Weiand A. Ozdaglar, "Distributed Alternating Direction Method of Multipliers", National Science Foundation under Career grant DMI-0545910 and AFOSR MURI FA9550-09-1-0538. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[40] G. Yuantao, J. Jian and M.Shunliang, "l_0 Norm Constraint LMS Algorithm for Sparse System Identification", Proc. IEEE Transactions on Signal Processing, vol. 16, no. 9, September 2009.

[41] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks-part II: Simultaneous and asynchronous node updating", Proc. IEEE Transactions on Signal Processing, vol. 58, no. 10, pages. 5292-5306, 2010.

[42] O. Jahromi and P. Aarabi, "Distributed spectrum estimation in sensor networks", Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, pages. 849-52, May 2004.

[43] R. Arablouei, S. Werner, Y.-F. Huang, K. Dogançay, "Distributed least mean-square estimation with partial diffusion", Proc. IEEE Transactions on Signal Processing, vol. 62, No. 2, pages. 472-484, January 2014.

[44] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis, "A sparsity promoting adaptive algorithm for distributed learning", Proc. IEEE Transactions on Signal Processing, vol. 60, no. 10, pages. 5412-5425, October 2012.

[45] S. Ozen, W. Hillery, M. Zoltowski, S.M. Nereyanuru, M. Fimoff, "Structured Channel Estimation Based Decision Feedback Equalizers for Sparse Multipath Channels with Applications to Digital TV Receivers", Proc. Signals, Systems and Computers, Asilomar, vol.1, pages.558 - 564, November 2002.

[46] L. Michael, D. David, M. P. John, "Sparse MRI: The Application of Compressed Sensing for Rapid MR Imaging", Magnetic Resonance in Medicine 58:1182-1195, 2007.

[47] X. Yunhai, S. Huina, W. Zhiguo, "A modified conjugate gradient algorithm with cyclic Barzilai-Borwein steplength for unconstrained optimization", Journal of Computational and Applied Mathematics, vol. 236 pages. 3101-3110, July 2012.

[48] J. Baocong, H. Jing and C. Lanping "A Modified Conjugate Gradient Algorithm with Sufficient Descent", Proc. Fourth International Joint Conference on Computational Sciences and Optimization (CSO), pages. 175-177, April 2011.