Energy and privacy aware estimation over adaptive networks

Ibrahim El Khalil Harrane

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

THÈSE DE DOCTORAT

Energy and privacy aware estimation over adaptive networks

Ibrahim El Khalil Harrane

Laboratoire J.-L. LAGRANGE

Présentée en vue de l'obtention du grade de docteur en science pour l'ingénieur
d'Université Côte d'Azur
Dirigée par : Cédric Richard
Co-encadrée par : Rémi Flamary
Soutenue le : 21 / 06 / 2019

Devant le jury, composé de :
David Brie, PU, Université de Lorraine
Olivier Michel, PU, Grenoble-INP
Frédéric Pascal, PU, CentraleSupélec
Alain Rakotomamonjy, PU, Université de Rouen
Jean-Yves Tourneret, PU, Université de Toulouse

Energy and privacy aware estimation over adaptive networks

Jury :

Rapporteurs
David Brie, PU, Université de Lorraine, France
Olivier Michel, PU, Grenoble-INP, France

Examinateurs
Frédéric Pascal, PU, CentraleSupélec, Gif-sur-Yvette, France
Alain Rakotomamonjy, PU, Université de Rouen, France
Jean-Yves Tourneret, PU, Université de Toulouse, France

Encadrants
Cédric Richard, PU, Université Côte d'azur, France
Rémi Flamary, MCF, Université Côte d'azur, France

Energy and privacy aware estimation over adaptive networks

Abstract

Distributed estimation over adaptive networks takes advantage of the interconnections between agents to perform parameter estimation from streaming data. Compared to their centralized counterparts, distributed strategies are resilient to links and agents failures, and are scalable. However, such advantages do not come without a cost. Distributed strategies require reliable communication between neighbouring agents, which is a substantial burden especially for agents with a limited energy budget. In addition to this high communication load, as for any distributed algorithm, there may be some privacy concerns particularly for applications involving sensitive data. The aim of this dissertation is to address these two challenges.

To reduce the communication load and consequently the energy consumption, we propose two strategies. The first one involves compression while the second one aims at limiting the communication cost by sparsifying the network.

For the first approach, we propose a compressed version of the diffusion LMS where only some random entries of the shared vectors are transmitted. We theoretically analyse the algorithm behaviour in the mean and mean square sense. We also perform numerical simulations that confirm the theoretical model accuracy. As energy consumption is the main focus, we carry out simulations with a realistic scenario where agents turn on and off to save energy. The proposed algorithm outperforms its state of the art counterparts.

The second approach takes advantage of the multitask setting to reduce the communication cost. In a multitask setting it is beneficial to only communicate with agents estimating similar quantities. To do so, we consider a network with two types of agents: cluster agents estimating the network structure, and regular agents tasked with estimating their respective objective vectors. We theoretically analyse the algorithm behaviour under two scenarios: one where all agents are properly clustered, and a second one where some agents are asigned to wrong clusters. We perform an extensive numerical analysis to confirm the fitness of the theoretical models and to study the effect of the algorithm parameters on its convergence.

To address the privacy concerns, we take inspiration from differentially private Algorithms to propose a privacy aware version of diffusion LMS. As diffusion strategies relies heavily on communication between agents, the data are in constant jeopardy. To avoid such risk and benefit from the information exchange, we propose to use Wishart matrices to corrupt the transmitted data. Doing so, we prevent data reconstruction by adversary neighbours as well as external threats. We theoretically and numerically analyse the algorithm behaviour. We also study the effect of the rank of the Wishart matrices on the convergence speed and privacy preservation.

Estimation distribuée respectueuse de la consommation d'énergie et de la confidentialité sur les réseaux adaptatifs

Résumé

L'estimation adaptative distribuée sur les réseaux tire parti des interconnexions entre agents pour effectuer une tâche d'estimation de paramètres à partir de flux continus de donnée. Comparées aux solutions centralisées, les stratégies distribuées sont robustes aux pertes de communications ou aux défaillances des agents. Cependant, ces avantages engendrent de nouveaux défis. Les stratégies distribuées nécessitent une communication permanente entre agents voisins, engendrant un coût considérable, en particulier pour les agents dont le budget énergétique est limité. Au-delà du coût de communication, comme pour tout algorithme distribué, on peut craindre des problèmes de confidentialité, en particulier pour les applications impliquant des données sensibles. L'objectif de cette thèse est de répondre à ces deux défis.

Pour réduire le coût de communication et par conséquent la consommation d'énergie, nous proposons deux stratégies. La première repose sur la compression tandis que la seconde vise à limiter les coûts de communication en considérant un réseau moins dense.

Pour la première approche, nous proposons une version compressée du diffusion LMS dans laquelle seules quelques composantes des vecteurs de données partagés, sélectionnées aléatoirement, sont transmises. Nous effectuons une analyse théorique de l'algorithme ainsi que des simulations numériques pour confirmer la validité du modèle théorique. Nous effectuons aussi des simulations selon un scénario réaliste dans lequel les agents s'allument et s'éteignent pour économiser les ressources énergétiques. L'algorithme proposé surpasse les performances des méthodes de l' état de l'art.

La seconde approche exploite l'aspect multitâche pour réduire les coûts de communication. Dans un environnement multitâche, il est avantageux de ne communiquer qu'avec des agents qui estiment des quantités similaires. Pour ce faire, nous considérons un réseau avec deux types d'agents: des agents de cluster estimant la structure du réseau et des agents réguliers chargés d'estimer leurs paramètres objectifs respectifs. Nous analysons théoriquement le comportement de l'algorithme dans les deux scénarios: l'un dans lequel tous les agents sont correctement groupés et l'autre dans lequel certains agents sont affectés au mauvais cluster. Nous effectuons une analyse numérique approfondie pour confirmer la validité des modèles théoriques et pour étudier l'effet des paramètres de l'algorithme sur sa convergence.

Pour répondre aux préoccupations en matière de confidentialité, nous nous sommes inspirés de la notion de "differential privacy" pour proposer une version du diffusion LMS prenant en compte la confidentialité des données. Sachant que le diffusion LMS repose sur la communication entre agents, la sécurité des données est constamment menacée. Pour éviter ce risque et tirer profit de l'échange d'informations, nous utilisons des matrices aléatoires de Wishart pour corrompre les données transmises. Ce faisant, nous empêchons la reconstruction des données par les voisins adverses ainsi que les menaces externes. Nous analysons théoriquement et numériquement le comportement de l'algorithme. Nous étudions également l'effet du rang des matrices de Wishart sur la vitesse de convergence et la préservation de confidentialité.

Acknowledgements

First of all, I wish to express my gratitude to my supervisors Cédric Richard and Rémi Flamary for offering me the opportunity to start my academic carrier through a PhD position. Rémi, thanks to his enthusiasm, was the first one to interest me in signal processing. I still remember the first day of my internship with him. That day I knew that working in this field was what I always wanted to do.

I would like to thank the Jury members: Prof. Prof. Frédéric Pascal, Prof. Alain Rakotomamonjy and Prof. Jean-Yves Tourneret for accepting to be part of my thesis committee. I wish to express my gratitude to David Brie, Prof. Olivier Michel for the effort and time they spent reviewing my thesis.

During the last few years, I worked closely with Rémi and Cédric. I was fortunate to learn from their expertise and immense knowledge. They were always present to guide me, encourage me, read my papers and support me. I can not imagine better supervisors, and for that I am very grateful.

Throughout my PhD, I was lucky to meet and befriend two incredible women, Roula and Rita. I am very thankful for the fun moments we had together. I would also like to thank my Phd fellow students: Ikram, fei and Micrea for their kindness and friendship.

A very special thank you to my lab colleagues and teachers, Prof. André Ferrari, Prof. Céline Theys, Prof. Claud aime for their support and the fun and stimulating lunch breaks. I am also very grateful to all my teachers who taught me and inspired me.

Last but not least, I wish to thank my parents: Mohamed and Ouarda, my brother Houcem and my sister Sihem for their continues support, encouragement and unconditional love.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

LMS	Least Mean Squares algorithm
DLMS	Diffusion LMS algorithm
CD	Compressed Diffusion LMS algorithm
DCD	Compressed Diffusion LMS algorithm
ATC	Adapt Then Combine strategy
CTA	Combine Then Adapt strategy
WSN	Wireless Sensor Networks
MSD	Mean Square Deviation
i.i.d.	independent and identically distributed
RHS	right-hand side

1 Introduction

Contents

In this very first chapter, we discuss distributed estimation over networks, its advantages and the challenges resulting from the distribution process. We first give a brief introduction to distributed strategies in the single and multitask settings. Then, we sum up the solution we proposed to tackle the previously discussed challenges.

1.1 Distributed processes: from nature to signal processing

During this last decade, the world has witnessed a huge numerical leap. Thanks to impressive achievements in various fields such as signal processing, machine learning and electrical engineering, we are living in a world where powerful computers are wrapped over one's wrist as a smart watch or held in a palm of a hand as a smart phone; in a world where autonomous cars and smart houses are no longer a science-fiction fantasy; a world where social networks are so powerful that they can determine one's carrier or an entire country's future. Behind such evolution there is a common thread, that is, networking. For instance autonomous cars relay on GPS signals, and smart houses on heterogenous sensors networks.

If we take a step back to reflect on this advancement and observe nature around us, we would realise that networking runs at a level as deep as the DNA level. Through natural selection, animals evolved to live in groups and cooperate to survive. Observe for instance, predatorial animals hunt in packs such as wolfs and prey herding to maximise their chances of survival against predators by relying on their strength as a group.

This evolution can be felt, as humans evolved to live in societies and cooperate for their survival. This biological evolution found a new expression in the numerical world as nothing else but social networks.

With the benefits of cooperation, it was only a matter of time for scientists to take inspiration and incorporate it in algorithms which resulted in the inception of different forms of distributed algorithms. For instance [Nedic and Ozdaglar, 2009] proposed a consensus based distributed algorithm

where agents cooperate to minimise a common cost function and reach consensus after convergence. Such condition seemed adequate for this sort of estimation problem since the agents are estimating the same quantity.

1.2 Distributed strategies

Distributed strategies mainly rely on two types of communication. On the one hand, we have strategies where each agent only communicates with one, and only one, neighbour in a circular manner. Such strategies are known as incremental strategies. They can be attractive as they limit the network communication. However, this advantage comes at a high cost: first of all, a cyclic communication path needs to be determined. Such problem is NP hard. In addition, such strategy has a glaring defect, that is link or agent failures. It takes only one severed link to bring the whole communication path down. On the other hand, we have diffusion strategies. In this case information is diffused to a subset of direct neighbours rather than incrementally transferred from an agent to another. Even though such protocol increases the communication load, it also hugely increases the algorithm resilience to links and agents failures and immensely simplifies the algorithm as it does not deal with an NP hard estimation problem. There are mainly two types of strategies, single task and multitask strategies.

Single task strategies Diffusion strategies consider a network of connected agents aiming to estimate one common parameter. Those agents are usually small devices with small computational capabilities. However, through communication and cooperation, they are able to accurately execute complex tasks. These agents deal with a stream of information which allows them to adapt to data statistics drifts. Such setting is highly attractive as these networks are highly scalable and adaptable. They can be tasked with a large variety of applications ranging from target location to power estimation in mobile networks.

Multitask strategies In many cases, there is more than one parameter to estimate, for instance, a location system might be interested in locating different targets in different areas. In such instance, we rely on multitask networks. Agents sharing the same task form clusters. Depending on whether the agent are aware of their clusters, there are two types of scenarios, a supervised and an unsupervised one. In the case of the later scenario, each agent needs to estimate the cluster it belongs to.

LMS Algorithm Several algorithms have benefited from a distributed make-over. However, one particular algorithm found a huge success among the community [Chen et al., 2016, Parreira et al., 2012, Sayed, 2013b, Liu et al., 2008b, Narayan and Peterson, 1981]. We are referring to the Least Mean Squares algorithm or *LMS* for short. This wide use is mostly due to its simple implementation and yet high performance. The *LMS* algorithm belongs to the family of stochastic algorithms. It resembles the steepest descent method where the gradient determines the descent direction. The

LMS algorithm uses a stochastic gradient of the mean square error to converge towards the optimum filter weights. Simplicity and low complexity have granted the *LMS* its status among the community.

1.3 Contributions

In this work, we try to improve diffusion LMS performance under different conditions. In the following chapter, we shall begin by recalling the single task diffusion LMS and its theoretical analysis from [Sayed, 2003]. Then, we explore the recent advances related the scope to this work. This chapter will serve as a reference for the following chapters.

In the third chapter we address the only advantage the incremental strategies have over diffusion strategies: the communication load. To do so, we consider a compression procedure that reduces the communicated entries of each shared vector to a set number. We then proceed by theoretically studying the algorithm weights convergence in the mean and mean square sense. Then we numerically verify the theoretical model accuracy. We found that such approach is highly adapted for networks with very low energy budgets as we simulated such network with agents turning on and off to save energy. This study can be viewed under another angle where a certain number of communicated packets is lost. The theoretical results may prove very useful in determining the performance loss under such conditions.

The fourth chapter focuses on the very sensitive subject of privacy. The problem seems straightforward as an encryption should suffice to guaranty the privacy of the shared data. However, this approach does not consider an adverse neighbour. For the agents to cooperate the encryption key needs to be shared and the data are fully disclosed to the neighbouring agents. In order to take advantage of the cooperation without compromising the data privacy, we took inspiration from *differential privacy* that compromises the data set through different means so that the individual information is preserved while the statistical trend can still be extracted. We therefore proposed an algorithm based on this approach that takes into account the streaming nature of the data in diffusion strategies. Furthermore, we maintain control over the data alteration through a parameter. We then carried out a theoretical analysis where we tested the theoretical model accuracy under different conditions. We also studied the impact of this approach on the algorithm performance. As expected, we reached the conclusion that there is a trade-off between the privacy and the algorithm accuracy.

The fifth chapter deals with multitask estimation. Although some authors considered unsupervised multitask estimation problems [Nassif et al., 2016d, Chen et al., 2015a] the rest of the literature mainly focused on the supervised counterpart. We then proposed an unsupervised distributed algorithm that takes advantage of both centralised and distributed methods with little if none of their drawbacks. The algorithm considers two types of agents, cluster agents tasked with stochastically estimating the cluster objective vector and the graph adjacency matrix. The other agents estimate their objective vector by minimizing a regularized cost function. Similarly to the previous chapters, we theoretically analysed the algorithm. However, this time, we considered two scenarios. The first one considers a network where all agents are correctly clustered while the second considers the case

where some of them are affected to wrong clusters. Finally, we conducted a numerical analysis to confirm the fitness of the theoretical models and to test the algorithm performance when agents are miss-assigned. We found out that, first, the theoretical models are accurate and the bias introduced in the case of mislabelled agents is related to the difference between the wrongly affected objective vectors and the correct ones.

Finally, we conclude the thesis by summing up the main finding and explore some future research directions.

1.4 Published papers

- Harrane, I. E. K., Flamary, R., and Richard, C. (2019). On reducing the communication cost of the diffusion lms algorithm. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):100–112.
- Harrane, I. E. K., Flamary, R., and Richard, C. (2016a). Doubly compressed diffusion lms over adaptive networks. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 987–991.
- Harrane, I. E. K., Flamary, R., and Richard, C. (2016b). Toward privacy-preserving diffusion strategies for adaptation and learning over networks. In *Proc. EUSIPCO*, Budapest, Hungary.

2 Diffusion Least Mean Square

Contents

In this chapter, we give an overview of distributed estimation with a focus on diffusion LMS algorithm. We start by formulating the estimation problem. Then, we present a thorough theoretical analysis of the algorithm starting with the mean error stability. Next we analyse the convergence in the mean square sense. We conclude the theoretical analysis with the network mean square performance in its steady and transient states. We also provide a numerical analysis where we demonstrate the theoretical model fitness. Finally, we take a deep dive into the literature and explore the recent developments of diffusion LMS.

2.1 Introduction

In the recent years, we have witnessed a staggering technological development, computers have never been as powerful, electrical chips as efficient and networks as big. With the later development in mind, distributed processing has received a huge attention. As the world is moving towards a connected model, many problems evolved to a networked structure such as sensor networks [Carmona et al., 2013, Vincent et al., 2014], energy grids, mobile phone networks, social and economical networks and swarming behaviour in the biological realm [Tu and Sayed, 2010].

All of these networks share a similar structure. They are built from interconnected agents, usually with limited computational power and are distributed over geographical areas. Distributed processing is based on the cooperation between agents to offer an enhanced global estimation performance [Sayed, 2003]. Agents are only allowed to communicate with their direct neighbours. They have fairly simple tasks to accomplish. However, the communication between agents allows a

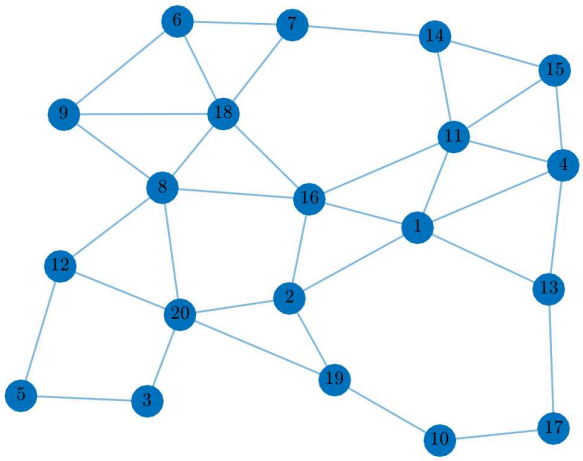

Figure 2.1: An interconnected network of $N = 20$ agents. Every agent k has a neighbourhood $\mathcal{N}_k$. For instance, $\mathcal{N}_2 = \{1, 2, 16, 19, 20\}$ and $\mathcal{N}_{16} = \{1, 2, 8, 11, 18\}$.

more complex networks behaviour. Nature is a perfect example, for instance birds flock, ants and micro-bacteria form colonies, sheep herd and fish form schools. All with the same objective, that is, increasing their survival rate through cooperation [Sayed et al., 2013].

Taking inspiration from nature, many distributed algorithms have been developed and studied to solve different kinds of problems such as inference and optimization problems. These algorithms found application in numerous domains such as machine learning, and networked sensors. In this kind of applications, nodes are tasked with an optimization problem to estimate a given parameter. Furthermore, since real life data are always shifting, networks need to cope with this drift. As a consequence, instead of learning from a finite data set, agents have to process a continuous stream of data and adapt accordingly. This is known as online learning.

In this chapter, we start setting the basic of diffusion LMS algorithm (diffusion LMS) based on [Sayed, 2013b]. Then we give a detailed theoretical analysis of both first and second order moments of the mean error. Finally, we explore the different variants of the algorithm ranging from single bit compression to multitask diffusion LMS.

2.2 Problem Formulation

In this section, we recall the Diffusion Least Mean Square algorithm as formulated in [Sayed, 2013b]. Let us consider an interconnected network of N nodes as it is depicted in Figure 2.1. Each node is tasked with estimating an $L \times 1$ unknown vector $\boldsymbol{w}^o$ from collected measurements. Node k has access to local streaming measurements $\{d_k(i), \boldsymbol{u}_{k,i}\}$ where $d_k(i)$ is a scalar zero-mean reference signal, and $\boldsymbol{u}_{k,i}$ is an $L \times 1$ zero-mean regression vector with covariance matrix $\boldsymbol{R}_{u_k} = \mathbb{E}\{\boldsymbol{u}_{k,i}\boldsymbol{u}_{k,i}^\top\} \succ 0$. The data at agent k and time i are assumed to be related via the linear regression model:

$$d_k(i) = \boldsymbol{u}_{k,i}^\top \boldsymbol{w}^o + v_k(i) \tag{2.1}$$

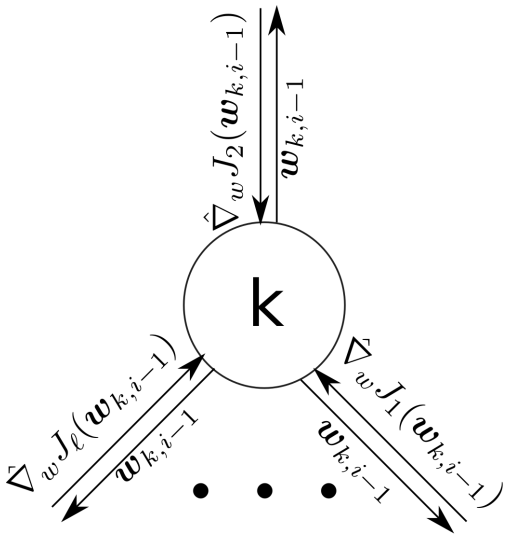

FIGURE 2.2: Communication between agents for diffusion LMS. At each iteration, an agent k sends it local estimate $\boldsymbol{w}_{k,i}$ to its neighbours. It then receives their stochastic gradients $\hat{\nabla}_{\boldsymbol{w}} J_\ell(\boldsymbol{w}_{k,i})$. Similarly the same agent k, receives its neighbours local estimates $\boldsymbol{w}_{\ell,i}$ and sends back it stochastic gradients $\hat{\nabla}_{\boldsymbol{w}} J_k(\boldsymbol{w}_{\ell,i})$.

where $\boldsymbol{w}^o$ is the unknown parameter vector to be estimated, and $v_k(i)$ is a zero-mean i.i.d. noise with variance $\sigma^2_{v,k}$. The noise $v_k(i)$ is assumed to be independent of any other signal. Let $J_k(\boldsymbol{w})$ be a differentiable convex cost function at agent k. For this chapter as well as the upcoming ones, we shall consider the mean-square-error criterion:

$$J_k(\boldsymbol{w}) = E\{|d_k(i) - \boldsymbol{u}_{k,i}^\top \boldsymbol{w}|^2\}. \tag{2.2}$$

Diffusion LMS strategies seek the minimizer of the following aggregate cost function:

$$J^{\text{glob}}(\boldsymbol{w}) = \sum_{k=1}^{N} J_k(\boldsymbol{w}) \tag{2.3}$$

in a cooperative manner through online adaptation. Let $\boldsymbol{w}_{k,i}$ denote the estimate of the minimiser of (2.3) at node k and time instant i. The general structure of diffusion LMS in its Adapt-then-Combine (ATC) form is given by:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1}) \tag{2.4}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_{\ell,i} \tag{2.5}$$

where $\hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1}) = -\boldsymbol{u}_{\ell,i}[d_\ell(i) - \boldsymbol{u}_{\ell,i}^\top \boldsymbol{w}_{k,i-1}]$, $\mathcal{N}_k$ denotes the neighbourhood of node k including k, and μ_k is a positive step-size. Nonnegative coefficients $\{a_{\ell k}\}$ and $\{c_{\ell k}\}$ define a left-stochastic matrix $\boldsymbol{A}$ and a right-stochastic matrix $\boldsymbol{C}$, respectively.

We illustrate the communication between agents in Figure 2.2. In the first step each agent k adapts its local estimate $\boldsymbol{w}_{k,i}$ based on its observations and gradients provided by its neighbours, hence

Table 2.1: List of the symbols and notations used in chapter 2

Symbol	Definition
L	length of the parameter vectors
N	network agents number
$\mathcal{N}_k$	neighbourhood of agent k including itself
$\boldsymbol{w}_{k,i}$	instantaneous estimate at agent k
$\boldsymbol{w}^o$	optimum parameter vector
$d_k(i)$	reference signal for agent k at time instant i
$\boldsymbol{u}_{k,i}$	regression vector of agent k
$v_k(i)$	additive noise at agent k

the appellation of this step as the adaptation step. The second step, as its name states, consists of combining the local estimates of the neighbouring nodes. Note that inverting the aforementioned steps results in a second variation of diffusion LMS known as Combine-then-Adapt (CTA). It must be noted that the Adapt-then-Combine (ATC) strategy outperforms the (CTA) as it has been proven in [Sayed, 2008]. The ATC diffusion run is described for an agent k in 2.1.

2.3 Theoretical analysis

In this section we are interested in analysing the behaviour of diffusion LMS in the mean and mean square sense as it has been performed in [Sayed, 2013b]. However, before proceeding with the analysis, let us introduce the following assumption on the regression data and the additive noise.

Assumption 2.1. *The regression vectors $\boldsymbol{u}_{k,i}$ arise from a zero-mean random process that is temporally white and spatially independent.*

Assumption 2.2. *The additive noise signals $v_k(i)$ are temporally white and spatially independent zero-mean random variables.*

A direct consequence of 2.1 is that $\boldsymbol{u}_{k,i}$ is independent of $\boldsymbol{w}_{\ell,j}$ for all ℓ and $j < i$. Let us now proceed with the algorithm analysis.

2.3.1 Weight error vector recursion

Let us introduce the error vectors $\widetilde{\boldsymbol{w}}_{k,i}$ and $\widetilde{\boldsymbol{\psi}}_{k,i}$ as

$$\widetilde{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i} \tag{2.6}$$

$$\widetilde{\boldsymbol{\psi}}_{k,i} = \boldsymbol{w}^o - \psi_{k,i} \tag{2.7}$$

We replace the reference signal $d_k(i)$ by its definition (2.1) in the equations (2.4)–(2.5)

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \boldsymbol{u}_{\ell,i} [\boldsymbol{u}_{\ell,i}^\top \boldsymbol{w}^o + v_\ell(i) - \boldsymbol{u}_{\ell,i}^\top \boldsymbol{w}_{k,i-1}] \tag{2.8}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_{\ell,i} \tag{2.9}$$

Subtracting $\boldsymbol{w}^o$ from both sides

$$\widetilde{\boldsymbol{\psi}}_{k,i} = \widetilde{\boldsymbol{w}}_{k,i-1} - \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \left(\boldsymbol{R}_{u_\ell,i} \, \widetilde{\boldsymbol{w}}_{k,i-1} + \boldsymbol{s}_{\ell,i} \right) \tag{2.10}$$

$$\widetilde{\boldsymbol{w}}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \widetilde{\boldsymbol{\psi}}_{\ell,i} \tag{2.11}$$

where

$$\boldsymbol{R}_{u_\ell,i} = \boldsymbol{u}_{\ell,i} \boldsymbol{u}_{\ell,i}^\top \tag{2.12}$$

$$\boldsymbol{s}_{\ell,i} = v_\ell(i) \boldsymbol{u}_{\ell,i} \tag{2.13}$$

We introduce concatenated versions of the error vectors $\widetilde{\boldsymbol{w}}_i$ and $\widetilde{\boldsymbol{\psi}}_i$ and the noise vector $\boldsymbol{s}_i$

$$\widetilde{\boldsymbol{w}}_i = \mathrm{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\} \tag{2.14}$$

$$\widetilde{\boldsymbol{\psi}}_i = \mathrm{col}\{\widetilde{\boldsymbol{\psi}}_{1,i}, \widetilde{\boldsymbol{\psi}}_{2,i}, \ldots, \widetilde{\boldsymbol{\psi}}_{N,i}\} \tag{2.15}$$

$$\boldsymbol{s}_i = \mathrm{col}\{\boldsymbol{s}_{1,i}, \boldsymbol{s}_{2,i}, \ldots, \boldsymbol{s}_{N,i}\} \tag{2.16}$$

$$\tag{2.17}$$

Due to the zero mean of both the regression data $\boldsymbol{u}_{k,i}$ and the noise $\boldsymbol{s}_{k,i}$ we have:

$$\mathbb{E}\{\boldsymbol{s}_i\} = \boldsymbol{0}_{N \times L,1} \tag{2.18}$$

and the covariance matrix of the vector $\boldsymbol{s}_i$ is an $(NM \times NM)$ bloc diagonal matrix

$$\boldsymbol{\mathcal{S}} = \mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{s}_i\} = \mathrm{diag}\{\boldsymbol{\sigma}_{v,1}^2 \boldsymbol{R}_{u_1}, \boldsymbol{\sigma}_{v,2}^2 \boldsymbol{R}_{u_2}, \ldots, \boldsymbol{\sigma}_{v,N}^2 \boldsymbol{R}_{u_N}\} \tag{2.19}$$

where

$$\boldsymbol{R}_{u_k} = \mathbb{E}\{\boldsymbol{R}_{u_k,i}\} \tag{2.20}$$

We further introduce the following matrices

$$\boldsymbol{\mathcal{R}}_i = \mathrm{diag}\{\boldsymbol{R}_{1,i}, \boldsymbol{R}_{2,i}, \ldots, \boldsymbol{R}_{N,i}\} \tag{2.21}$$

$$\boldsymbol{R}_{k,i} = \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \boldsymbol{R}_{u_\ell,i} \tag{2.22}$$

$$\boldsymbol{\mathcal{A}} = \boldsymbol{A} \otimes \boldsymbol{I}_L \tag{2.23}$$

$$\boldsymbol{\mathcal{M}} = \mathrm{diag}\{\mu_1 \boldsymbol{I}_L, \mu_2 \boldsymbol{I}_L, \ldots, \mu_N \boldsymbol{I}_L\} \tag{2.24}$$

$$\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_L \tag{2.25}$$

Algorithm 2.1 Local updates at node k for diffusion LMS

for i **do**=1 ...
  for $\ell \in \mathcal{N}_k \setminus \{k\}$ **do**
    send $\boldsymbol{w}_{k,i}$ to node ℓ
    receive from node ℓ the gradient vector: $\hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1})$
  end for
  update the intermediate estimate: $\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1})$
  calculate the local estimate: $\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_{\ell,i}$
end for

where $\otimes$ denotes the Kronecker product. We reformulate the equations (2.10)–(2.11) using the newly introduced error vectors

$$\begin{aligned} \widetilde{\boldsymbol{\psi}}_i &= (\boldsymbol{I}_{LN} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i)\, \widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top \boldsymbol{s}_i \\ \widetilde{\boldsymbol{w}}_i &= \boldsymbol{\mathcal{A}}^\top \widetilde{\boldsymbol{\psi}}_i \end{aligned} \tag{2.26}$$

Finally we combine the equations (2.26) to obtain

$$\widetilde{\boldsymbol{w}}_i = \underbrace{\boldsymbol{\mathcal{B}}_i \widetilde{\boldsymbol{w}}_{i-1}}_{\text{Data term}} - \underbrace{\boldsymbol{\mathcal{G}}\boldsymbol{s}_i}_{\text{Noise term}} \tag{2.27}$$

where

$$\boldsymbol{\mathcal{B}}_i = \boldsymbol{\mathcal{A}}^\top (\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i) \tag{2.28}$$

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top \tag{2.29}$$

The error vector $\widetilde{\boldsymbol{w}}_i$ has two terms, one term solely depending on the data (left term) and a noise term (right term).

2.3.2 Mean error analysis

In this part, we are interested in studying the algorithm stability in the mean.

Taking expectation of both sides of (2.27) leads to

$$\mathbb{E}\{\widetilde{\boldsymbol{w}}_i\} = \mathbb{E}\{\boldsymbol{\mathcal{A}}^\top (\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i)\, \widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top \boldsymbol{s}_i\} \tag{2.30}$$

using the independence between the regression data and the error vector $\widetilde{\boldsymbol{w}}_i$ as assumed in Assumptions 2.1 and 2.2, in addition to the zero mean of the noise vector $\boldsymbol{s}_i$ we find that:

$$\mathbb{E}\{\widetilde{\boldsymbol{w}}_i\} = \boldsymbol{\mathcal{B}}\, \mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}\} \tag{2.31}$$

where

$$\boldsymbol{\mathcal{B}} = \mathbb{E}\{\boldsymbol{\mathcal{B}}_i\} = \boldsymbol{\mathcal{A}}^\top \left(\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}\right) \tag{2.32}$$

$$\boldsymbol{\mathcal{R}} = \mathbb{E}\{\boldsymbol{\mathcal{R}}_i\} \tag{2.33}$$

It is noteworthy that the matrix $\boldsymbol{\mathcal{B}}$ depends on the networks topology encoded in the matrix $\boldsymbol{A}$, the data profile represented by the matrix $\boldsymbol{\mathcal{R}}$ and the step-size parameters as they build up the matrix $\boldsymbol{\mathcal{M}}$.

From (2.31) all the estimators across the network will converge to the optimal solution $\boldsymbol{w}^o$ provided that the matrix $\boldsymbol{\mathcal{B}}$ is stable [Sayed, 2014] i.e.:

$$\rho\left(\boldsymbol{\mathcal{B}}\right) < 1 \tag{2.34}$$

where $\rho(\cdot)$ denotes the spectral radius of its matrix argument. This condition is verified for small enough step-sizes μ_k such as

$$0 < \mu_k < \frac{2}{\lambda_{max}(\boldsymbol{R}_k)} \tag{2.35}$$

where $\lambda_{max}(.)$ denotes the largest eigenvalue of it matrix argument. In other words for a small enough step-size μ_k, when $i \to \infty$, for all the agents k, $\mathbb{E}\{\boldsymbol{w}_{k,i}\} \to \boldsymbol{w}^o$.

2.3.3 Mean square error analysis

While the algorithm stability is crucial, its accuracy is of utmost importance. In order of measure this quantity, we use the mean square deviation (MSD). For a wider study, we consider the weighted measure $||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\Sigma}}$ where $\boldsymbol{\Sigma}$ is a positive-definite matrix. The choice of $\boldsymbol{\Sigma}$ will determine the evaluated measure. Setting it, for example, to $\boldsymbol{\mathcal{R}}_u$ will result in the excess means square error EMSD.

The quantity of interest is the following

$$\begin{aligned}
\mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\Sigma}} &= \mathbb{E}\{\left(\boldsymbol{\mathcal{B}}_i\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{G}}\boldsymbol{s}_i\right)^\top \boldsymbol{\Sigma}\left(\boldsymbol{\mathcal{B}}_i\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{G}}\boldsymbol{s}_i\right)\} \\
&= \mathbb{E}\{\widetilde{\boldsymbol{w}}^\top_{i-1}\boldsymbol{\mathcal{B}}_i^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_i\widetilde{\boldsymbol{w}}_{i-1}\} - \mathbb{E}\{\widetilde{\boldsymbol{w}}^\top_{i-1}\boldsymbol{\mathcal{B}}_i^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}\boldsymbol{s}_i\} \\
&- \mathbb{E}\{\boldsymbol{s}_i^\top\boldsymbol{\mathcal{G}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_i\widetilde{\boldsymbol{w}}_{i-1}\} + \mathbb{E}\{\boldsymbol{s}_i^\top\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}\boldsymbol{s}_i\}
\end{aligned} \tag{2.36}$$

using the independence between the signals and noise, and noise zero mean, we find that:

$$\mathbb{E}\{||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\Sigma}}\} = \underbrace{\mathbb{E}\{\widetilde{\boldsymbol{w}}^\top_{i-1}\boldsymbol{\mathcal{B}}_i^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_i\widetilde{\boldsymbol{w}}_{i-1}\}}_{\text{Data term}} + \underbrace{\mathbb{E}\{\boldsymbol{s}_i^\top\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}\boldsymbol{s}_i\}}_{\text{Noise term}} \tag{2.37}$$

We shall start with the last term of (2.37)

$$\begin{aligned}
\mathbb{E}\{\boldsymbol{s}_i^\top\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}\boldsymbol{s}_i\} &= \text{trace}\left(\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}\mathbb{E}\{\boldsymbol{s}_i\boldsymbol{s}_i^\top\}\right) \\
&= \text{trace}\left(\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}\mathbb{E}\{\boldsymbol{s}_i\boldsymbol{s}_i^\top\}\right) \\
&= \text{trace}\left(\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{S}}\right)
\end{aligned} \tag{2.38}$$

Table 2.2: List of symbols defined throughout the performance analysis chapter 2

Symbol	Equation
$\widetilde{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i}$	(2.6)
$\widetilde{\boldsymbol{\psi}}_{k,i} = \boldsymbol{w}^o - \psi_{k,i}$	(2.7)
$\boldsymbol{\mathcal{R}}_i = \text{diag}\{\boldsymbol{R}_{1,i}, \boldsymbol{R}_{2,i}, \ldots, \boldsymbol{R}_{N,i}\}$	(2.21)
$\boldsymbol{R}_{k,i} = \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \boldsymbol{R}_{u_\ell,i}$	(2.22)
$\boldsymbol{R}_{u_\ell,i} = \boldsymbol{u}_{\ell,i} \boldsymbol{u}_{\ell,i}^\top$	(2.12)
$\boldsymbol{R}_{u_k} = \mathbb{E}\{\boldsymbol{R}_{u_k,i}\}$	(2.20)
$\boldsymbol{s}_{\ell,i} = v_\ell(i) \boldsymbol{u}_{\ell,i}$	(2.13)
$\boldsymbol{\mathcal{S}} = \mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{s}_i\} = \text{diag}\{\boldsymbol{\sigma}_{v,1}^2 \boldsymbol{R}_{u_1}, \boldsymbol{\sigma}_{v,2}^2 \boldsymbol{R}_{u_2}, \ldots, \boldsymbol{\sigma}_{v,N}^2 \boldsymbol{R}_{u_N}\}$	(2.19)
$\boldsymbol{\mathcal{A}} = \boldsymbol{A} \otimes \boldsymbol{I}_L$	(2.23)
$\boldsymbol{\mathcal{M}} = \text{diag}\{\mu_1 \boldsymbol{I}_L, \mu_2 \boldsymbol{I}_L, \ldots, \mu_N \boldsymbol{I}_L\}$	(2.24)
$\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_L$	(2.25)
$\boldsymbol{\mathcal{B}}_i = \boldsymbol{\mathcal{A}}^\top (\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{R}}_i)$	(2.28)
$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}^\top$	(2.29)
$\boldsymbol{\mathcal{B}} = \mathbb{E}\{\boldsymbol{\mathcal{B}}_i\} = \boldsymbol{\mathcal{A}}^\top (\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{R}})$	(2.32)
$\boldsymbol{\mathcal{R}} = \mathbb{E}\{\boldsymbol{\mathcal{R}}_i\}$	(2.33)
$\boldsymbol{\mathcal{F}} = (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}}) - (\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{\mathcal{A}}) - (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}})$	(2.47)
$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{G}}^\top \boldsymbol{\mathcal{S}} \boldsymbol{\mathcal{G}}$	(2.49)

where we used the trace(.) properties. $\boldsymbol{\mathcal{S}}$ is defined in (2.19). Regarding the first term of the expectation (2.37), we have

$$\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}^\top \boldsymbol{\mathcal{B}}_i^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_i \widetilde{\boldsymbol{w}}_{i-1}\} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{i-1}||_{\boldsymbol{\Sigma}'}^2 \tag{2.39}$$

where we introduce the new weighing matrix $\boldsymbol{\Sigma}'$

$$\begin{aligned} \boldsymbol{\Sigma}' &= \mathbb{E}\{\boldsymbol{\mathcal{B}}_i^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_i\} \\ &= \mathbb{E}\{(\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{R}}_i)^\top \boldsymbol{\mathcal{A}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}}^\top (\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{R}}_i)\} \\ &= \boldsymbol{\mathcal{A}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}}^\top - \boldsymbol{\mathcal{A}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{R}} - \boldsymbol{\mathcal{R}} \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{A}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}}^\top + \mathcal{O}(\mu^2) \end{aligned} \tag{2.40}$$

where $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{C}}$ and $\boldsymbol{\mathcal{M}}$ are defined in (2.23)–(2.25) and $\boldsymbol{\mathcal{R}}$ in (2.33). The term $\mathcal{O}(\mu^2)$ denotes a quantity dependent on the squared step sizes μ_k^2 such as

$$\mathcal{O}(\mu^2) = \mathbb{E}\{\boldsymbol{\mathcal{R}}_i \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{A}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{R}}_i\} \tag{2.41}$$

While it is possible to carry on the study considering the term $\mathcal{O}(\mu^2)$, we focus on the case where the step-sizes are small enough to ignore the terms involving their higher moments [Sayed, 2013a]. Therefore we continue the analysis with $\boldsymbol{\Sigma}'$ defined as

$$\boldsymbol{\Sigma}' = \boldsymbol{\mathcal{A}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{A}}^\top - \boldsymbol{\mathcal{A}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{A}}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{A}}^\top \tag{2.42}$$

The weighing matrix $\boldsymbol{\Sigma}'$ carries several information about the set-up. The matrix $\boldsymbol{\mathcal{A}}$ or $\boldsymbol{\mathcal{C}}$ defines the network topology, the data profile is encoded with the aggregated covariance matrix $\boldsymbol{\mathcal{R}}$ and the step-sizes through the matrix $\boldsymbol{\mathcal{M}}$.

As in [Sayed et al., 2013], we express the matrix $\boldsymbol{\Sigma}$ under its vector form such as $\boldsymbol{\sigma}' = \text{vec}(\boldsymbol{\Sigma})$. To do so, we use the following properties:

$$\text{vec}(\boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Z}) = (\boldsymbol{Z}^\top \otimes \boldsymbol{Q})\boldsymbol{\sigma} \tag{2.43}$$

$$\text{trace}(\boldsymbol{\Sigma}\boldsymbol{Q}) = [\text{vec}(\boldsymbol{Q}^\top)]^\top\boldsymbol{\sigma} \tag{2.44}$$

where $\boldsymbol{Q}$ and $\boldsymbol{Z}$ are arbitrary matrices of compatible sizes. Applying the vec(.) operator on both sides of (2.42) we find that

$$\boldsymbol{\sigma}' = (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}})\,\boldsymbol{\sigma} - (\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{\mathcal{A}})\,\boldsymbol{\sigma} - (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}})\,\boldsymbol{\sigma} \tag{2.45}$$

which leads to

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma} \tag{2.46}$$

where we introduce the $(NL)^2 \times (NL)^2$ matrix $\boldsymbol{\mathcal{F}}$ defined as

$$\boldsymbol{\mathcal{F}} = (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}}) - (\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{\mathcal{A}}) - (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}}) \tag{2.47}$$

Using the trace(.) and vec(.) properties on (2.38) and substituting it alongside (2.39) in (2.37) results in

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\sigma}} = \mathbb{E}\{||\widetilde{\boldsymbol{w}}_{i-1}||^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}\} + \left[\text{vec}\left(\boldsymbol{\mathcal{Y}}^\top\right)\right]^\top \boldsymbol{\sigma} \tag{2.48}$$

where

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{G}}^\top\boldsymbol{\mathcal{S}}\boldsymbol{\mathcal{G}} \tag{2.49}$$

From (2.48), the diffusion LMS is stable in the mean square sense if and only if the coefficient matrix $\boldsymbol{\mathcal{F}}$ is stable. This condition is satisfied when the step-sizes are sufficiently small

$$\mu_k < \frac{2}{\lambda_{max}(\boldsymbol{R}_k)} \tag{2.50}$$

where $\boldsymbol{R}_k$ is defined in (2.22).

2.3.4 Network Mean-Square Performance

Using the equation (2.48), we can evaluate the network as well as every single agent accuracy. As the algorithm is stable for sufficiently small step-sizes (2.50) we can take the limits of the equation (2.48) as

$$\lim_{i\to\infty} \mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\sigma}} = \lim_{i\to\infty} \mathbb{E}||\widetilde{\boldsymbol{w}}_{i-1}||^2_{\boldsymbol{\mathcal{F}\sigma}} + \mathrm{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \boldsymbol{\sigma} \tag{2.51}$$

Grouping terms leads to

$$\lim_{i\to\infty} \mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{(\boldsymbol{I}-\boldsymbol{\mathcal{F}})\boldsymbol{\sigma}} = \underbrace{\mathrm{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \boldsymbol{\sigma}}_{\text{Noise term}} \tag{2.52}$$

Consider the mean square deviation of the network defined as follows:

$$\mathrm{MSD}^{Network} = \tfrac{1}{N}||\widetilde{\boldsymbol{w}}_i||^2 \tag{2.53}$$

Estimating (2.53) can be performed by setting $\boldsymbol{\sigma}$ in (2.52) as follows:

$$\boldsymbol{\sigma} = \tfrac{1}{N}(\boldsymbol{I}-\boldsymbol{\mathcal{F}})^{-1}\mathrm{vec}(I) \tag{2.54}$$

This leads to:

$$\mathrm{MSD}^{Network} = \tfrac{1}{N}\mathrm{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top(\boldsymbol{I}-\boldsymbol{\mathcal{F}})^{-1}\mathrm{vec}(I) \tag{2.55}$$

2.3.5 Transient State Analysis

To analyse the algorithm transient behaviour, we iterate the equation (2.48) from $i = 0$ to find

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\sigma}} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{-1}||^2_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}} + \mathrm{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \sum_{j=0}^{i} \boldsymbol{\mathcal{F}}^j \boldsymbol{\sigma} \tag{2.56}$$

where $\widetilde{\boldsymbol{w}}_{-1}$ is the initial condition term. Comparing the recursion (2.56) for the time instants i and $i-1$ we find

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\sigma}} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{i-1}||^2_{\boldsymbol{\sigma}} + \mathrm{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \boldsymbol{\mathcal{F}}^i \boldsymbol{\sigma} - \mathbb{E}||\widetilde{\boldsymbol{w}}_{-1}||^2_{(\boldsymbol{I}-\boldsymbol{\mathcal{F}})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}} \tag{2.57}$$

In order to evaluate the transient Mean-Square Deviation, the weighing vector $\boldsymbol{\sigma}$ needs to be set to $\frac{1}{N}\mathrm{vec}(\boldsymbol{I})$.

Note that similar performance information can be extracted for a single agent by setting the weighting matrix $\boldsymbol{\Sigma}$ to zeros except for the block matrix corresponding to the said agent which shall be set to the identity matrix.

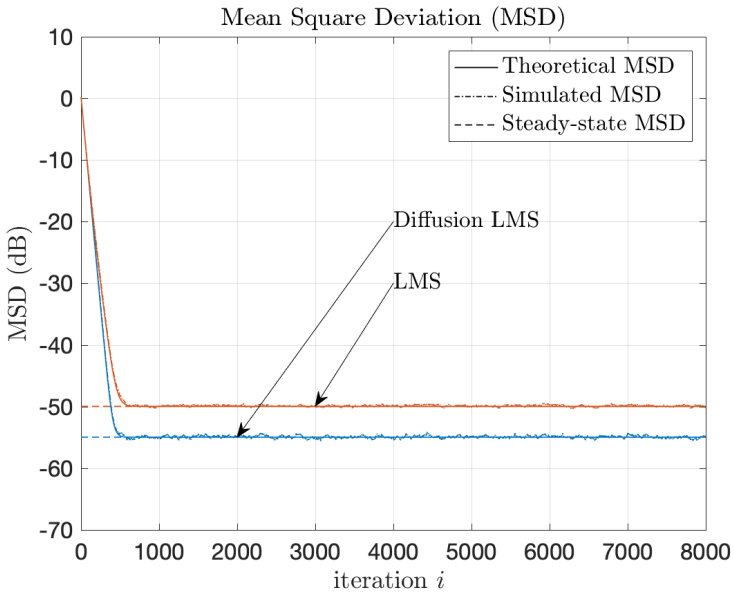

Figure 2.3: Comparison of the Mean square deviation (MSD) for *diffusion LMS* and single agent *LMS*. For this comparison, we considered a two dimensional objective vector we randomly drawn from a Gaussian distribution $\boldsymbol{w}^o \sim \mathcal{N}(0, \boldsymbol{I}_2)$. The network consists of $N = 10$ agents. We set step-size parameters $\mu_k = 10^{-2}$. We used Gaussian regression data $\boldsymbol{u}_{k,i} \sim \mathcal{N}(0, \boldsymbol{I}_2)$ and additive noise $v_k(i) \sim \mathcal{N}(0, 10^{-3})$.

2.4 Numerical examples

For the sake of demonstration, we consider a network of $N = 10$ agents estimating $\boldsymbol{w}^o$ a two dimensional vector. The regression data is randomly drawn from a Gaussian distribution $\boldsymbol{u}_{k,i} \sim \mathcal{N}(0, \boldsymbol{I}_2)$ and so is the additive noise $v_k(i) \sim \mathcal{N}(0, 10^{-3})$. We carry out 100 Monte-Carlo runs of the diffusion LMS algorithm.

In Figure 2.3 we illustrate the theoretical model accuracy and the diffusion LMS performance compared to a single agent LMS. Through collaboration, diffusion LMS algorithm achieves a better performance than the single agent LMS. Every agent has access not only to its measurements but also to its neighbours through the shared gradient and local estimates, which enhances the performance N folds.

In Figure 2.4, we depict the effect of the step-size parameters on the convergence rate and accuracy. If the steps sizes are small the accuracy is enhanced however, the algorithm converges slowly and vice-versa. This aspect is of high importance especially when comparing algorithms. In order to remain fair and consistent, both compared algorithms must have either the same convergence rate or final accuracy.

Finally in Figure 2.5, we illustrate the $2D$ local estimates $\boldsymbol{w}_{k,i}$ convergence trajectories towards the optimal solution $\boldsymbol{w}^o$. Note that all agents converge toward a temporary estimate then converge together to the right minimiser.

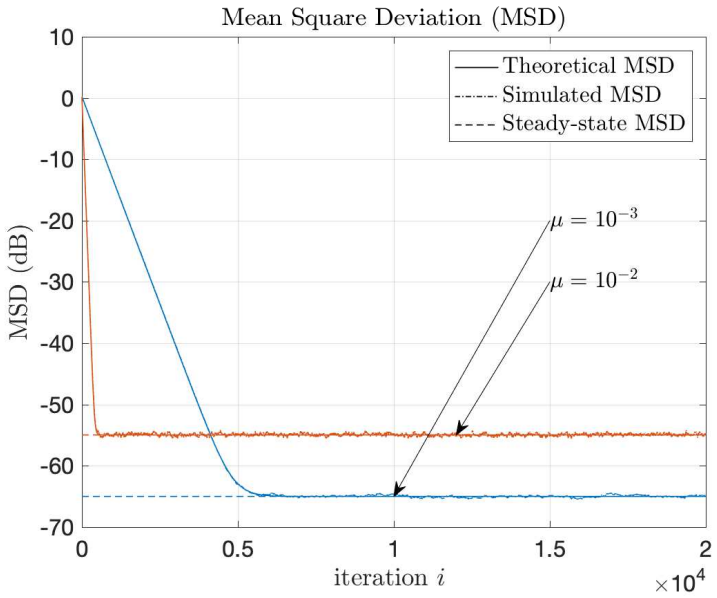

Figure 2.4: The effect of the step-size parameter on the convergence rate and the algorithm accuracy. For this experiment, we used the same data profile and $N = 10$ agents network as in the previous experiment. However, we considered two step sizes $\mu = 10^{-2}$ and $\mu = 10^{-3}$.

2.5 Recent developments of diffusion LMS

Diffusion LMS has been widely studied in the literature. Using the basic analysis provided above, we shall explore the different aspects studied in the state of the art.

Diffusion LMS over noisy communication links Since diffusion strategies heavily rely on communication between agents, it is of interest to study the effects of noisy communication links on the network performance. In [Khalili et al., 2012a, Khalili et al., 2012b], the authors analysed the steady state as well as the transient behaviour of the algorithm under these conditions. In order to model the noisy links, they introduced an additive zero mean i.i.d. noise in the adaptation step (2.4) so the modified version of the algorithm is formulated as such:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1}) + \boldsymbol{q}_i \tag{2.58}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_{\ell,i}. \tag{2.59}$$

where $\boldsymbol{q}_i$ is the additive i.i.d. noise. They concluded that diffusion LMS still converges toward the optimal solution. However, due the to noisy links, the mean square performance of the algorithm is severely impaired in comparison to diffusion LMS over perfect communication links.

In [Zhao et al., 2012], the authors went even further and considered noisy links for all the communications between every two agents ℓ and k as it is depicted in Figure 2.6. While in [Khalili et al., 2012b] the noisy links were introduced in the adaptation step of the algorithm, Zaho et al.

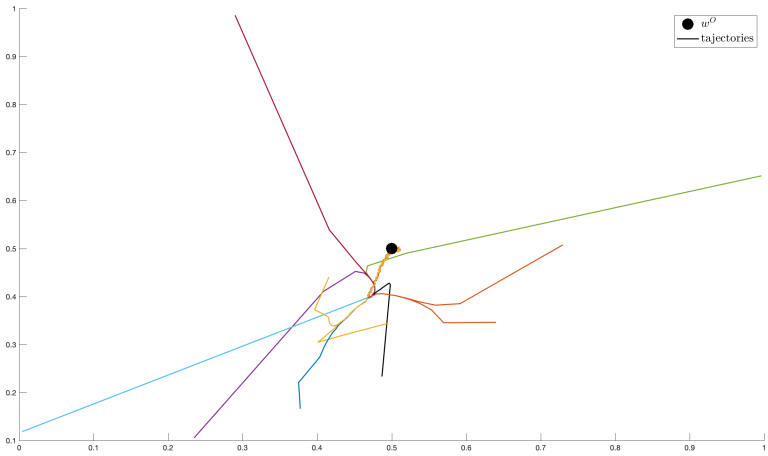

FIGURE 2.5: The trajectories taken by the local estimates $\boldsymbol{w}_{k,i}$ when converging towards the $2D$ optimum parameter vector $\boldsymbol{w}^o$. We used the same setting as the previous experiment with $\mu_k = 10^{-2}$. Note that in this experiment the agents first reached a consensus, then this consensus converged to the optimum solution.

introduced it directly in the transmitted quantities as it is shown below

$$\boldsymbol{w}_{\ell k,i} = \boldsymbol{w}_{\ell,i} + \boldsymbol{v}^{\boldsymbol{w}}_{\ell k,i} \tag{2.60}$$

$$\boldsymbol{\psi}_{\ell k,i} = \boldsymbol{\psi}_{\ell,i} + \boldsymbol{v}^{\boldsymbol{\psi}}_{\ell k,i} \tag{2.61}$$

$$\boldsymbol{u}_{\ell k,i} = \boldsymbol{u}_{\ell,i} + \boldsymbol{v}^{\boldsymbol{u}}_{\ell k,i} \tag{2.62}$$

$$d_{\ell k}(i) = d_{\ell}(i) + v^{d}_{\ell k}(i) \tag{2.63}$$

where the noises $\boldsymbol{v}^{\boldsymbol{w}}_{\ell k,i}$, $\boldsymbol{v}^{\boldsymbol{\psi}}_{\ell k,i}$, $\boldsymbol{v}^{\boldsymbol{u}}_{\ell k,i}$ and $v^{d}_{\ell k}(i)$ are assumed to be temporally white and spatially independent random processes with zero means.

After carrying out a theoretical analysis, the authors arrived to the conclusion that, as expected, the noisy links deteriorate the algorithm performance. Furthermore, the regressor data transmission noise $\boldsymbol{v}^{\boldsymbol{u}}_{\ell k,i}$ leads to a biased solution. Other related studies have been conducted such as [Takahashi et al., 2010] where adaptive combination rules were used to counteract the non stationarity or [Abdolee et al., 2016] where the authors investigated diffusion LMS in realistic scenarios by considering wireless networks with signal fading and path loss.

In the previous studies, the authors considered conditions where the agents respond to data synchronously. However, in several application agents may not have such luxury as they can be subject to different sources of uncertainty such as link failures or random topology changes. In [Zhao and Sayed, 2015a, Zhao and Sayed, 2015b], the authors used a random step size to model link failures and random combination weights to model random topology changes. As in the previous studies the authors theoretically and numerically analysed the algorithm performance.

Sparse estimation Among the many attractive features of diffusion strategies is their versatility. They can be adapted for different applications just by selecting an appropriate cost function or

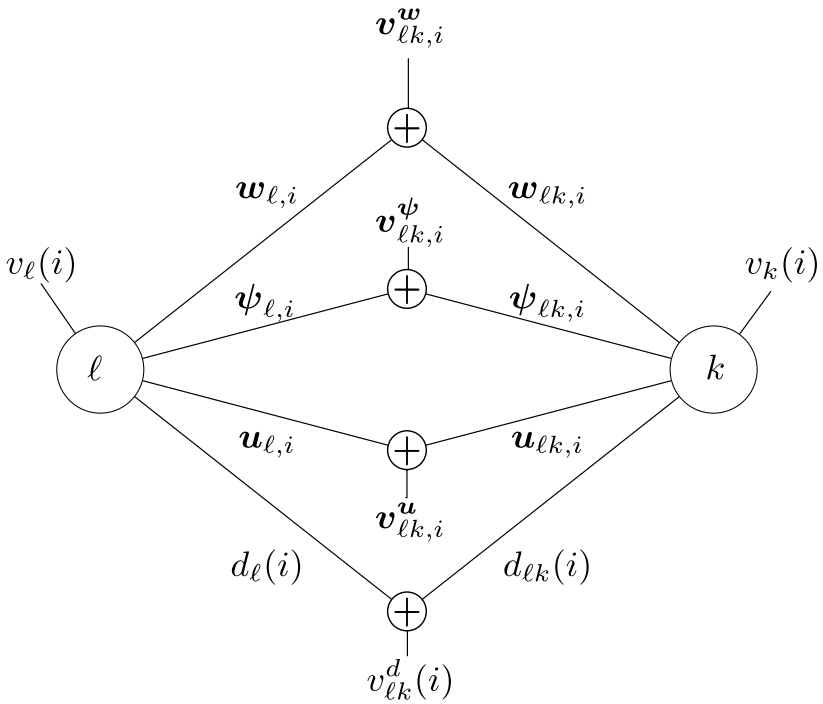

Figure 2.6: Imperfect communication links representation between a source agent ℓ and a sink agent k [Sayed, 2013a]. There are four types of noises: 1) regression data noise $\boldsymbol{v}^{\boldsymbol{u}}_{\ell k,i}$, 2) reference signal noise $v^d_{\ell k}$, 3) local estimate noise $\boldsymbol{v}^{\boldsymbol{\psi}}_{\ell k,i}$ and weight vector noise $\boldsymbol{v}^{\boldsymbol{w}}_{\ell k,i}$.

regularisation. For instance, in [Chen and Sayed, 2012] authors considered a second-order approximation of the cost function to ease the algorithm analysis. They also provided application examples. The first one consisted in estimating a sparse vector. To consider the data sparsity, they used a regularisation function $\Omega(\boldsymbol{w})$ and parameter ρ such as:

$$J_k(\boldsymbol{w}) = \mathbb{E}||d_{k,i} - \boldsymbol{u}_{k,i}^\top \boldsymbol{w}||^2 + \tfrac{\rho}{N}\Omega(\boldsymbol{w}) \tag{2.64}$$

where

$$\Omega(\boldsymbol{w}) = \sum_{\ell}^{L} \sqrt{|[w]_m| + \epsilon} \tag{2.65}$$

with ϵ a very small quantity. Note that $\Omega(w)$ is an approximation of $||\boldsymbol{w}||_1$. The authors also considered the non-convex location problem. In this case the global optimization function is formulated as

$$J^{glob}(w) = \tfrac{1}{4}\textstyle\sum_k^N \mathbb{E}\{|d_k(i) - ||w - \boldsymbol{x}_k||_2^2|^2\} \tag{2.66}$$

where $\boldsymbol{x}_k$ is node's k position vector.

Besides [Chen and Sayed, 2012] other authors tackled the sparse optimization problem. In [Liu et al., 2012], the authors incorporated the ℓ_1 and ℓ_0 norms into the cost function to promote sparsity while the set-theoretic estimation rationale was used in [Chouvardas et al., 2012] for the same purpose. There are many other variants of cost functions and regularisers promoting data sparsity [Di Lorenzo and Sayed, 2013, Wen and Liu, 2015]. Further details can be found in [Gharehshiran et al., 2013, Chouvardas et al., 2011] and the references therein.

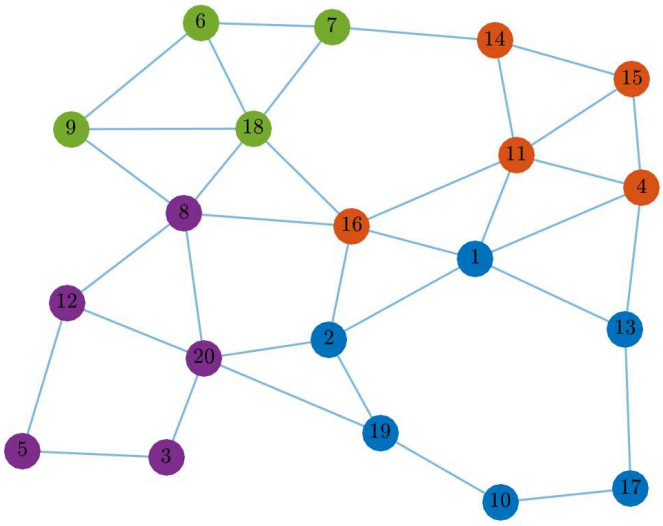

Figure 2.7: A multitask network of $N = 20$ agents and $Q = 4$ tasks. The clusters are defined by each node color. For instance the agents $1, 2, 10, 13, 17, 19$ form the blue cluster.

Compression in diffusion LMS With the over increasing data dimension and the limitation of communicating networks, the community took interest in data compression to avoid network congestion and reach the full potential of diffusion strategies. There are mainly two approaches, the first one consists of restricting the number of agents which each agent communicates with [Lopes and Sayed, 2008, Arablouei et al., 2015] while the second approach limits the transferred data [Sayin and Kozat, 2014, Arablouei et al., 2014b, Arablouei et al., 2014a, Vadidpour et al., 2015]. This aspect of diffusion LMS shall be further discussed in chapter 3.

Structured Data in diffusion LMS Due to the recent surge of IOT networks, data is characterised by complex structures that could sometimes be captured by a graph representation. While it is still possible to process the data without considering their inherent structure, it is more beneficial to consider a graph centric method where the data structure is preserved and used to fully analyse the data. In [Nassif et al., 2017a], the authors propose a step by step method to blend concepts from adaptive networks and graph signal processing. As a first step they propose a centralized adaptive method for streaming graph signals based on LMS strategy. Then they provide a way to distribute it over graph nodes using diffusion adaptation over networks concept. Finally, they carry out a theoretical analysis of the proposed method and verify the model accuracy through numerical experimentation.

Multitask Diffusion LMS In some cases we are interested in estimating different parameters that are not necessarily common to all the agents. In order to accomplish that, some authors explored diffusion LMS for networks seeking to minimize different cost functions. It seems counter productive in the beginning as such setting will lead to a biased solution. However, in hindsight, this approach can be beneficial when agents share similar objectives in the same network. As for

the single task diffusion LMS, multitask diffusion LMS has been studied under different conditions. For instance in [Nassif et al., 2016c] the authors considered an asynchronous network and in [Nassif et al., 2016e] used a regularization term to promote sparsity. It is worth mentioning that multitask learning has been first considered in a single agent setting in [Caruana, 1997, Argyriou et al., 2007, Evgeniou and Pontil, 2004a] to cite a few.

In order for multitask diffusion strategy to succeed, it is necessary for the agents to know which neighbours share the same objective. This case has been considered in [Nassif et al., 2017b, Nassif et al., 2016c, Nassif et al., 2016e, Hua et al., 2017, Chen and Sayed, 2013a]. Sometimes this information is available, in this case multitask diffusion LMS converges towards an unbiased optimal solution [Chen and Sayed, 2013b]. Otherwise a clustering strategy is needed. In the absence of such strategy, the network will converge to a Pareto optimal solution [Chen and Sayed, 2013b]. We shall further discuss this aspect of diffusion LMS in chapter 5.

In this section we gave a bird eyes view of diffusion LMS with a focus on the most relevant topics for the scope of this thesis. However, it goes without saying that diffusion LMS field is much wider than what it is depicted above. For instance, in [Chainais and Richard, 2013] authors considered distributed dictionary learning and in [Chen et al., 2014a] considered multitask diffusion LMS with node hypothesis spaces partly overlapping, to mention a few. In the following chapters we will extend some relevant parts of the state of the art and build upon the concepts introduced in this chapter to analyse the newly proposed algorithms.

3 Compressed Diffusion LMS

Contents

The world has become more connected and networked as a consequence of the recent rise of digital and mobile communications. This evolution has resulted in an unprecedented volume of data flowing between sources, data centers, or processes. While this data may be processed in a centralized manner, it is often more suitable to consider distributed strategies such as diffusion LMS as they are scalable and can handle large amounts of data by distributing tasks over networked agents. Although it is relatively simple to implement diffusion strategies over a cluster, it appears to be challenging to deploy them in an ad-hoc network with limited energy budget for communication. In this chapter, we introduce a diffusion LMS strategy that significantly reduces communication costs without compromising the performance. Then, we analyse the proposed algorithm in the mean and mean-square sense. Next, we conduct numerical experiments to verify the theoretical findings. Finally, we perform large scale simulations to test the algorithm efficiency in a scenario where energy is limited.

The work presented in this chapter was published in:

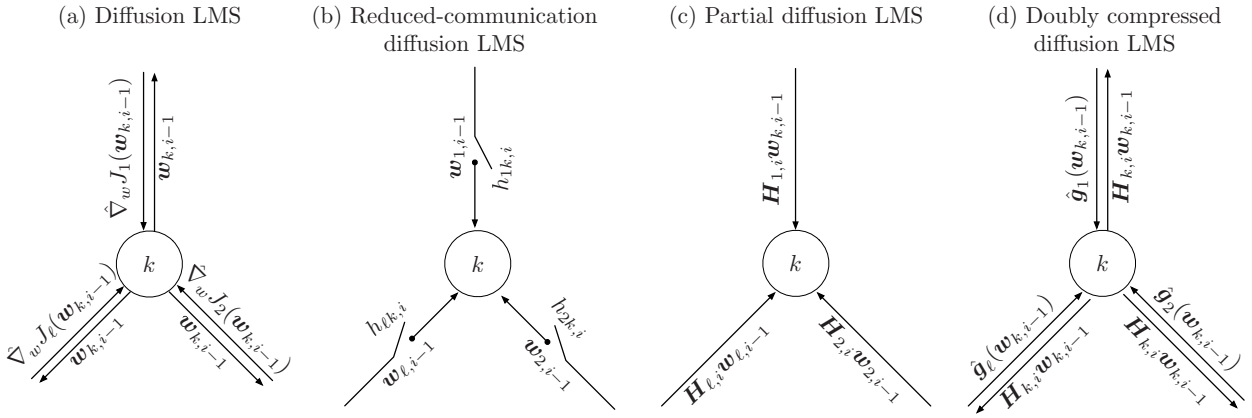

Figure 3.1: Illustrative representation of transmitted data for the diffusion LMS and different approaches aiming at reducing the communication load for a node k. Part (a) represents the communication for diffusion LMS where the weighting matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ are different from the identity matrix. In part (b), we illustrate the communication link between agents as proposed in [Lopes and Sayed, 2008]. The authors proposed to share data with only a random subset of their neighbours. Other authors considered to limit the number of shared entries rather than agents as it is depicted in part (c). This approaches was proposed in [Arablouei et al., 2014a]. Note that for both parts (b) and (c) the weighting matrix $\boldsymbol{C}$ is set to the identity. Finally, part (d) represents the approach we adopted in this study, that is, to only share partial stochastic gradient vectors. To do so, we set the weighting matrix $\boldsymbol{A} = \boldsymbol{I}_N$ and use selection matrices $\boldsymbol{H}_{k,i}$ and $\boldsymbol{Q}_{k,i}$ as explained hereafter.

- Harrane, I. E. K., Flamary, R., and Richard, C. (2019). On reducing the communication cost of the diffusion lms algorithm. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):100–112.
- Harrane, I. E. K., Flamary, R., and Richard, C. (2016a). Doubly compressed diffusion lms over adaptive networks. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 987–991.

3.1 Introduction

In the recent decade the world has witnessed an unprecedented evolution of networked devices giving birth to what is known now as Internet of Things or IOT. Such advancement opened new horizons to diffusion strategies. However it has also opened doors to new challenges. Indeed, as illustrated in Figure 3.1 (a), due to constant communication required by diffusion strategies, all nodes need to exchange information with their neighbours at each iteration. In the case of diffusion LMS, as will be detailed in the next section, this information can be either local estimates and gradients of local cost functions, or local estimates only. Even in the latter case, this requirement imposes a substantial burden on communication and energy resources. Reducing the communication cost while maintaining the benefits of cooperation is therefore of major importance for systems with limited energy budget such as wireless sensor networks.

In the recent years, several strategies were proposed to address this issue. As stated in section 2.5, there are mainly two approaches which we illustrate in Fig. 3.1 (b) and (c). On the one hand, some authors proposed to restrict the number of active links between neighbouring nodes at each time instant [Lopes and Sayed, 2008, Arablouei et al., 2015]. If we take [Arablouei et al., 2015] as an example, the equations (2.4) and (2.5) would be reformulated as:

$$\begin{cases} \boldsymbol{\psi}_{k,i} & = \boldsymbol{w}_{k,i-1} + \mu_k \boldsymbol{u}_{k,i} \left(d_{k,i} - \boldsymbol{u}_{k,i}^{\top} \boldsymbol{w}_{k,i-1} \right) \\ \boldsymbol{w}_{k,i} & = a_{kk} \boldsymbol{\psi}_{k,i} + \sum\limits_{\ell \in \mathcal{N}_k / k} a_{\ell k} \left[h_{\ell k,i} \boldsymbol{\psi}_{\ell,i} + (1 - h_{\ell k,i}) \boldsymbol{\psi}_{k,i} \right] \end{cases} \tag{3.1}$$

where $h_{\ell k}$ is a binary random variable with the probability of it being equal to one defined as

$$P(h_{\ell k} = 1) = \frac{M}{L} \tag{3.2}$$

where M is the number of selected communicating neighbours. Note that such distribution does not ensure M communicating neighbours for every iteration i. This condition is only verified in expectation.

On the other hand, there are authors that recommend to reduce the communication load by transmitting only partial parameter vectors [Arablouei et al., 2014b, Arablouei et al., 2014a, Vadidpour et al., 2015]. Under such conditions diffusion LMS (2.4)–(2.5) is reformulated as:

$$\begin{cases} \boldsymbol{\psi}_{k,i} & = \boldsymbol{w}_{k,i-1} + \mu_k \boldsymbol{u}_{k,i} \left(d_{k,i} - \boldsymbol{u}_{k,i}^{\top} \boldsymbol{w}_{k,i-1} \right) \\ \boldsymbol{w}_{k,i} & = \sum\limits_{\ell \in \mathcal{N}_k} a_{\ell k} \left[\boldsymbol{H}_{\ell,i} \boldsymbol{\psi}_{\ell,i} + (\boldsymbol{I}_L - \boldsymbol{H}_{\ell,i}) \boldsymbol{\psi}_{k,i} \right] \end{cases} \tag{3.3}$$

in this case $\boldsymbol{H}_{\ell,i}$ is a diagonal selection matrix where its entries are Bernoulli random variables defined similarly to $h_{\ell k}$ with a slight difference as M in this case, is the number of selected entries. Note that again, this method does not guaranty M transferred entry at each iteration i because of the nature of the chosen distribution law for the selection matrix $\boldsymbol{H}_{\ell,i}$. Another similar method has been adopted in [Sayin and Kozat, 2014] where the compression is accomplished by projecting parameter vectors onto lower dimensional spaces before transmission such as:

$$\begin{cases} \boldsymbol{\psi}_{k,i} & = \boldsymbol{w}_{k,i-1} + \mu_k \boldsymbol{u}_{k,i} \left(d_{k,i} - \boldsymbol{u}_{k,i}^{\top} \boldsymbol{w}_{k,i-1} \right) \\ \boldsymbol{w}_{k,i} & = a_{kk} \boldsymbol{\psi}_{k,i} + \sum\limits_{\ell \in \mathcal{N}_k / k} a_{\ell k} \boldsymbol{\gamma}_{\ell,i} \end{cases} \tag{3.4}$$

where $\gamma_{\ell,i} = \boldsymbol{\gamma}_{\ell,i-1} + \eta_\ell \boldsymbol{c}_{\ell,i} h(\epsilon_{\ell,i})$ is the constructed estimate of $\boldsymbol{\psi}_{\ell,i}$ when $\ell \neq k$, with η_ℓ a positive step-size, $c_{\ell,i}$ a projection vector and $\epsilon_{\ell,i} = \boldsymbol{c}_{\ell,i}^{\top} (\boldsymbol{\psi}_{\ell,i} - \boldsymbol{\gamma}_{\ell,i})$ is the construction error. Depending of $h(\epsilon_{\ell,i})$, $\gamma_{\ell,i}$ can be either a compressed version or a single bit representation of $\boldsymbol{\psi}_{\ell,i}$.

$$h(\epsilon_{\ell,i}) = \begin{cases} \epsilon_{\ell,i} & \text{for a compressed version of } \boldsymbol{\psi}_{\ell,i} \\ h(\epsilon_{\ell,i}) & \text{for a single bit representation of } \boldsymbol{\psi}_{\ell,i} \end{cases} \tag{3.5}$$

These ideas are related to what is known in the literature as coordinate-descent constructions for

Table 3.1: List of the symbols and notations used in chapter 3

Symbol	Definition
L	length of the parameter vectors
N	network agent count
$\mathcal{N}_k$	neighbourhood of the agent k including it self
$\boldsymbol{w}_{k,i}$	instantaneous estimate at the agent k
$\boldsymbol{w}^o$	optimum parameter vector
$d_k(i)$	reference signal for the agent k at the time instant i
$\boldsymbol{u}_{k,i}$	regression vector of the agent k
$v_k(i)$	additive noise at the agent k
M	number of shared entries of the vector $\boldsymbol{w}_{k,i}$
M_∇	number of shared entries of the stochastic gradient vector $\hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1})$

single agent optimization. Note that they have recently been extended to distributed settings in [Necoara et al., 2017, Xi and Khan, 2017]. They have also been applied to general diffusion networks in [Wang et al., 2018] for general convex cost functions, with a detailed analysis of the performance and stability of the resulting network. In that paper, the authors consider the diffusion LMS and assume that the adaptation step at each node has only access to a random subset of the entries of the approximate gradient vector. At each node, however, all the entries of the local estimates of the neighbouring nodes remain available for the combination step.

With the exception of some few papers such as [Wang et al., 2018], the literature mainly focused on the case where the nodes only share a subset of the entries of their local estimates. Nevertheless, it is also of interest to consider the case where both the local estimates and the approximate gradient vectors of the local cost functions are partially shared. This situation may arise due to missing entries. Such schemes are also useful, as considered in this work, in reducing communication cost at each iteration in large scale data applications.

This chapter is structured in two parts. In the first part, we focus on the compression aspect by defining the probability distribution function used for the entries selection. Then, as a first step, to ease the theoretical analysis, we shall propose a partially compressed diffusion LMS where the local estimate vectors $\boldsymbol{w}_{k,i}$ are partially shared while the stochastic gradient vectors are fully transmitted. To do so, we use a selection matrix (which we define in the sequel) to choose a random subset of entries to share. Next, we introduce and theoretically analyse the fully compressed version of diffusion LMS. This time, both local estimate and stochastic gradient vectors are partially shared.

The second part consists of a numerical analysis of both the proposed algorithms to validate the theoretical models as a first objective and to compare them to the state of the art algorithms. We also analyse the algorithms under a realistic scenario where energy resources are scarce.

3.1.1 Problem formulation

Consider a connected network composed of N nodes. The aim of each node is to estimate an $L \times 1$ unknown parameter vector $\boldsymbol{w}^o$ from collected measurements. Node k has access to local streaming measurements $\{d_k(i), \boldsymbol{u}_{k,i}\}$ where $d_k(i)$ is a scalar zero-mean reference signal, and $\boldsymbol{u}_{k,i}$ is an $L \times 1$

zero-mean regression vector with a positive definite covariance matrix $\boldsymbol{R}_{u_k} = \mathbb{E}\{\boldsymbol{u}_{k,i}\boldsymbol{u}_{k,i}^{\top}\}$. The data at agent k and time i are assumed to be related via the linear regression model:

$$d_k(i) = \boldsymbol{u}_{k,i}^{\top}\boldsymbol{w}^o + v_k(i) \tag{3.6}$$

where $\boldsymbol{w}^o$ is the unknown parameter vector to be estimated, and $v_k(i)$ is a zero-mean i.i.d. noise with variance $\sigma_{v,k}^2$. The noise $v_k(i)$ is assumed to be independent of any other signal. Let $J_k(\boldsymbol{w})$ be a differentiable convex cost function at agent k. In this chapter, we shall consider the mean-square-error criterion, namely,

$$J_k(\boldsymbol{w}) = E\{|d_k(i) - \boldsymbol{u}_{k,i}^{\top}\boldsymbol{w}|^2\} \tag{3.7}$$

This criterion is strongly convex, second-order differentiable, and minimized at $\boldsymbol{w}^o$.

We seek to collaboratively minimize the aggregate cost function in a distributed manner:

$$J^{\text{glob}}(\boldsymbol{w}) = \sum_{k=1}^{N} J_k(\boldsymbol{w}) \tag{3.8}$$

In addition, we also require to minimise the network communication load. To do so, we consider a compressed version of diffusion LMS that we shall introduce in upcoming section.

3.2 Diffusion LMS with compression

In this part, we shall introduce and study both the compressed versions of diffusion LMS then theoretically analyse them. Before proceeding with the analysis, we study the moments of the probability distribution law we shall use for the proposed algorithms. Such result will immensely ease the theoretical study.

3.2.1 Selection matrix probability distribution

We shall now introduce and study the random selection moments which will be used later in the theoretical analysis. The selection matrix denoted by $\boldsymbol{H}_{\ell,i} = \text{diag}\{\boldsymbol{h}_{\ell,i}\}$ is a diagonal matrix with binary entries. It can be characterized by two parameters: its dimension L and M the number of the non zero entries in its diagonal. The entries of $\boldsymbol{H}_{\ell,i}$ are assumed to be equally likely and i.i.d over time and space. The random vector $\boldsymbol{h}_{\ell,i}$ is drawn as

$$\boldsymbol{h}_{\ell,i} \sim \left\{ \boldsymbol{h}_{\ell,i} \in \{0,1\}^L, \sum_{j=1}^{L} h_{\ell,i_j} = M \right\} \tag{3.9}$$

where h_{ℓ,i_j} is the j^{th} entry of the vector $\boldsymbol{h}_{\ell,i}$. Definition (3.9) leads to the following results:

$$P(h_{\ell,i} = 1) = \tfrac{M}{L} \tag{3.10}$$

$$\mathbb{E}\{\boldsymbol{H}_{\ell,i}\} = \tfrac{M}{L}\boldsymbol{I}_L \tag{3.11}$$

Given a positive definite $L \times L$ matrix $\boldsymbol{\Sigma}$ we have:

- For $\ell \neq k$: since the matrices $\boldsymbol{H}_{\ell,i}$ and $\boldsymbol{H}_{k,i}$ are assumed to be independent we have:

$$\begin{aligned}\mathbb{E}\{\boldsymbol{H}_{\ell,i}\boldsymbol{\Sigma}\boldsymbol{H}_{k,i}\} &= \mathbb{E}\{\boldsymbol{H}_{\ell,i}\}\boldsymbol{\Sigma}\mathbb{E}\{\boldsymbol{H}_{k,i}\} \\ &= \left(\tfrac{M}{L}\right)^2 \boldsymbol{\Sigma}\end{aligned} \tag{3.12}$$

- When $k = \ell$

$$\begin{aligned}\mathbb{E}\{\boldsymbol{H}_{\ell,i}\boldsymbol{\Sigma}\boldsymbol{H}_{\ell,i}\} &= \mathbb{E}\{\boldsymbol{\Sigma} \odot (\boldsymbol{h}_{\ell,i}\boldsymbol{h}_{\ell,i}^\top)\} \\ &= \boldsymbol{\Sigma} \odot \mathbb{E}\{(\boldsymbol{h}_{\ell,i}\boldsymbol{h}_{\ell,i}^\top)\}\end{aligned} \tag{3.13}$$

  where $\odot$ denotes Hadamard product. Evaluating $\mathbb{E}\{\boldsymbol{h}_{\ell,i}\boldsymbol{h}_{\ell,i}^\top\}$ leads to:

$$\mathbb{E}\{\boldsymbol{h}_{\ell,i}\boldsymbol{h}_{\ell,i}^\top\}_{mn} = \mathbb{E}\{h_m h_n\} \tag{3.14}$$

  – for $m = n$

$$\begin{aligned}\mathbb{E}\{\boldsymbol{h}_{\ell,i}\boldsymbol{h}_{\ell,i}^\top\}_{mm} &= \mathbb{E}\{h_m h_m\} \\ &= P(h_m = 1) + 0P(h_m = 0) \\ &= \tfrac{\mathrm{C}_{L-1}^{M-1}}{\mathrm{C}_L^M} \\ &= \tfrac{M}{L}\end{aligned} \tag{3.15}$$

  – in the case where $m \neq n$

$$\begin{aligned}\mathbb{E}\{\boldsymbol{h}_{\ell,i}\boldsymbol{h}_{\ell,i}^\top\}_{mn} &= \mathbb{E}\{h_m h_n\} \\ &= P(h_n = 1|h_m = 1) + 0(P(h_n = 1|h_m = 0) \\ &+ P(h_n = 0|h_m = 1) + P(h_n = 0|h_m = 0))\end{aligned} \tag{3.16}$$

$$\begin{aligned}\mathbb{E}\{\boldsymbol{h}_{\ell,i}\boldsymbol{h}_{\ell,i}^\top\}_{mn} &= P(h_n = 1|h_m = 1) = \tfrac{\mathrm{C}_{L-2}^{M-2}}{\mathrm{C}_L^M} \\ &= \tfrac{M(M-1)}{L(L-1)}\end{aligned} \tag{3.17}$$

  using (3.15) and (3.17) in (3.13) we get:

$$\begin{aligned}\mathbb{E}\{\boldsymbol{H}_{\ell,i}\boldsymbol{\Sigma}\boldsymbol{H}_{\ell,i}\} &= \tfrac{1}{\mathrm{C}_L^M}\left(\mathrm{C}_{L-1}^{M-1}\boldsymbol{I}_L \odot \boldsymbol{\Sigma} + \mathrm{C}_{L-2}^{M-2}(\boldsymbol{\Sigma} - \boldsymbol{I}_N \odot \boldsymbol{\Sigma})\right) \\ &= \tfrac{M}{L}\left(\left(1 - \tfrac{M-1}{L-1}\right)\boldsymbol{I}_L \odot \boldsymbol{\Sigma} + \tfrac{M-1}{L-1}\boldsymbol{\Sigma}\right)\end{aligned} \tag{3.18}$$

Finally we can sum up the results as:

$$\mathbb{E}\{\boldsymbol{H}_{\ell,i}\boldsymbol{\Sigma}\boldsymbol{H}_{k,i}\} = \begin{cases} \frac{M}{L}\left(\left(1-\frac{M-1}{L-1}\right)\boldsymbol{I}_L \odot \boldsymbol{\Sigma} + \frac{M-1}{L-1}\boldsymbol{\Sigma}\right) & \text{if } \ell = k \\ \left(\frac{M}{L}\right)^2 \boldsymbol{\Sigma} & \text{otherwise} \end{cases} \tag{3.19}$$

For an in depth analysis of the proposed algorithms, we will need to extend the results found in (3.19) to block diagonal matrices following two configurations. For that, we consider an arbitrary $(NL \times NL)$ matrix $\boldsymbol{\Pi}$. Furthermore we define the block matrix $\boldsymbol{\mathcal{H}}_i$ as follows:

$$\boldsymbol{\mathcal{H}}_i = \text{diag}\{\boldsymbol{H}_{1,i}, \boldsymbol{H}_{2,i}, \ldots, \boldsymbol{H}_{N,i}\} \tag{3.20}$$

Let us start with a simple case $E\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\Pi}\boldsymbol{\mathcal{H}}_i\}$. We start writing the expectation in its block matrix form:

$$\mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\Pi}\boldsymbol{\mathcal{H}}_i\}_{k\ell} = \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\Pi}]_{k\ell}\boldsymbol{H}_{\ell,i}\} \tag{3.21}$$

According to (3.19), we have two cases namely $k = \ell$ and $k \neq \ell$. For the first one we have:

$$\begin{aligned} \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\Pi}\boldsymbol{\mathcal{H}}_i\}_{kk} &= \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\Pi}]_{kk}\boldsymbol{H}_{k,i}\} \\ &= \frac{M}{L}\left(\left(1-\frac{M-1}{L-1}\right)\boldsymbol{I}_L \odot [\boldsymbol{\Pi}]_{kk} + \frac{M-1}{L-1}[\boldsymbol{\Pi}]_{kk}\right) \end{aligned} \tag{3.22}$$

For $k \neq \ell$ we find:

$$\begin{aligned} \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\Pi}\boldsymbol{\mathcal{H}}_i\}_{k\ell} &= \mathbb{E}\{\boldsymbol{H}_{k,i}\}[\boldsymbol{\Pi}]_{k\ell}\mathbb{E}\{\boldsymbol{H}_{\ell,i}\} \\ &= \left(\frac{M}{L}\right)^2 [\boldsymbol{\Pi}]_{k\ell} \end{aligned} \tag{3.23}$$

These results can be rearranged into a compact form using Hadamard product:

$$\begin{aligned} \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\Pi}\boldsymbol{\mathcal{H}}_i\} &= \frac{M}{L}\left(\left(1-\frac{M-1}{L-1}\right)\boldsymbol{I}_{NL} \odot \boldsymbol{\Pi} + \frac{M-1}{L-1}(\boldsymbol{I}_N \otimes \boldsymbol{1}_{LL}) \odot \boldsymbol{\Pi}\right) \\ &+ \left(\frac{M}{L}\right)^2 (\boldsymbol{\Pi} - (\boldsymbol{I}_N \otimes \boldsymbol{1}_{LL}) \odot \boldsymbol{\Pi}) \\ &= \frac{M}{L}\left(1-\frac{M-1}{L-1}\right)\boldsymbol{I}_{NL} \odot \boldsymbol{\Pi} + \frac{M}{L}\left(\frac{M-1}{L-1} - \frac{M}{L}\right)(\boldsymbol{I}_N \otimes \boldsymbol{1}_{LL}) \odot \boldsymbol{\Pi} + \left(\frac{M}{L}\right)^2 \boldsymbol{\Pi} \end{aligned} \tag{3.24}$$

For ease of use we define:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\mathcal{H}}}(\boldsymbol{\Pi}) &= \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\Pi}\boldsymbol{\mathcal{H}}_i\} \\ &= \beta_1 (\boldsymbol{I}_N \otimes \boldsymbol{1}_{LL}) \odot \boldsymbol{\Pi} + \beta_2 \boldsymbol{I}_{NL} \odot \boldsymbol{\Pi} + \beta_3 \boldsymbol{\Pi} \end{aligned} \tag{3.25}$$

where

$$\beta_1 = \frac{M}{L}\left(\frac{M-1}{L-1} - \frac{M}{L}\right) \tag{3.26}$$

$$\beta_2 = \frac{M}{L}\left(1-\frac{M-1}{L-1}\right) \tag{3.27}$$

$$\beta_3 = \left(\frac{M}{L}\right)^2 \tag{3.28}$$

When analysing the algorithm, we will be confronted to a second type of expectation involving the matrix $\boldsymbol{H}_i$ which is denoted by:

$$[\boldsymbol{\varphi}_H(\boldsymbol{\Pi})]_{k\ell} = \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\Pi}]_{k\ell}\boldsymbol{H}_{k,i}\} \tag{3.29}$$

In this case we are interested in evaluating the diagonal and non diagonal entries of each block matrix separately regardless of the block matrix situation with regards to the diagonal :

$$[\boldsymbol{\varphi}_H(\boldsymbol{\Pi})]_{k\ell} = \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\Pi}]_{k\ell}\boldsymbol{H}_{k,i}\} \tag{3.30}$$

For the diagonal entries of each block matrix we have:

$$[\boldsymbol{\varphi}_h(\boldsymbol{\Pi})]_{k\ell,mm} = \tfrac{M}{L}[\boldsymbol{\Pi}]_{k\ell,mm} \tag{3.31}$$

where the subscript $k\ell, mm$ refers to (m, m) entry of the matrix block $k\ell$. For the rest of the entries we find:

$$[\boldsymbol{\varphi}_H(\boldsymbol{\Pi})]_{k\ell,mn} = \tfrac{M(M-1)}{L(L-1)}[\boldsymbol{\Pi}]_{k\ell,mn} \tag{3.32}$$

Similarly to $\mathbb{E}_{\boldsymbol{\mathcal{H}}}$, we can rewrite the results under a compact form using Hadamard product

$$\begin{aligned}
\boldsymbol{\varphi}_H(\boldsymbol{\Pi}) &= \tfrac{M}{L}(\mathbf{1}_{NN} \otimes \boldsymbol{I}_{LL}) \odot \boldsymbol{\Pi} + \tfrac{M(M-1)}{L(L-1)}(\boldsymbol{\Pi} - (\mathbf{1}_{NN} \otimes \boldsymbol{I}_{LL}) \odot \boldsymbol{\Pi}) \\
&= \tfrac{M}{L}\left(1 - \tfrac{M-1}{L-1}\right)(\mathbf{1}_{NN} \otimes \boldsymbol{I}_{LL}) \odot \boldsymbol{\Pi} + \tfrac{M(M-1)}{L(L-1)}\boldsymbol{\Pi} \\
&= \beta_2\,(\mathbf{1}_{NN} \otimes \boldsymbol{I}_{LL}) \odot \boldsymbol{\Pi} + (\beta_1 + \beta_3)\,\boldsymbol{\Pi}
\end{aligned} \tag{3.33}$$

Note that in the case where the matrix $\boldsymbol{\Pi}$ is block diagonal the operators $\boldsymbol{\varphi}(.)$ and $\mathbb{E}_{\boldsymbol{\mathcal{H}}}(.)$ are equivalent:

$$\begin{aligned}
[\mathbb{E}_{\boldsymbol{\mathcal{H}}}(\boldsymbol{\Pi})]_{kk} &= \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\Pi}]_{kk}\boldsymbol{H}_{k,i}\} = \mathbb{E}\{\boldsymbol{H}_{\ell,i}[\boldsymbol{\Pi}]_{kk}\boldsymbol{H}_{\ell,i}\} \\
&= [\boldsymbol{\varphi}(\boldsymbol{\Pi})]_{kk} \\
\mathbb{E}_{\boldsymbol{\mathcal{H}}}(\boldsymbol{\Pi}) &= \boldsymbol{\varphi}(\boldsymbol{\Pi})
\end{aligned} \tag{3.34}$$

In Table 3.2 we sum up the results. For $\boldsymbol{\mathcal{H}}_i = \mathrm{diag}\{\boldsymbol{H}_{1,i}, \boldsymbol{H}_{2,i}, \ldots, \boldsymbol{H}_{N,i}\}$ where $\boldsymbol{H}_{\ell,i} = \mathrm{diag}\{\boldsymbol{h}_{\ell,i}\}$, $\boldsymbol{h}_{\ell,i}$ defined in (3.9) and $\boldsymbol{\Sigma}$ and $\boldsymbol{\Pi}$ two arbitrary matrices of dimensions $L \times L$ and $(L \times N) \times (L \times N)$, respectively.

3.2.2 Compressed Diffusion LMS

We shall now introduce our first version of *Compressed diffusion LMS* (*CD*) and study its stochastic behaviour which is an intermediate step towards the *Doubly Compressed Diffusion LMS* (*DCD*). The DC run at each node k is described in 3.1. We consider the same conditions as for diffusion LMS except for the matrices $\boldsymbol{A}$ and $\boldsymbol{C}$. For the sake of simplicity, we choose the matrix $\boldsymbol{C}$ to be

Table 3.2: Moments of the selection matrix

Moment	Expression
$\mathbb{E}\{\boldsymbol{H}_{\ell,i}\}$	$\frac{M}{L}\boldsymbol{I}_L$
$\mathbb{E}\{\boldsymbol{H}_\ell \boldsymbol{\Sigma} \boldsymbol{H}_k\}$	$\begin{cases} \frac{M}{L}\left(\left(1-\frac{M-1}{L-1}\right)\boldsymbol{I}_L \odot \boldsymbol{\Sigma} + \frac{M-1}{L-1}\boldsymbol{\Sigma}\right) & \text{if } \ell = k \\ \left(\frac{M}{L}\right)^2 \boldsymbol{\Sigma} & \text{otherwise} \end{cases}$
$\mathbb{E}\{\boldsymbol{\mathcal{H}}_i \boldsymbol{\Pi} \boldsymbol{\mathcal{H}}_i\} = \mathbb{E}_{\boldsymbol{\mathcal{H}}}(\boldsymbol{\Pi})$	$\beta_1 \left(\boldsymbol{I}_N \otimes \boldsymbol{1}_{LL}\right) \odot \boldsymbol{\Pi} + \beta_2 \boldsymbol{I}_{NL} \odot \boldsymbol{\Pi} + \beta_3 \boldsymbol{\Pi}$
$\mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\Pi}]_{k\ell}\boldsymbol{H}_{k,i}\} = [\boldsymbol{\varphi}_H(\boldsymbol{\Pi})]_{k\ell}$	$[\beta_2 \left(\boldsymbol{1}_{NN} \otimes \boldsymbol{I}_{LL}\right) \odot \boldsymbol{\Pi} + (\beta_1 + \beta_3)\boldsymbol{\Pi}]_{k\ell}$

Algorithm 3.1 Local updates at node k for CD

for $i = 1, \ldots$ **do**
  randomly generate $\boldsymbol{H}_{k,i}$
  for $\ell \in \mathcal{N}_k \setminus \{k\}$ **do**
    send $\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i}$ to node ℓ
    receive from node ℓ the gradient vector:

$$\hat{\nabla}_w J_\ell(\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}_{\ell,i-1})$$

  end for
  update the intermediate estimate:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \hat{\nabla}_w J_\ell(\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}_{\ell,i-1})$$

  calculate the local estimate:

$$\boldsymbol{w}_{k,i} = a_{kk}\boldsymbol{\psi}_{k,i} + \sum_{\ell \in \mathcal{N}_k \setminus \{k\}} a_{\ell k} \left[\boldsymbol{H}_{\ell,i}\, \boldsymbol{w}_{\ell,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{\ell,i})\, \boldsymbol{\psi}_{k,i}\right]$$

end for

doubly stochastic. The matrix $\boldsymbol{A}$ is equal to the identity matrix $\boldsymbol{I}_N$, with the purpose of limiting the network load. We define recursion for the *CD* as:

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell,k}\boldsymbol{u}_{\ell,i}[d_\ell(i) - \boldsymbol{u}_{\ell,i}^\top(\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}_{\ell,i-1})] \tag{3.35}$$

where $\boldsymbol{H}_{\ell,i} = \text{diag}\{\boldsymbol{h}_{\ell,i}\}$ and $\boldsymbol{h}_{\ell,i}$ is a random vector defined in (3.9). Note that node ℓ receives exactly M entries from its neighbours and uses its own estimate to complete the $(L - M)$ missing entries.

We briefly recall the notation from diffusion LMS chapter 2

$$\widetilde{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i} \tag{3.36}$$

$$\widetilde{\boldsymbol{w}}_i = \mathrm{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\} \tag{3.37}$$

$$\boldsymbol{\mathcal{M}} = \mathrm{diag}\{\mu_1 \boldsymbol{I}_L, \mu_2 \boldsymbol{I}_L, \ldots, \mu_N \boldsymbol{I}_L\} \tag{3.38}$$

$$\boldsymbol{\mathcal{R}}_i = \mathrm{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell,1}\boldsymbol{R}_{u_\ell,i}, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell,N}\boldsymbol{R}_{u_\ell,i}\right\} \tag{3.39}$$

$$\boldsymbol{\mathcal{H}}_i = \mathrm{diag}\{\boldsymbol{H}_{1,i}, \boldsymbol{H}_{2,i}, \ldots, \boldsymbol{H}_{N,i}\} \tag{3.40}$$

$$\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_L \tag{3.41}$$

$$\boldsymbol{\mathcal{R}}_{u,i} = \mathrm{diag}\{\boldsymbol{R}_{u_1,i}, \boldsymbol{R}_{u_2,i}, \ldots, \boldsymbol{R}_{u_N,i}\} \tag{3.42}$$

$$\boldsymbol{R}_{u_\ell,i} = \boldsymbol{u}_{\ell,i}\boldsymbol{u}_{\ell,i}^\top \tag{3.43}$$

where we also introduce the matrix $[\boldsymbol{\mathcal{R}}_{m,i}]_{k\ell}$ defined as

$$[\boldsymbol{\mathcal{R}}_{m,i}]_{k\ell} = c_{\ell k}\boldsymbol{R}_{u_\ell,i}(\boldsymbol{I}_{NL} - \boldsymbol{H}_{k,i}) \tag{3.44}$$

Using the recursion (3.35) and the definitions (3.36), (3.37) we get:

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{B}}_{CD_i}\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{G}}\boldsymbol{s}_i \tag{3.45}$$

where

$$\boldsymbol{\mathcal{B}}_{CD_i} = \boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i\boldsymbol{\mathcal{H}}_i - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{m,i} \tag{3.46}$$

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top \tag{3.47}$$

$$\boldsymbol{s}_i = \mathrm{col}\{\boldsymbol{u}_{1,i}v_1(i), \boldsymbol{u}_{2,i}v_2(i), \ldots, \boldsymbol{u}_{N,i}v_N(i)\} \tag{3.48}$$

Mean weight behaviour analysis

Before delving into the theoretical analysis, let us introduce the following assumptions:

Assumption 3.1. *The regression vectors $\boldsymbol{u}_{k,i}$ arise from a zero-mean random process that is temporally white and spatially independent.*

Assumption 3.2. *The matrices $\boldsymbol{H}_{i,k}$ and $\boldsymbol{Q}_{\ell,i}$ arise from a random process that is temporally white, spatially independent, and independent of each other as well as any other process.*

Taking expectation of both sides of recursion (3.45), using Assumptions 3.1 and 3.2, and $\mathbb{E}\{\boldsymbol{s}_i\} = 0$, we find that:

$$\begin{aligned}\mathbb{E}\{\widetilde{\boldsymbol{w}}_i\} &= \left(\boldsymbol{I}_{NL} - \mathbb{E}\{\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i\boldsymbol{\mathcal{H}}_i\} - \mathbb{E}\{\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{m,i}\}\right)\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}\} - \mathbb{E}\{\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{s}_i\} \\ &= \left(\boldsymbol{I}_{NL} - \tfrac{M}{L}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \left(1 - \tfrac{M}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{R}}_u\right)\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}\}\end{aligned} \tag{3.49}$$

where

$$\boldsymbol{\mathcal{H}} = \tfrac{M}{L}\boldsymbol{I}_{NL} \tag{3.50}$$

$$\boldsymbol{\mathcal{R}} = \mathbb{E}\{\boldsymbol{\mathcal{R}}_i\} = \mathrm{diag}\{\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N\} \tag{3.51}$$

$$\boldsymbol{\mathcal{R}}_u = \mathbb{E}\{\boldsymbol{\mathcal{R}}_{u,i}\} = \mathrm{diag}\{\boldsymbol{R}_{u_1}, \boldsymbol{R}_{u_2}, \ldots, \boldsymbol{R}_{u_N}\} \tag{3.52}$$

and

$$\boldsymbol{R}_k = \sum_{\ell \in \mathcal{N}_k} c_{\ell,k} \boldsymbol{R}_{u_\ell} \tag{3.53}$$

with $\boldsymbol{\mathcal{R}}_{u_\ell}$ being the covariance matrix of the regression vector $\boldsymbol{u}_{\ell,i}$.

From (3.49), the algorithm asymptotically converges in the mean to $\boldsymbol{w}^o$ if and only if matrix $\left(\boldsymbol{I}_{NL} - \frac{M}{L}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \left(1 - \frac{M}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{R}}_u\right)$ is stable, meaning that all its eigenvalues lie strictly inside the unit disc. From [Sayed, 2013a] we have:

$$\begin{aligned}\rho\left(\boldsymbol{I}_{NL} - \tfrac{M}{L}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \left(1 - \tfrac{M}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{R}}_u\right) &\leq ||\boldsymbol{I}_{NL} - \tfrac{M}{L}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \left(1 - \tfrac{M}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{R}}_u||_{b,\infty} \\ &\leq N \max_{\ell,k} ||\boldsymbol{I}_{NL} - \tfrac{M}{L}\mu_k\boldsymbol{R}_k - \left(1 - \tfrac{M}{L}\right)\mu_k c_{\ell k}\boldsymbol{R}_{u_\ell}||\end{aligned} \tag{3.54}$$

since $\boldsymbol{R}_k$ and $\boldsymbol{R}_{u_\ell}$ are Hermitian matrices we have the following condition on the step-size parameters μ_k:

$$\mu_k < \frac{2}{N \max_\ell [\frac{M}{L}\lambda_{\max}(\boldsymbol{R}_k) + \left(1 - \frac{M}{L}\right) c_{\ell k}\lambda_{\max}(\boldsymbol{R}_{u_\ell})]} \tag{3.55}$$

where $\lambda_{\max}(\cdot)$ stands for the maximum eigenvalue of its matrix argument [Sayed et al., 2013].

Mean-square error behaviour analysis

With the aim of generality, we evaluate the weighted mean-square deviation $\mathbb{E}\{||\boldsymbol{w}||^2_{\boldsymbol{\Sigma}}\}$ where $\boldsymbol{\Sigma}$ denotes a non-negative definite block diagonal matrix of $L \times L$ block entries. The choice of the matrix $\boldsymbol{\Sigma}$ will determine the type of extracted information related to the network and nodes. Using the independence 3.1 and (3.45) we find:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|^2_{\boldsymbol{\Sigma}} = \mathbb{E}\{\widetilde{\boldsymbol{w}}^\top_{i-1}\boldsymbol{\mathcal{B}}^\top_{CD_i}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{CD_i}\widetilde{\boldsymbol{w}}_{i-1}\} + \mathbb{E}\{\boldsymbol{s}_i^\top\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}\boldsymbol{s}_i\} \tag{3.56}$$

The right-hand side term has already been calculated in [Sayed, 2013a] and exhibited in chapter 2. We shall simply use the result:

$$\mathbb{E}\{\boldsymbol{s}_i^\top\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}\boldsymbol{s}_i\} = \mathrm{trace}\big(\boldsymbol{\mathcal{G}}\,\boldsymbol{\mathcal{S}}\,\boldsymbol{\mathcal{G}}^\top\boldsymbol{\Sigma}\big) \tag{3.57}$$

with

$$\boldsymbol{\mathcal{S}} = \mathrm{diag}(\sigma^2_{v,1}\boldsymbol{R}_{u,1}, \ldots, \sigma^2_{v,N}\boldsymbol{R}_{u,N}) \tag{3.58}$$

In contrast with the previous term, the left-hand side term has not been calculated in the literature and needs to be evaluated.

$$\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}^{\top}\boldsymbol{\mathcal{B}}_{CD,i}^{\top}\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{CD,i}\widetilde{\boldsymbol{w}}_{i-1}\} = \mathbb{E}\|\widetilde{\boldsymbol{w}}_{i-1}\|_{\boldsymbol{\Sigma}'_{CD}}^{2} \tag{3.59}$$

where the weighting matrix $\boldsymbol{\Sigma}'_{CD_i}$ is defined as

$$\begin{aligned}\boldsymbol{\Sigma}'_{CD} &= \mathbb{E}\{\boldsymbol{\mathcal{B}}_{CD_i}^{\top}\boldsymbol{\Sigma}\,\boldsymbol{\mathcal{B}}_{CD_i}\}\\ &= \boldsymbol{\Sigma} - \tfrac{M}{L}\boldsymbol{\Sigma}\,\boldsymbol{\mathcal{M}}\,\boldsymbol{\mathcal{R}} - \left(1-\tfrac{M}{L}\right)\boldsymbol{\Sigma}\,\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^{\top}\boldsymbol{\mathcal{R}}_u - \tfrac{M}{L}\boldsymbol{\mathcal{R}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma} - \left(1-\tfrac{M}{L}\right)\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\\ &+ \boldsymbol{P}_1 + \boldsymbol{P}_2 + \boldsymbol{P}_3 + \boldsymbol{P}_4\end{aligned} \tag{3.60}$$

with

$$\begin{aligned}\boldsymbol{P}_1 &= \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}_i^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i\boldsymbol{\mathcal{H}}_i\}\\ &= \mathbb{E}_{\boldsymbol{\mathcal{H}}}(\boldsymbol{\mathcal{R}}_i^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i)\end{aligned} \tag{3.61}$$

where $\mathbb{E}_{\boldsymbol{\mathcal{H}}}$ is defined in Table 3.2. Following [Sayed, 2013a], we focus on the case of sufficiently small step sizes $\{\mu_k\}$ where the effect of terms involving higher powers of the step-sizes can be ignored. A reasonable approximation for sufficiently small step sizes is:

$$\mathbb{E}\{\boldsymbol{\mathcal{R}}_i^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i\} = \mathbb{E}\{\boldsymbol{\mathcal{R}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}\}$$

We finally get:

$$\boldsymbol{P}_1 = \mathbb{E}_{\boldsymbol{\mathcal{H}}}(\boldsymbol{\mathcal{R}}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}) \tag{3.62}$$

Let us now evaluate the term $\boldsymbol{P}_2$, using the results (3.11) – (3.19) and the same approximation as in (3.62) we get:

$$\begin{aligned}\boldsymbol{P}_2 &= \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}_i^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{m,i}\}\\ [\boldsymbol{P}_2]_{k\ell} &= \tfrac{M}{L}\boldsymbol{R}_k[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}c_{\ell k}\boldsymbol{R}_{u_\ell} - \mathbb{E}\{\boldsymbol{H}_{k,i}\boldsymbol{R}_{k,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}c_{\ell k}\boldsymbol{R}_{u_\ell,i}\boldsymbol{H}_{k,i}\}\\ \boldsymbol{P}_2 &= \tfrac{M}{L}\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^{\top}\boldsymbol{\mathcal{R}}_u - \boldsymbol{\varphi}_H(\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^{\top}\boldsymbol{\mathcal{R}}_u)\end{aligned} \tag{3.63}$$

where $\boldsymbol{\varphi}_H$ is defined in Table 3.2. Note that the term $\boldsymbol{P}_3$ is the transposed $\boldsymbol{P}_2$, hence

$$\boldsymbol{P}_3 = \tfrac{M}{L}\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \boldsymbol{\varphi}_H(\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}) \tag{3.64}$$

We finally express the last term, using the same steps and approximation as the previous terms we find:

$$\begin{aligned}\boldsymbol{P}_4 &= \mathbb{E}\{\boldsymbol{\mathcal{R}}_{m,i}^{\top}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{m,i}\}\\ [\boldsymbol{P}_4]_{k\ell} &= \sum_{m=1}^{N}\mathbb{E}\{[\boldsymbol{\mathcal{R}}_{m,i}]_{km}^{\top}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{mm}[\boldsymbol{\mathcal{R}}_{m,i}]_{m\ell}\}\end{aligned}$$

$$\boldsymbol{P}_4 = \left(1 - 2\tfrac{M}{L}\right) \boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{M}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}^\top \boldsymbol{\mathcal{R}}_u \boldsymbol{\varphi}_h(\boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{M}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}^\top \boldsymbol{\mathcal{R}}_u) \tag{3.65}$$

The matrix $\boldsymbol{\Sigma}'_{CD}$ defined in (3.60) can be expressed in a vector form as:

$$\boldsymbol{\sigma}'_{CD} = \boldsymbol{\mathcal{F}}_{CD} \boldsymbol{\sigma} \tag{3.66}$$

where $\boldsymbol{\sigma} = \text{vec}(\boldsymbol{\Sigma})$ and $\boldsymbol{\mathcal{F}}_{CD}$ defined as:

$$\begin{aligned} \boldsymbol{\mathcal{F}}_{CD} =& \boldsymbol{I}_{(NM)^2} - \tfrac{M}{L}(\boldsymbol{\mathcal{R}} \boldsymbol{\mathcal{M}} \otimes \boldsymbol{I}_{NL}) - (1 - \tfrac{M}{L})(\boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{M}} \otimes \boldsymbol{I}_{NL}) - \tfrac{M}{L}(\boldsymbol{I}_{NL} \otimes \boldsymbol{\mathcal{R}} \boldsymbol{\mathcal{M}}) \\ & -(1 - \tfrac{M}{L})(\boldsymbol{I}_{NL} \otimes \boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{M}}) + \boldsymbol{Z}_1 + \boldsymbol{Z}_2 - \boldsymbol{Z}_3 + \boldsymbol{Z}_4 \end{aligned} \tag{3.67}$$

where $\boldsymbol{Z}_j$ are obtained by applying the vec(.) operator on $\boldsymbol{P}_j$ and using the following property:

$$\text{vec}(\boldsymbol{ABC}) = \text{vec}(\boldsymbol{C}^\top \otimes \boldsymbol{A}) \, \text{vec}(\boldsymbol{B}) \tag{3.68}$$

Replacing (3.59) and (3.57) in (3.56), and applying vec(.) operator on both sides we get:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}_{i-1}\|^2_{\boldsymbol{\mathcal{F}}_{CD}\boldsymbol{\sigma}} + [\text{vec}(\boldsymbol{\mathcal{Y}}^\top)]^\top \boldsymbol{\sigma} \tag{3.69}$$

where

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{G}} \, \boldsymbol{\mathcal{S}} \, \boldsymbol{\mathcal{G}}^\top \tag{3.70}$$

Using (3.69), and depending on the weighting matrix $\boldsymbol{\Sigma}$, various statistical properties of the network can be extracted such as MSD or EMSE as it shall be demonstrated in the upcoming sections.

Network Mean-Square Performance

We use the equation (3.69) to study the network excess mean square error (EMSE). As the algorithm is stable for sufficiently small step-sizes (3.55) we can take the limits as

$$\lim_{i\to\infty} \mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\sigma}} = \lim_{i\to\infty} \mathbb{E}||\widetilde{\boldsymbol{w}}_{i-1}||^2_{\boldsymbol{\mathcal{F}}_{CD}\boldsymbol{\sigma}} + \text{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \boldsymbol{\sigma} \tag{3.71}$$

Rearranging the terms results in

$$\lim_{i\to\infty} \mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{(\boldsymbol{I}-\boldsymbol{\mathcal{F}}_{CD})\boldsymbol{\sigma}} = \text{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \boldsymbol{\sigma} \tag{3.72}$$

We have

$$\text{EMSD}^{Network} = \tfrac{1}{N}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\mathcal{R}}_u} \tag{3.73}$$

which leads to

$$\boldsymbol{\sigma} = \tfrac{1}{N}(\boldsymbol{I} - \boldsymbol{\mathcal{F}}_{CD})^{-1} \text{vec}(\boldsymbol{\mathcal{R}}_u) \tag{3.74}$$

Finally, we find

$$\text{MSD}^{Network} = \tfrac{1}{N}\text{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top (\boldsymbol{I} - \boldsymbol{\mathcal{F}}_{CD})^{-1} \text{vec}(\boldsymbol{\mathcal{R}}_u) \tag{3.75}$$

Transient State Analysis

We study the algorithm transient behaviour by iterating the equation (3.69) from $i = 0$. We find

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\sigma}} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{-1}||^2_{\boldsymbol{\mathcal{F}}^{i+1}_{CD}\boldsymbol{\sigma}} + \mathrm{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \sum_{j=0}^{i} \boldsymbol{\mathcal{F}}^j_{CD}\boldsymbol{\sigma} \tag{3.76}$$

where $\widetilde{\boldsymbol{w}}_{-1}$ is the initial condition term. Comparing the recursion (3.76) for the time instants i and $i-1$ we find

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\sigma}} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{i-1}||^2_{\boldsymbol{\sigma}} + \mathrm{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \boldsymbol{\mathcal{F}}^i_{CD}\boldsymbol{\sigma} - \mathbb{E}||\widetilde{\boldsymbol{w}}_{-1}||^2_{(\boldsymbol{I}-\boldsymbol{\mathcal{F}}_{CD})\boldsymbol{\mathcal{F}}^i_{CD}\boldsymbol{\sigma}} \tag{3.77}$$

In order to evaluate the transient excess mean square error (EMSE), the weighing vector $\boldsymbol{\sigma}$ needs to be set to $\frac{1}{N}\mathrm{vec}(\boldsymbol{\mathcal{R}}_u)$ where $\boldsymbol{\mathcal{R}}_u$ is defined in (3.52).

Note that similar performance information can be extracted for a single agent by setting the weighting matrix $\boldsymbol{\Sigma}$ to zeros except for the block matrix corresponding to the said agent k that shall be set to the $\boldsymbol{R}_{u_k}$.

3.2.3 Doubly Compressed Diffusion LMS

Let un now introduce the fully compressed version of diffusion LMS: *Doubly Compressed Diffusion LMS*. The DCD algorithm run at each node k is shown in 3.2. Matrices $\boldsymbol{H}_{k,i}$ and $\boldsymbol{Q}_{k,i}$ are diagonal entry-selection matrices with M and M_∇ ones on their diagonal, respectively. The other diagonal entries of these two matrices are set to zero. First, we consider the adaptation step. The matrix $\boldsymbol{H}_{k,i}$ selects M entries (over L) of $\boldsymbol{w}_{k,i-1}$ that are sent to node ℓ to approximate $\nabla_w J_\ell(\boldsymbol{w}_{k,i-1})$ in (2.4). Node ℓ fills the missing entries of $\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1}$ by using its own entries $(\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}_{\ell,i-1}$, and calculates the instantaneous approximation of the gradient vector at this point. Then node ℓ selects M_∇ entries (over L) of this gradient vector using $\boldsymbol{Q}_{k,i}$ and send them to node k. Node k fills the missing entries by using its own local estimate. Finally, node k considers the partial parameter vectors $\boldsymbol{H}_{\ell,i}\boldsymbol{w}_{\ell,i-1}$ received from its neighbours ℓ during the adaptation step, and fills the missing entries by using its own local estimate $\boldsymbol{\psi}_{k,i}$. Then it aggregates them to obtain the local estimate $\boldsymbol{w}_{k,i}$.

We can formulate the algorithm under the following form:

$$\begin{cases} \boldsymbol{\psi}_{k,i} &= \boldsymbol{w}_{k,i-1} + \mu_k \sum\limits_{\ell\in\mathcal{N}_k} c_{\ell k}\boldsymbol{g}_{\ell,i} \\ \boldsymbol{w}_{k,i} &= a_{kk}\boldsymbol{\psi}_{k,i} + \sum\limits_{\ell\in\mathcal{N}_k\setminus\{k\}} a_{\ell k}\left[\boldsymbol{H}_{\ell,i}\,\boldsymbol{w}_{\ell,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{\ell,i})\,\boldsymbol{\psi}_{k,i}\right] \end{cases} \tag{3.78}$$

Algorithm 3.2 Local updates at node k for DCD

for $i = 1, \ldots$ **do**
  randomly generate $\boldsymbol{H}_{k,i}$ and $\boldsymbol{Q}_{k,i}$
  for $\ell \in \mathcal{N}_k \setminus \{k\}$ **do**
    send $\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i}$ to node ℓ
    receive from node ℓ the partial gradient vector:

$$\boldsymbol{Q}_{\ell,i}\hat{\nabla}_w J_\ell(\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}_{\ell,i-1})$$

    complete the missing entries using those available at node k, which results in $\boldsymbol{g}_{\ell,i}$ defined in (3.79)
  end for
  update the intermediate estimate:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}\boldsymbol{g}_{\ell,i}$$

  calculate the local estimate:

$$\boldsymbol{w}_{k,i} = a_{kk}\boldsymbol{\psi}_{k,i} + \sum_{\ell \in \mathcal{N}_k \setminus \{k\}} a_{\ell k}\left[\boldsymbol{H}_{\ell,i}\,\boldsymbol{w}_{\ell,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{\ell,i})\,\boldsymbol{\psi}_{k,i}\right]$$

end for

with

$$\begin{aligned}\boldsymbol{g}_{\ell,i} = \boldsymbol{Q}_{\ell,i}\boldsymbol{u}_{\ell,i}\big[d_\ell(i) - \boldsymbol{u}_{\ell,i}^\top(\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\,\boldsymbol{w}_{\ell,i-1})\big] \\ +(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i})\boldsymbol{u}_{k,i}\big[d_k(i) - \boldsymbol{u}_{k,i}^\top\boldsymbol{w}_{k,i-1})\big]\end{aligned} \tag{3.79}$$

where $\boldsymbol{H}_{\ell,i} = \text{diag}\{\boldsymbol{h}_{\ell,i}\}$, $\boldsymbol{Q}_{\ell,i} = \text{diag}\{\boldsymbol{q}_{\ell,i}\}$, $\boldsymbol{h}_{\ell,i}$ and $\boldsymbol{q}_{\ell,i}$ are binary vectors randomly drawn following the definition (3.9) in subsection 3.2.1.

We shall now analyse the stochastic behaviour of the DCD algorithm. For the sake of simplicity, we shall consider that matrix $\boldsymbol{C}$ is doubly stochastic. We shall also set matrix $\boldsymbol{A}$ to the identity matrix. Focusing this way on the adaptation step and gradient vector sharing, helps simplifying the analysis. Note that the distributed LMS with partial diffusion (3.3), which exclusively addresses how reducing the communication load induced by the combination step, was analysed in [Arablouei et al., 2014b]. Combining both analyses into a single general one is challenging and beyond the scope of this study. However, in the sequel, we shall illustrate the efficiency of the DCD algorithm in both cases $\boldsymbol{A} = \boldsymbol{I}_L$ and $\boldsymbol{A} \neq \boldsymbol{I}_L$, and compare it with the existing strategies.

Before proceeding with the algorithm analysis, let us introduce the following assumptions on the regression data and selection matrices.

Assumption 3.3. *The matrices $\boldsymbol{H}_{i,k}$ and $\boldsymbol{Q}_{\ell,i}$ arise from a random process that is temporally white, spatially independent, and independent of each other as well as any other process.*

Following the same steps as CD algorithm we re-introduce the $L \times 1$ concatenated error vector:

$$\widetilde{\boldsymbol{w}}_i = \text{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\} \tag{3.80}$$

where

$$\widetilde{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i} \tag{3.81}$$

We also introduce:

$$\boldsymbol{\mathcal{R}}_{Q,i} = \text{diag}\Big\{ \sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i}, \sum_{\ell\in\mathcal{N}_2} c_{\ell 2}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i}, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i} \Big\} \tag{3.82}$$

$$\boldsymbol{\mathcal{Q}}'_i = \text{diag}\Big\{ \sum_{\ell\in\mathcal{N}_1} c_{\ell 1}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i}), \sum_{\ell\in\mathcal{N}_2} c_{\ell 1}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i}) \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i}) \Big\} \tag{3.83}$$

$$\boldsymbol{\mathcal{Q}}_i = \text{diag}\{\boldsymbol{Q}_{1,i}, \boldsymbol{Q}_{2,i}, \ldots, \boldsymbol{Q}_{N,i}\} \tag{3.84}$$

Finally, we introduce the $N \times N$ block matrix $\boldsymbol{\mathcal{R}}_{Q(I-H),i}$ with each block (k,ℓ) defined as:

$$[\boldsymbol{\mathcal{R}}_{Q(I-H),i}]_{k\ell} = c_{\ell k}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i}(\boldsymbol{I}_L - \boldsymbol{H}_{k,i}) \tag{3.85}$$

Combining recursion (3.78) and definition (3.81), and replacing $d_k(i)$ by its definition (3.6), we find:

$$\begin{aligned}\widetilde{\boldsymbol{w}}_{k,i} = \widetilde{\boldsymbol{w}}_{k,i-1} - \mu_k \sum_{\ell\in\mathcal{N}_k} c_{\ell k}\boldsymbol{Q}_{\ell,i}\boldsymbol{u}_{\ell,i}\big[\boldsymbol{u}_{\ell,i}^\top \boldsymbol{w}^o + v_\ell(i) \\ -\boldsymbol{u}_{\ell,i}^\top(\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}_{\ell,i-1})\big] \\ -\mu_k \sum_{\ell\in\mathcal{N}_k} c_{\ell k}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i})\boldsymbol{u}_{k,i}\big[\boldsymbol{u}_{k,i}^\top \boldsymbol{w}^o + v_k(i) - \boldsymbol{u}_{k,i}^\top \boldsymbol{w}_{k,i-1}\big]\end{aligned} \tag{3.86}$$

Note that $\boldsymbol{w}^o = \boldsymbol{H}_{k,i}\boldsymbol{w}^o + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}^o$. Replacing $\boldsymbol{w}^o$ by its new expression, and using definition (3.80), leads to:

$$\begin{aligned}\widetilde{\boldsymbol{w}}_{k,i} = \widetilde{\boldsymbol{w}}_{k,i-1} - \mu_k \sum_{\ell\in\mathcal{N}_k} c_{\ell k}\boldsymbol{Q}_{\ell,i}\boldsymbol{u}_{\ell,i}\big[\boldsymbol{u}_{\ell,i}^\top \boldsymbol{H}_{k,i}\widetilde{\boldsymbol{w}}_{k,i-1} + \boldsymbol{u}_{\ell,i}^\top(\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\widetilde{\boldsymbol{w}}_{\ell,i-1} + v_\ell(i)\big] \\ - \mu_k \sum_{\ell\in\mathcal{N}_k} c_{\ell k}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i})\boldsymbol{u}_{k,i}\big[\boldsymbol{u}_{k,i}^\top \widetilde{\boldsymbol{w}}_{k,i-1} + v_k(i)\big]\end{aligned} \tag{3.87}$$

Rearranging the terms in (3.87), and using definitions (3.80)–(3.84), leads to:

$$\begin{aligned}\widetilde{\boldsymbol{w}}_i = \big(\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q,i}\boldsymbol{\mathcal{H}}_i - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}'_i\boldsymbol{\mathcal{R}}_{u,i} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q(I-H),i}\big)\widetilde{\boldsymbol{w}}_{i-1} \\ - \big(\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{Q}}_i + \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}'_i\big)\boldsymbol{s}_i\end{aligned} \tag{3.88}$$

where $\boldsymbol{s}_i$ is defined in (3.48) as

$$\boldsymbol{s}_i = \text{col}\{\boldsymbol{u}_{1,i}v_1(i), \boldsymbol{u}_{2,i}v_2(i), \ldots, \boldsymbol{u}_{N,i}v_N(i)\}$$

Mean weight behaviour analysis

We shall now examine the convergence in the mean for the DCD algorithm and derive a necessary convergence condition. We start by rewriting the weight-error vector recursion (3.88) as:

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{B}}_{DCD_i}\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{G}}_{DCD_i}\boldsymbol{s}_i \tag{3.89}$$

where the coefficient matrices $\boldsymbol{\mathcal{B}}_i$ and $\boldsymbol{\mathcal{G}}_i$ are this time defined as:

$$\boldsymbol{\mathcal{B}}_{DCD_i} = \boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q,i}\boldsymbol{\mathcal{H}}_i - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}_i'\boldsymbol{\mathcal{R}}_{u,i} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q(I-H),i} \tag{3.90}$$

$$\boldsymbol{\mathcal{G}}_{DCD_i} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{Q}}_i + \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}_i' \tag{3.91}$$

Taking expectations of both sides of (3.89), using the Assumptions 3.1 and 3.3, and $\mathbb{E}\{\boldsymbol{s}_i\} = 0$, we find:

$$\mathbb{E}\{\widetilde{\boldsymbol{w}}_i\} = \boldsymbol{\mathcal{B}}_{DCD}\,\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}\}$$

where

$$\boldsymbol{\mathcal{B}}_{DCD} = \boldsymbol{I}_{NL} - \tfrac{MM_\nabla}{L^2}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \left(1 - \tfrac{M_\nabla}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_u - \tfrac{M_\nabla}{L}\left(1 - \tfrac{M}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{R}}_u \tag{3.92}$$

and $\boldsymbol{\mathcal{R}}$, $\boldsymbol{\mathcal{R}}_u$ and $\boldsymbol{R}_k$ are defined in (3.51) – (3.53), respectively as:

$$\begin{aligned}
\boldsymbol{\mathcal{R}} &= \operatorname{diag}\{\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N\} \\
\boldsymbol{\mathcal{R}}_u &= \mathbb{E}\{\boldsymbol{\mathcal{R}}_{u,i}\} = \operatorname{diag}\{\boldsymbol{R}_{u_1}, \boldsymbol{R}_{u_2}, \ldots, \boldsymbol{R}_{u_N}\} \\
\boldsymbol{R}_k &= \sum_{\ell \in \mathcal{N}_k} c_{\ell,k}\boldsymbol{R}_{u_\ell}
\end{aligned}$$

From (3.92), we observe that the algorithm (3.78) asymptotically converges in the mean toward $\boldsymbol{w}^o$ if, and only if,

$$\rho(\boldsymbol{\mathcal{B}}_{DCD}) < 1 \tag{3.93}$$

where $\rho(\cdot)$ denotes the spectral radius of its matrix argument. We know that $\rho(\boldsymbol{X}) \leq \|\boldsymbol{X}\|$ for any induced norm. Then:

$$\begin{aligned}
\rho(\boldsymbol{\mathcal{B}}) &\leq \|\boldsymbol{\mathcal{B}}_{DCD}\|_{b,\infty} \\
&\leq N \max_{k,\ell} \|[\boldsymbol{\mathcal{B}}_{DCD}]_{k\ell}\|
\end{aligned} \tag{3.94}$$

where $\|\cdot\|_{b,\infty}$ denotes the block maximum norm. From (3.94) we have:

$$\rho(\boldsymbol{\mathcal{B}}_{DCD}) \leq N \max_{\ell,k} \left\| \boldsymbol{I}_L - \mu_k \left[\tfrac{MM_\nabla}{L^2}\boldsymbol{R}_k + \tfrac{M}{L}\left(1 - \tfrac{M_\nabla}{L}\right)\boldsymbol{R}_{u_k} + \tfrac{M_\nabla}{L}\left(1 - \tfrac{M}{L}\right)c_{\ell k}\boldsymbol{R}_{u_\ell} \right] \right\| \tag{3.95}$$

As a linear combination with positive coefficients of positive definite matrices $\boldsymbol{R}_k$ and $\boldsymbol{R}_{u_\ell}$, the matrix in square brackets on the RHS of (3.95) is positive definite. Condition (3.93) then holds if

for all k:

$$\mu_k < \frac{2}{N\lambda_{\max,k}} \tag{3.96}$$

with

$$\begin{aligned}\lambda_{\max,k} = \tfrac{MM_\nabla}{L^2}\lambda_{\max}(\boldsymbol{R}_k) + \tfrac{M}{L}\left(1-\tfrac{M_\nabla}{L}\right)\lambda_{\max}(\boldsymbol{R}_{u_k}) \\ + \tfrac{M_\nabla}{L}\left(1-\tfrac{M}{L}\right)\max_{\ell\in\boldsymbol{\mathcal{N}}_k} c_{\ell k}\,\lambda_{\max}(\boldsymbol{R}_{u_\ell})\end{aligned} \tag{3.97}$$

where we used Jensen's inequality with $\lambda_{\max}(\cdot)$ operator, that stands for the maximum eigenvalue of its matrix argument. It is worth mentioning that when $M = M_\nabla = L$ we retrieve the convergence condition of the diffusion LMS as derived in [Sayed, 2013a]:

$$\lambda_{\max,k} = \lambda_{\max}(\boldsymbol{R}_k) \tag{3.98}$$

With this setting, we also retrieve the matrices $\boldsymbol{\mathcal{B}}$ (2.32) and $\boldsymbol{\mathcal{G}}$ (2.29) of the diffusion LMS defined in chapter 2.

Mean-square error behaviour analysis

As for the CD algorithm, we are now interested in providing a global solution for studying the mean square error. With this aim, we consider the weighted square measure $\mathbb{E}\{\|\widetilde{\boldsymbol{w}}_i\|^2_{\boldsymbol{\Sigma}}\}$ where $\boldsymbol{\Sigma}$ denotes a $N\times N$ block diagonal weighting matrix. By setting $\boldsymbol{\Sigma}$ to different values, we can extract various types of information about the nodes and the network such as the network mean square deviation MSD, or the excess mean square error EMSE.

We start by using the independence 2.2 and (3.89) to write

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|^2_{\boldsymbol{\Sigma}} = \mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}^\top \boldsymbol{\mathcal{B}}_{DCD,i}^\top \boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{DCD,i}\widetilde{\boldsymbol{w}}_{i-1}\} + \mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{G}}_{DCD,i}^\top \boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}_{DCD,i}\boldsymbol{s}_i\} \tag{3.99}$$

On the one hand, the second term on the RHS of (3.99) can be written as:

$$\begin{aligned}\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{G}}_{DCD,i}^\top \boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}_{DCD,i}\boldsymbol{s}_i\} &= \mathrm{trace}\big(\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{G}}_{DCD,i}^\top \boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}_{DCD,i}\boldsymbol{s}_i\}\big) \\ &= \mathrm{trace}\big(\mathbb{E}\{\boldsymbol{\mathcal{G}}_{DCD,i}^\top \boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}_{DCD,i}\}\,\mathbb{E}\{\boldsymbol{s}_i\boldsymbol{s}_i^\top\}\big) \\ &= \mathrm{trace}\big(\mathbb{E}\{\boldsymbol{\mathcal{G}}_{DCD,i}^\top \boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}_{DCD,i}\}\,\boldsymbol{\mathcal{S}}\big)\end{aligned} \tag{3.100}$$

where $\boldsymbol{\mathcal{S}}$ is defined in (3.58) as

$$\boldsymbol{\mathcal{S}} = \mathbb{E}\{\boldsymbol{s}_i\boldsymbol{s}_i^\top\} = \mathrm{diag}(\sigma^2_{v,1}\boldsymbol{R}_{u,1},\ldots,\sigma^2_{v,N}\boldsymbol{R}_{u,N}) \tag{3.101}$$

and

$$\begin{aligned}\mathbb{E}\{\boldsymbol{\mathcal{G}}_{DCD,i}^\top \boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}_{DCD,i}\} &= \mathbb{E}\{\big(\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{Q}}_i + \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}'_i\big)^\top \boldsymbol{\Sigma}\big(\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{Q}}_i + \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}'_i\big)\} \\ &= \boldsymbol{\Theta}_1 + \boldsymbol{\Theta}_2 + \boldsymbol{\Theta}_2^\top + \boldsymbol{\Theta}_3\end{aligned} \tag{3.102}$$

with

$$\boldsymbol{\Theta}_1 = \mathbb{E}\{\boldsymbol{\mathcal{Q}}_i \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top \boldsymbol{\mathcal{Q}}_i\} \tag{3.103}$$

$$\boldsymbol{\Theta}_2 = \mathbb{E}\{\boldsymbol{\mathcal{Q}}_i \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}_i'\} \tag{3.104}$$

$$\boldsymbol{\Theta}_3 = \mathbb{E}\{\boldsymbol{\mathcal{Q}}_i' \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}_i'\} \tag{3.105}$$

We can now proceed with the evaluation of (3.103)–(3.105). Using the results from Table 3.2 we find:

$$\boldsymbol{\Theta}_1 = \mathbb{E}\{\boldsymbol{\mathcal{Q}}_i \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top \boldsymbol{\mathcal{Q}}_i\} \tag{3.106}$$

$$= \mathbb{E}_{\boldsymbol{\mathcal{Q}}}(\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top) \tag{3.107}$$

Consider now $\boldsymbol{\Theta}_2$ in (3.102). We have:

$$\begin{aligned}[\boldsymbol{\Theta}_2]_{k\ell} = \tfrac{M_\nabla}{L}[\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{k\ell} - c_{k\ell}^2\,\mathbb{E}\{\boldsymbol{\mathcal{Q}}_{k,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{\ell\ell}\boldsymbol{\mathcal{Q}}_{k,i}\} \\ - \left(\tfrac{M_\nabla}{L}\right)^2 \left([\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{k\ell} - c_{k\ell}^2\,[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{\ell\ell}\right)\end{aligned} \tag{3.108}$$

We can use the operator $\boldsymbol{\varphi}_Q$ from Table 3.2 to calculate the second term in the RHS of the above equation since $\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}$ is block diagonal. This yields:

$$\boldsymbol{\Theta}_2 = \tfrac{M_\nabla}{L}\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{C}}_2\boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}) - \left(\tfrac{M_\nabla}{L}\right)^2(\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{C}}_2\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}) \tag{3.109}$$

where

$$\boldsymbol{\mathcal{C}}_2 = \boldsymbol{\mathcal{C}} \odot \boldsymbol{\mathcal{C}} \tag{3.110}$$

Finally, we calculate the last term $\boldsymbol{\Theta}_3$ in the RHS of (3.102). Matrix $\boldsymbol{\Theta}_3$ is block diagonal, with each diagonal block defined as follows:

$$\begin{aligned}[\boldsymbol{\Theta}_3]_{kk} &= \mathbb{E}\{[\boldsymbol{\mathcal{Q}}_i']_{kk}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}[\boldsymbol{\mathcal{Q}}_i']_{kk}\} \\ &= \sum_{m,n=1}^{N} c_{mk}c_{nk}\,\mathbb{E}\{\left(\boldsymbol{I}_L - \boldsymbol{Q}_{m,i}\right)[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\left(\boldsymbol{I}_L - \boldsymbol{Q}_{n,i}\right)\}\end{aligned}$$

Using (3.11), we get:

$$[\boldsymbol{\Theta}_3]_{kk} = \left(1 - 2\tfrac{M_\nabla}{L}\right)[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\sum_{m,n=1}^{N} c_{mk}c_{nk} \quad + \sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\boldsymbol{Q}_{n,i}\}$$

Applying (3.19) leads to:

$$[\boldsymbol{\Theta}_3]_{kk} = \left(1 - 2\tfrac{M_\nabla}{L}\right)[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk} + \sum_{m}^{N} c_{mk}^2 \mathbb{E}\{\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{Q}_{m,i}\} + \left(\tfrac{M_\nabla}{L}\right)^2 \left(\sum_{m,n=1}^{N} c_{mk}c_{nk}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk} - \sum_{m=1}^{N} c_{mk}^2[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\right) \tag{3.111}$$

Finally, using (3.33), we can write (3.111) in a compact form:

$$\boldsymbol{\Theta}_3 = \left(1 - 2\tfrac{M_\nabla}{L}\right)\boldsymbol{\mathcal{M}\Sigma\mathcal{M}} + (\boldsymbol{I}_{NL} \odot \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{C}}^\top)\,\boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}) + \left(\tfrac{M_\nabla}{L}\right)^2 \left(\boldsymbol{\mathcal{M}\Sigma\mathcal{M}} - (\boldsymbol{I}_{NL} \odot \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{C}}^\top)\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\right) \tag{3.112}$$

It is interesting to notice that the second term in the RHS of (3.99) does not depend on M. Moreover, setting $M_\nabla = L$ results in cancelling $\boldsymbol{\Theta}_2$ and $\boldsymbol{\Theta}_3$ since $\boldsymbol{\mathcal{Q}}_i' = \boldsymbol{0}$.

The first term on the RHS of (3.99) depends on both parameters M and M_∇ and can be expressed as:

$$\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}^\top \boldsymbol{\mathcal{B}}_{DCD_i}^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_{DCD_i} \widetilde{\boldsymbol{w}}_{i-1}\} = \mathbb{E}\|\widetilde{\boldsymbol{w}}_{i-1}\|_{\boldsymbol{\Sigma}'_{DCD}}^2 \tag{3.113}$$

where the weighting matrix $\boldsymbol{\Sigma}'$ is defined as:

$$\boldsymbol{\Sigma}'_{DCD} = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{DCD_i}^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_{DCD_i}\} \tag{3.114}$$

Replacing $\boldsymbol{\mathcal{B}}_i$ by its definition (3.90) leads to:

$$\begin{aligned}\boldsymbol{\Sigma}'_{DCD} = \boldsymbol{\Sigma} &- \tfrac{MM_\nabla}{L^2}\boldsymbol{\Sigma\mathcal{M}\mathcal{R}} - \left(1 - \tfrac{M_\nabla}{L}\right)\boldsymbol{\Sigma\mathcal{M}\mathcal{R}}_u - \tfrac{M_\nabla}{L}\left(1 - \tfrac{M}{L}\right)\boldsymbol{\Sigma\mathcal{M}\mathcal{C}}^\top\boldsymbol{\mathcal{R}}_u \\ &- \tfrac{MM_\nabla}{L^2}\boldsymbol{\mathcal{R}\mathcal{M}\Sigma} - \left(1 - \tfrac{M_\nabla}{L}\right)\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{M}\Sigma} - \tfrac{M_\nabla}{L}\left(1 - \tfrac{M}{L}\right)\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{M}\Sigma} \\ &+ \sum_{j=1}^{6}\boldsymbol{P}_j + \boldsymbol{P}_2^\top + \boldsymbol{P}_3^\top + \boldsymbol{P}_5^\top\end{aligned}$$

where

$$\boldsymbol{P}_1 = \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}_{Q,i}^\top\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q,i}\boldsymbol{\mathcal{H}}_i\} \tag{3.115}$$

$$\boldsymbol{P}_2 = \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}_{Q,i}^\top\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{Q}}_i'\boldsymbol{\mathcal{R}}_{u,i}\} \tag{3.116}$$

$$\boldsymbol{P}_3 = \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}_{Q,i}^\top\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q(I-H),i}\} \tag{3.117}$$

$$\boldsymbol{P}_4 = \mathbb{E}\{\boldsymbol{\mathcal{R}}_{u,i}^\top\boldsymbol{\mathcal{Q}}_i'\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{Q}}_i'\boldsymbol{\mathcal{R}}_{u,i}\} \tag{3.118}$$

$$\boldsymbol{P}_5 = \mathbb{E}\{\boldsymbol{\mathcal{R}}_{u,i}^\top\boldsymbol{\mathcal{Q}}_i'\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q(I-H),i}\} \tag{3.119}$$

$$\boldsymbol{P}_6 = \mathbb{E}\{\boldsymbol{\mathcal{R}}_{Q(I-H),i}^\top\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q(I-H),i}\} \tag{3.120}$$

We shall now evaluate each one of the six terms.

Term $\boldsymbol{P}_1$ calculation Matrix $\boldsymbol{P}_1$ is a block diagonal matrix. Its k-th diagonal block is given by:

$$[\boldsymbol{P}_1]_{kk} = \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\mathcal{R}}_{Q,i}]_{kk}^{\top}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}[\boldsymbol{\mathcal{R}}_{Q,i}]_{kk}\boldsymbol{H}_{k,i}\} \tag{3.121}$$

Substituting $\boldsymbol{\mathcal{R}}_{Q,i}$ by its expression (3.82) leads to:

$$[\boldsymbol{P}_1]_{kk} = \sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{H}_{k,i}\boldsymbol{R}_{u_m,i}\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\boldsymbol{Q}_{n,i}\boldsymbol{R}_{u_n,i}\boldsymbol{H}_{k,i}\}$$

We rewrite $[\boldsymbol{P}_1]_{kk}$ as a sum of two terms, one for $m = n$ and one for $m \neq n$. Using (3.11), we get:

$$\begin{aligned}[\boldsymbol{P}_1]_{kk} = &\sum_{m=1}^{N} c_{mk}^2\mathbb{E}\{\boldsymbol{H}_{k,i}\boldsymbol{R}_{u_m,i}\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\boldsymbol{Q}_{m,i}\boldsymbol{R}_{u_m,i}\boldsymbol{H}_{k,i}\} \\ &+ \left(\tfrac{M_\nabla}{L}\right)^2\Big(\textstyle\sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{H}_{k,i}\boldsymbol{R}_{u_m,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\boldsymbol{R}_{u_n,i}\boldsymbol{H}_{k,i}\} - \\ &\sum_{m=1}^{N} c_{mk}^2\mathbb{E}\{\boldsymbol{H}_{k,i}\boldsymbol{R}_{u_m,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\boldsymbol{R}_{u_m,i}\boldsymbol{H}_{k,i}\}\Big)\end{aligned} \tag{3.122}$$

The terms in (3.122) depends of higher-order moments of the regression data. While we can continue the analysis by calculating these terms, it is sufficient for the exposition to focus on the case of sufficiently small step-sizes where a reasonable approximation is [Sayed, 2013a]:

$$\mathbb{E}\{\boldsymbol{R}_{u_m,i}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\boldsymbol{R}_{u_m,i}\} = \boldsymbol{R}_{u_m}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}\boldsymbol{R}_{u_m} \tag{3.123}$$

Note that this approximation will also be used in the sequel.

Finally, using 3.3, we can reformulate $\boldsymbol{P}_1$ as:

$$\begin{aligned}\boldsymbol{P}_1 = &\sum_{m=1}^{N} \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}_{c_m}\boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}})\boldsymbol{\mathcal{R}}_{c_m}\boldsymbol{\mathcal{H}}_i\} \\ &+ \left(\tfrac{M_\nabla}{L}\right)^2\Big(\mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{H}}_i\} - \textstyle\sum_{m=1}^{N}\mathbb{E}\{\boldsymbol{\mathcal{H}}_i\boldsymbol{\mathcal{R}}_{c_m}\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{c_m}\}\boldsymbol{\mathcal{H}}_i\Big)\end{aligned} \tag{3.124}$$

where the matrices $\boldsymbol{\mathcal{R}}_{c_k}$ are defined as:

$$\boldsymbol{\mathcal{R}}_{c_k} = \mathrm{diag}\{c_{k1}\boldsymbol{R}_{u_k}, \ldots, c_{kN}\boldsymbol{R}_{u_k}\} \tag{3.125}$$

Term $\boldsymbol{P}_2$ calculation Using the same steps as above, we have:

$$[\boldsymbol{P}_2]_{kk} = \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\mathcal{R}}_{Q,i}]_{kk}^{\top}[\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk}[\boldsymbol{\mathcal{Q}}_i']_{kk}\boldsymbol{R}_{u_k,i}\}$$

We substitute $[\boldsymbol{\mathcal{R}}_{Q,i}]_{kk}$ and $[\boldsymbol{\mathcal{Q}}_i']_{kk}$ by their respective definitions (3.82) and (3.83):

$$[\boldsymbol{P}_2]_{kk} = \sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{H}_{k,i}\boldsymbol{R}_{u_m,i}\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\left(\boldsymbol{I}_L - \boldsymbol{Q}_{n,i}\right)\boldsymbol{R}_{u_k,i}\}$$

Using (3.11) we find that:

$$\begin{aligned}[\boldsymbol{P}_2]_{kk} = &\tfrac{MM_\nabla}{L^2}\textstyle\sum_{m=1}^{N} c_{mk}^2\mathbb{E}\{\boldsymbol{R}_{u_m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{R}_{u_k,i}\}\\ &-\tfrac{M}{L}\textstyle\sum_{m=1}^{N} c_{mk}^2\mathbb{E}\{\boldsymbol{R}_{u_m,i}\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{Q}_{m,i}\boldsymbol{R}_{u_k,i}\}\\ &+\tfrac{MM_\nabla}{L^2}\left(1-\tfrac{M_\nabla}{L}\right)\Big(\textstyle\sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{R}_{u_m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{R}_{u_k,i}\}\\ &-\sum_{m=1}^{N} c_{mk}^2\mathbb{E}\{\boldsymbol{R}_{u_m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{R}_{u_k,i}\}\Big)\end{aligned}$$

Finally, we write:

$$\begin{aligned}\boldsymbol{P}_2 = &\tfrac{M_\nabla M}{L^2}\boldsymbol{\mathcal{R}}_2\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_u - \tfrac{M}{L}\boldsymbol{\mathcal{R}}_2\boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}\Sigma\mathcal{M}})\boldsymbol{\mathcal{R}}_u\\ &+\tfrac{MM_\nabla}{L^2}\left(1-\tfrac{M_\nabla}{L}\right)(\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_u - \boldsymbol{\mathcal{R}}_2\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_u)\end{aligned} \tag{3.126}$$

where

$$\boldsymbol{\mathcal{R}}_2 = \left\{\sum_{m=1}^{N} c_{m1}^2\boldsymbol{R}_{u_m},\ldots,\sum_{m=1}^{N} c_{mN}^2\boldsymbol{R}_{u_m}\right\} \tag{3.127}$$

Term $\boldsymbol{P}_3$ calculation Term $\boldsymbol{P}_3$ can be expressed as

$$[\boldsymbol{P}_3]_{k\ell} = \mathbb{E}\{\boldsymbol{H}_{k,i}[\boldsymbol{\mathcal{R}}_{Q,i}]_{kk}^{\top}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}[\boldsymbol{\mathcal{R}}_{Q(I-H),i}]_{k\ell}\} \tag{3.128}$$

Replacing $[\boldsymbol{\mathcal{R}}_{Q,i}]_{kk}$ and $[\boldsymbol{\mathcal{R}}_{Q(I-H),i}]$ by their definitions (3.82) and (3.85), respectively, we get:

$$[\boldsymbol{P}_3]_{k\ell} = \sum_{m=1}^{N} c_{mk}c_{\ell k}\mathbb{E}\{\boldsymbol{H}_{k,i}\boldsymbol{R}_{u_m,i}\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i}\left(\boldsymbol{I}_L - \boldsymbol{H}_{k,i}\right)\}$$

Applying the results from Table 3.2 leads to:

$$\begin{aligned}
[\boldsymbol{P}_3]_{k\ell} = & \tfrac{M}{L}\, c_{\ell k}^2\, \mathbb{E}\{\boldsymbol{R}_{u_\ell} \boldsymbol{Q}_{\ell,i} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \boldsymbol{Q}_{\ell,i} \boldsymbol{R}_{u_\ell}\} \\
& - c_{\ell k}^2\, \mathbb{E}\{\boldsymbol{H}_{k,i} \boldsymbol{R}_{u_\ell} \boldsymbol{Q}_{\ell,i} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \boldsymbol{Q}_{\ell,i} \boldsymbol{R}_{u_\ell} \boldsymbol{H}_{k,i}\} \\
& + \left(\tfrac{M_\nabla}{L}\right)^2 \Big(\tfrac{M}{L} \textstyle\sum_{m=1}^N c_{mk} c_{\ell k} \boldsymbol{R}_{u_m} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \boldsymbol{R}_{u_\ell} \\
& - \sum_{m=1}^N c_{mk} c_{\ell k} \mathbb{E}\{\boldsymbol{H}_{k,i} \boldsymbol{R}_{u_m} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \boldsymbol{R}_{u_\ell} \boldsymbol{H}_{k,i}\} \\
& - \tfrac{M}{L}\, c_{\ell k}^2\, \boldsymbol{R}_{u_\ell} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \boldsymbol{R}_{u_\ell} \\
& + c_{\ell k}^2\, \mathbb{E}\{\boldsymbol{H}_{k,i} \boldsymbol{R}_{u_\ell} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \boldsymbol{R}_{u_\ell} \boldsymbol{H}_{k,i}\} \Big)
\end{aligned} \tag{3.129}$$

We have:

$$c_{\ell k}^2\, \boldsymbol{R}_{u_\ell} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \boldsymbol{R}_{u_\ell} = \sum_{m=1}^N [\boldsymbol{\mathcal{R}}'_{u_m} \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}_2^\top \boldsymbol{\mathcal{I}}_m \boldsymbol{\mathcal{R}}_u]_{k\ell} \tag{3.130}$$

where

$$\boldsymbol{\mathcal{R}}'_{u_m} = \boldsymbol{I}_L \otimes \boldsymbol{R}_{u_m} \tag{3.131}$$

$$\boldsymbol{\mathcal{I}}_m = \mathrm{diag}\{0, 0, \ldots, \boldsymbol{I}_L, 0, \ldots, 0\} \tag{3.132}$$

All the entries of the matrix $\boldsymbol{\mathcal{I}}_m$ are equal to zero except the (m,m)-th block which is equal to $\boldsymbol{I}_L$. Using (3.130), we find that:

$$\begin{aligned}
\boldsymbol{P}_3 = & \tfrac{M}{L} \textstyle\sum_{m=1}^N \boldsymbol{\mathcal{R}}'_{u_m} \boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}) \boldsymbol{\mathcal{C}}_2^\top \boldsymbol{\mathcal{I}}_m \boldsymbol{\mathcal{R}}_u \\
& - \sum_{m=1}^N \boldsymbol{\varphi}_H(\boldsymbol{\mathcal{R}}'_{u_m} \boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}) \boldsymbol{\mathcal{C}}_2^\top \boldsymbol{\mathcal{I}}_m \boldsymbol{\mathcal{R}}_u) \\
& + \left(\tfrac{M_\nabla}{L}\right)^2 \Bigg(\tfrac{M}{L} \boldsymbol{\mathcal{R}} \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}^\top \boldsymbol{\mathcal{R}}_u - \boldsymbol{\varphi}_H(\boldsymbol{\mathcal{R}} \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}^\top \boldsymbol{\mathcal{R}}_u) \\
& - \tfrac{M}{L} \textstyle\sum_{m=1}^N \boldsymbol{\mathcal{R}}'_{u_m} \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}_2^\top \boldsymbol{\mathcal{I}}_m \boldsymbol{\mathcal{R}}_u \\
& + \sum_{m=1}^N \boldsymbol{\varphi}_H(\boldsymbol{\mathcal{R}}'_{u_m} \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{C}}_2^\top \boldsymbol{\mathcal{I}}_m \boldsymbol{\mathcal{R}}_u) \Bigg)
\end{aligned} \tag{3.133}$$

Term $\boldsymbol{P}_4$ calculation We express $\boldsymbol{P}_4$ as follows:

$$[\boldsymbol{P}_4]_{kk} = \mathbb{E}\{\boldsymbol{R}_{u_k,i} [\boldsymbol{\mathcal{Q}}'_i]_{kk} [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} [\boldsymbol{\mathcal{Q}}'_i]_{kk} \boldsymbol{R}_{u_k,i}\}$$

Substituting $[\boldsymbol{\mathcal{Q}}'_i]_{kk}$ by its definition (3.83) we get:

$$[\boldsymbol{P}_4]_{kk} = \sum_{m,n=1}^N c_{mk} c_{nk} \mathbb{E}\{\boldsymbol{R}_{u_k,i} \left(\boldsymbol{I}_L - \boldsymbol{Q}_{m,i}\right) [\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}]_{kk} \left(\boldsymbol{I}_L - \boldsymbol{Q}_{n,i}\right) \boldsymbol{R}_{u_k,i}\}$$

Using (3.11) leads to

$$[\boldsymbol{P}_4]_{kk} = \left(1 - 2\tfrac{M_\nabla}{L}\right) \sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{R}_{u_k,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{R}_{u_k,i}\} + \sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{R}_{u_k,i}\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{Q}_{n,i}\boldsymbol{R}_{u_k,i}\}$$

We rearrange the sum as follows:

$$\begin{aligned}[\boldsymbol{P}_4]_{kk} = &\left(1 - 2\tfrac{M_\nabla}{L}\right) \sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{R}_{u_k,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{R}_{u_k,i}\} \\ &+ \sum_{m=1}^{N} c_{mk}^2\mathbb{E}\{\boldsymbol{R}_{u_k,i}\boldsymbol{Q}_{m,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{Q}_{m,i}\boldsymbol{R}_{u_k,i}\} \\ &+ \left(\tfrac{M_\nabla}{L}\right)^2 \left(\sum_{m,n=1}^{N} c_{mk}c_{nk}\mathbb{E}\{\boldsymbol{R}_{u_k,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{R}_{u_k,i}\} \right. \\ &\left. - \sum_{m=1}^{N} c_{mk}^2\mathbb{E}\{\boldsymbol{R}_{u_k,i}[\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk}\boldsymbol{R}_{u_k,i}\} \right)\end{aligned}$$

Finally, we can write $\boldsymbol{P}_4$ in a compact form:

$$\begin{aligned}\boldsymbol{P}_4 = &\left(1 - 2\tfrac{M_\nabla}{L}\right) \boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_u + \boldsymbol{\mathcal{R}}_u(\boldsymbol{I}_{NL} \odot \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{C}}^\top)\,\boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}\Sigma\mathcal{M}})\boldsymbol{\mathcal{R}}_u \\ &+ \left(\tfrac{M_\nabla}{L}\right)^2 (\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_u - \boldsymbol{\mathcal{R}}_u(\boldsymbol{I}_{NL} \odot \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{C}}^\top)\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}\boldsymbol{\mathcal{R}}_u\Big)\end{aligned} \tag{3.134}$$

Term $\boldsymbol{P}_5$ calculation Expanding $\boldsymbol{P}_5$ leads to:

$$\boldsymbol{P}_5 = \boldsymbol{P}_{5,1} - \boldsymbol{P}_{5,2} - \boldsymbol{P}_{5,3} + \boldsymbol{P}_{5,4} \tag{3.135}$$

where:

$$\boldsymbol{P}_{5,1} = \sum_{m=1}^{N} c_{mk} c_{\ell k} \mathbb{E}\{\boldsymbol{R}_{u_k,i} [\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk} \boldsymbol{Q}_{\ell,i} \boldsymbol{R}_{u_\ell,i}\} \tag{3.136}$$

$$\boldsymbol{P}_{5,2} = \sum_{m=1}^{N} c_{mk} c_{\ell k} \mathbb{E}\{\boldsymbol{R}_{u_k,i} \boldsymbol{Q}_{m,i} [\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk} \boldsymbol{Q}_{\ell,i} \boldsymbol{R}_{u_\ell,i}\} \tag{3.137}$$

$$\boldsymbol{P}_{5,3} = \sum_{m=1}^{N} c_{mk} c_{\ell k} \mathbb{E}\{\boldsymbol{R}_{u_k,i} [\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk} \boldsymbol{Q}_{\ell,i} \boldsymbol{R}_{u_\ell,i} \boldsymbol{H}_{k,i}\} \tag{3.138}$$

$$\boldsymbol{P}_{5,4} = \sum_{m=1}^{N} c_{mk} c_{\ell k} \mathbb{E}\{\boldsymbol{R}_{u_k,i} \boldsymbol{Q}_{m,i} [\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}]_{kk} \boldsymbol{Q}_{\ell,i} \boldsymbol{R}_{u_\ell,i} \boldsymbol{H}_{k,i}\} \tag{3.139}$$

Following the same steps as earlier, and using the results from Table 3.2, leads to:

$$\boldsymbol{P}_{5,1} = \tfrac{M_\nabla}{L} \boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}^\top \boldsymbol{\mathcal{R}}_u \tag{3.140}$$

$$\begin{aligned} \boldsymbol{P}_{5,2} = \tfrac{M}{L} \Big[& \boldsymbol{\mathcal{R}}_u \boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}) \boldsymbol{\mathcal{C}}_2^\top \boldsymbol{\mathcal{R}}_u \\ & + \left(\tfrac{M_\nabla}{L}\right)^2 \left(\boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}^\top \boldsymbol{\mathcal{R}}_u - \boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}_2^\top \boldsymbol{\mathcal{R}}_u \right) \Big] \end{aligned} \tag{3.141}$$

$$\boldsymbol{P}_{5,3} = \tfrac{M}{L} \boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}^\top \boldsymbol{\mathcal{R}}_u \tag{3.142}$$

$$\boldsymbol{P}_{5,4} = \tfrac{M}{L} \boldsymbol{\varphi}_Q(\boldsymbol{\mathcal{M}\Sigma\mathcal{M}}) \boldsymbol{\mathcal{C}}_2^\top \boldsymbol{R}_u + \tfrac{M}{L} \left(\tfrac{M_\nabla}{L}\right)^2 \left(\boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}^\top \boldsymbol{R}_u - \boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}_2^\top \boldsymbol{R}_u \right)$$

Term $\boldsymbol{P}_6$ calculation Proceeding as previously we find:

$$\begin{aligned} \boldsymbol{P}_6 = & \left(1 - \tfrac{2M}{L}\right) \boldsymbol{\mathcal{R}}_u \mathbb{E}\{\boldsymbol{\mathcal{Q}}_i \boldsymbol{\mathcal{C}\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}^\top \boldsymbol{\mathcal{Q}}_i\} \boldsymbol{\mathcal{R}}_u \\ & + \boldsymbol{\varphi}_H(\boldsymbol{\mathcal{R}}_u \mathbb{E}\{\boldsymbol{\mathcal{Q}}_i \boldsymbol{\mathcal{C}\mathcal{M}\Sigma\mathcal{M}\mathcal{C}}^\top \boldsymbol{\mathcal{Q}}_i\} \boldsymbol{\mathcal{R}}_u) \end{aligned} \tag{3.143}$$

Following the same reasoning as in [Sayed, 2013a], we express $\boldsymbol{\Sigma}'$ in a vector form as:

$$\boldsymbol{\sigma}'_{DCD} = \boldsymbol{\mathcal{F}}_{DCD} \boldsymbol{\sigma} \tag{3.144}$$

where

$$\begin{aligned} \boldsymbol{\sigma} &= \mathrm{vec}(\boldsymbol{\Sigma}) \\ \boldsymbol{\sigma}'_{DCD} &= \mathrm{vec}(\boldsymbol{\Sigma}'_{DCD}) \end{aligned}$$

and the coefficient matrix $\boldsymbol{\mathcal{F}}_{DCD}$ of size $(MN)^2 \times (MN)^2$ is defined as:

$$\begin{aligned}\boldsymbol{\mathcal{F}}_{DCD} = \boldsymbol{I}_{(NM)^2} &- \tfrac{MM_\nabla}{L^2}\boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{I}_{LN} - \left(1 - \tfrac{M_\nabla}{L}\right)\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{M}} \otimes \boldsymbol{I}_{LN} \\ &- \tfrac{M_\nabla}{L}\left(1 - \tfrac{M}{L}\right)\boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}} \otimes \boldsymbol{I}_{LN} \\ &- \tfrac{MM_\nabla}{L^2}\boldsymbol{I}_{LN} \otimes \boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{M}} - \left(1 - \tfrac{M_\nabla}{L}\right)\boldsymbol{I}_{LN} \otimes \boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{M}} \\ &- \tfrac{M_\nabla}{L}\left(1 - \tfrac{M}{L}\right)\boldsymbol{I}_{LN} \otimes \boldsymbol{\mathcal{R}}_u\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{M}} + \textstyle\sum_{j=1}^{6}\boldsymbol{Z}_j + \boldsymbol{Z}_{2^\top} + \boldsymbol{Z}_{3^\top} + \boldsymbol{Z}_{5^\top}\end{aligned} \tag{3.145}$$

where the matrices $\boldsymbol{Z}_j$ and $\boldsymbol{Z}_{j^\top}$ are obtained when applying the vec$(\cdot)$ operator to $\boldsymbol{P}_j$ and $\boldsymbol{P}_j^\top$, respectively.

Substituting (3.100) and (3.113) into (3.99), and applying the vec$(\cdot)$ operator to both sides, we get:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\widetilde{\boldsymbol{w}}_{i-1}\|^2_{\boldsymbol{\mathcal{F}}_{DCD}\boldsymbol{\sigma}} + \text{trace}\big(\mathbb{E}\{\boldsymbol{\mathcal{G}}^\top_{DCD,i}\boldsymbol{\Sigma}\,\boldsymbol{\mathcal{G}}_{DCD,i}\}\,\boldsymbol{\mathcal{S}}\big) \tag{3.146}$$

Using (3.146), it is possible to extract useful information about the network or a specific node. For instance, we calculate the network mean square deviation or excess mean square error by setting $\boldsymbol{\Sigma} = \boldsymbol{I}_{LN}$ and $\boldsymbol{\Sigma} = \boldsymbol{\mathcal{R}}_u$, respectively. The DCD can be seen as an extension of the diffusion LMS in the case where the weighting matrix $\boldsymbol{A}$ is the identity matrix. Indeed, it is possible to recover the diffusion LMS, and derive other variants such as the compressed diffusion LMS, by properly setting matrices $\{\boldsymbol{H}_{k,i}, \boldsymbol{Q}_{k,i}\}$ and parameters $\{M, M_\nabla\}$.

As the steady and transient states analysis of this algorithm are similar to the CD and diffusion LMS ones from chapter 2 and [Sayed, 2013b], we chose not to repeat them.

3.3 Numerical analysis

In this section we evaluate the accuracy of the theoretical models for the proposed algorithms. Then, we shall perform different experiments to characterise their performances. This section shall be structured in two parts. In the first part, we will perform two experiments, where we first evaluate the accuracy of the mean-square error behaviour models for the compressed version of diffusion LMS namely *CD* and *DCD*. Then, we characterise the performance of the CD and DCD algorithms for different compression ratios. Note that the compression ratios of the CD and DCD algorithms are equal to $\frac{2L}{M+L}$ and $\frac{2L}{M+M_\nabla}$. In the final part, we will consider a realistic scenario where the compression is used as an energy preserving algorithm to address the shortage of energy resources.

3.3.1 Compressed diffusion numerical analysis

For this part we first consider a small network to validate the theoretical model. Then, we use a larger network and high dimensional measurements with the aim of testing the algorithm in a larger scale setting. For both experiments, the parameter vectors $\boldsymbol{w}^o$ were generated from a zero-mean

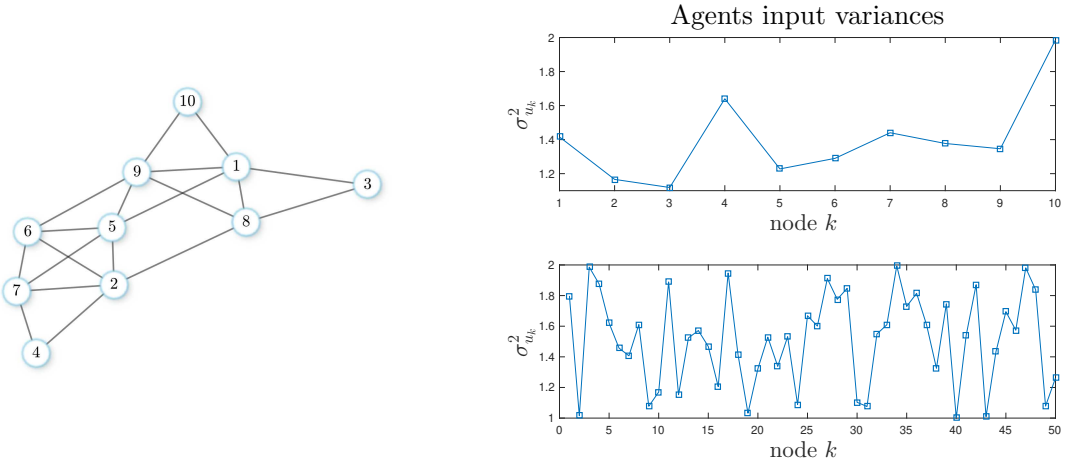

FIGURE 3.2: (left) Network topology. (right) Variance $\sigma^2_{u_k}$ of regressors for the theoretical model validation where we considered a network of $N = 10$ (top). On the bottom part we depict the variances σ^2_{uk} in the case where we considered a larger network of $N = 50$ agents.

Gaussian distribution. The input data $\boldsymbol{u}_{k,i}$ were drawn from zero-mean Gaussian distributions, with covariance $\boldsymbol{R}_{u,k} = \sigma^2_{u,k}\boldsymbol{I}_L$ reported in Figure 3.2 (right). The weighting matrices $\boldsymbol{C}$ were generated using the Metropolis rule [Sayed, 2013a]. Noises $v_k(i)$ were zero-mean, i.i.d. and Gaussian distributed with variance $\sigma^2_{v,k} = 10^{-3}$. Simulation results were averaged over 100 Monte-Carlo runs.

Theoretical model validation

We considered the network with $N = 10$ nodes depicted in Figure 3.2 (left). We set the parameters as follows: $\mu_k = 10^{-3}$, $L = 5$, $M = 3$, $M_\nabla = 1$. This resulted in compression ratio of $\frac{10}{8}$ and $\frac{10}{4}$ for compressed diffusion and doubly compressed diffusion LMS, respectively. It can be observed in Figure 3.3 (left) that the theoretical model accurately fits the simulated results. Unsurprisingly, the diffusion LMS algorithm outperformed its compressed counterparts at the expense of a higher communication load.

Algorithm performance for large networks and data sizes

Since compression is particularly relevant for relatively large data flows, we considered a network with $N = 50$ agents. We set the algorithm parameters as follows: $\mu_k = 3 \cdot 10^{-2}$ and $L = 50$. Due to the high dimensionality of the matrix $\boldsymbol{\mathcal{F}}$ ($2500^2 \times 2500^2$), we only performed Monte-Carlo simulations using C language scripts. Figures 3.4 and 3.5 depict the performance of the algorithms for different compression ratios. The largest compression ratio that can be reached by the CD algorithm equals $\frac{100}{55}$ as it transmits the whole gradient vectors ($\boldsymbol{Q}_{\ell,i} = \boldsymbol{I}_L$). On the other hand, the CDC algorithm offers more flexibility and can adapt to the network communication load by adjusting M and M_∇.

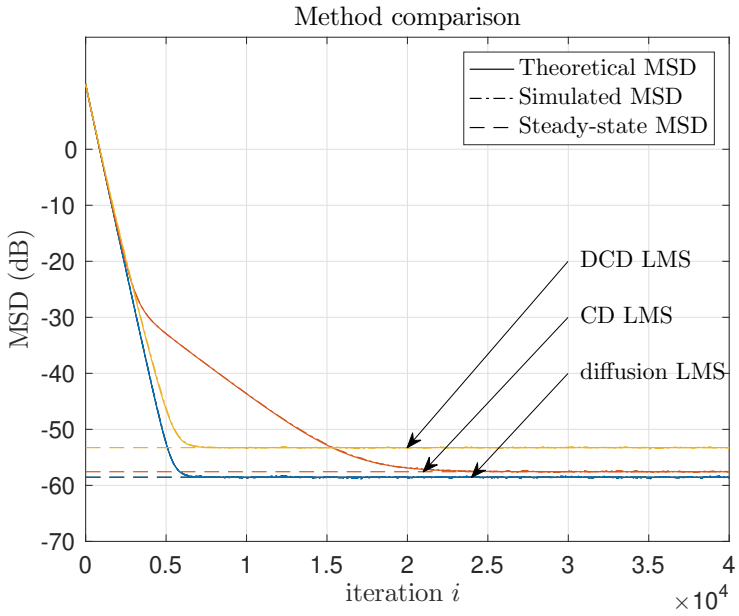

Figure 3.3: Theoretical and simulated MSD curves for diffusion LMS and its compressed versions. We considered a network of $N = 10$, data dimension $L = 5$, step-size parameters $\mu_k = 10^{-3}$. We chose to share $M = 3$ entries of the local estimate vectors, for both versions of the compressed diffusion LMS. For the DCLMS we share $M_\nabla = 1$ entries of the stochastic gradient vectors.

3.3.2 Algorithm performance for energy aware networks

We shall now consider a realistic scenario with respect to the energy consumption. Indeed, in a realistic wireless sensor network (WSN) implementation, nodes have limited energy reserves and cannot be active all the time. One of the most promising solution for this issue is to adopt an ENO strategy, where ENO stands for Energy Neutral Operation. In other words, the agents consume at most the amount of energy they harvest, hence achieving the neutral energy condition. Theoretically, neutral energy condition guarantees an infinite sensor lifetime. In order to implement an ENO strategy, nodes must be endowed with energy harvesting and storage capabilities. Agents alternate between two phases: a brief active phase and a sleeping phase. During the active phase, each agent k performs its assigned tasks and calculates the duration $T_{s_k,i}$ of the sleeping phase based on the available energy, the consumed energy and an estimate of the energy that will be harvested [Le et al., 2013]. For the sake of limiting energy consumption, the agents then switch to sleep mode for a duration of $T_{s_k,i}$. The corresponding DCD based algorithm is presented in 3.3.

We considered a solar energy based WSN with Bluetooth capabilities. To calculate $T_{s_k,i}$, we used [Le et al., 2013]:

$$T_{s_k,i} = \frac{e_{c_k,i} - \eta\, e_{s_k,i}}{\eta\,(P_{\text{harv},k,i} - P_{\text{leak}}) - P_{\text{sleep}}} \tag{3.147}$$

where $e_{c_k,i}$ and $e_{s_k,i}$ denote the consumed energy and the stored energy, respectively, η is the power manager efficiency, $P_{\text{harv},k,i}$ is the harvested power, P_{leak} is the capacitor leakage power, and P_{sleep} is the power consumed during sleep phase. These parameters and other parameters used for the experiment are defined in Table 3.4.

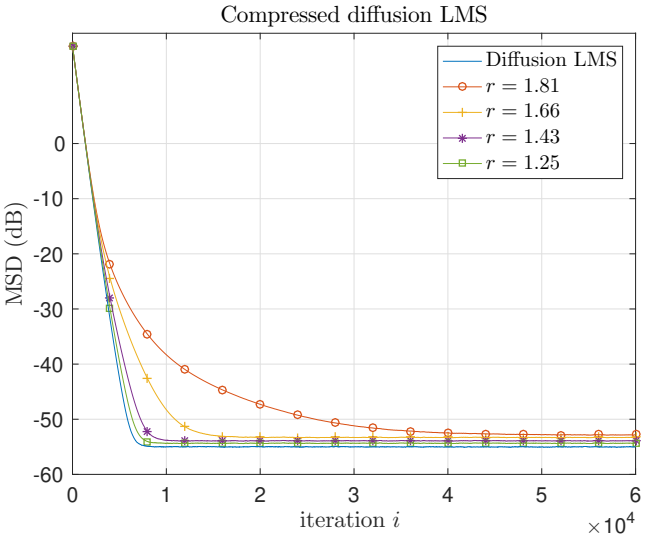

FIGURE 3.4: Evolution of the mean square deviation (MSD) as a function of the compression ratio for *compressed diffusion LMS*. We have a network of $N = 50$ agents and data dimension $L = 50$. We set the step-size parameters to $\mu_k = 10^{-2}$. We use a similar data profile as the previous experiment.

Following [Le et al., 2013], we estimated $e_{c_k,i}$ as follows:

$$e_{c_k,i} = e_a + P_{\text{sleep}} T_{s_k,i-1} \tag{3.148}$$

where e_a denotes the consumed energy during the active phase, assumed to be constant and known, and $P_{\text{sleep}} T_{s_k,i-1}$ is a prediction of the consumed energy during the sleep phase i based on the duration of the sleep phase $i-1$. Quantity e_a depends on the algorithm. It is essentially dictated by the volume of transferred data because of the excessive energy consumption of the Bluetooth module. As P_{sleep}, it was determined based on our own measurements and an estimation of the number of frames sent by each algorithm. See Table 3.4.

Finally, we considered the following law to simulate the amount of harvested energy:

$$E_{\text{harv},k,i} = \max(0, E_0 \sin(2\pi f i) + n(i)) \tag{3.149}$$

with $E_0 = 0.67J$, $E_{\text{harv},k,i}$ the harvested energy at node k and time i, $f = 10^{-5}$ a frequency, and $n_k(i)$ a zero-mean Gaussian noise with variance $\sigma_n^2 = 10^{-6}$. Note that the additive noise was used to diversify the amount of harvested energy during the Monte-Carlo runs. While it would have been possible to use a constant value over time for the harvested energy, we induced periodicity through the $\sin(\cdot)$ function to roughly model solar energy.

We considered the network in Figure 3.6. It consists of $N = 80$ agents scattered over a hill with different lighting levels. We set L to 40. To compare the algorithms, we set their compression ratio to $r = 20$. One exception was made for the CD algorithm. As parameter r cannot reach such a large value, it was set to $r = \frac{80}{65}$. Next the step size of each algorithm was set according to Table 3.5 in order to reach the same steady-state MSD. When $\boldsymbol{A} \neq \boldsymbol{I}_L$, matrix $\boldsymbol{A}$ was set using the Metropolis rule [Sayed, 2013a].

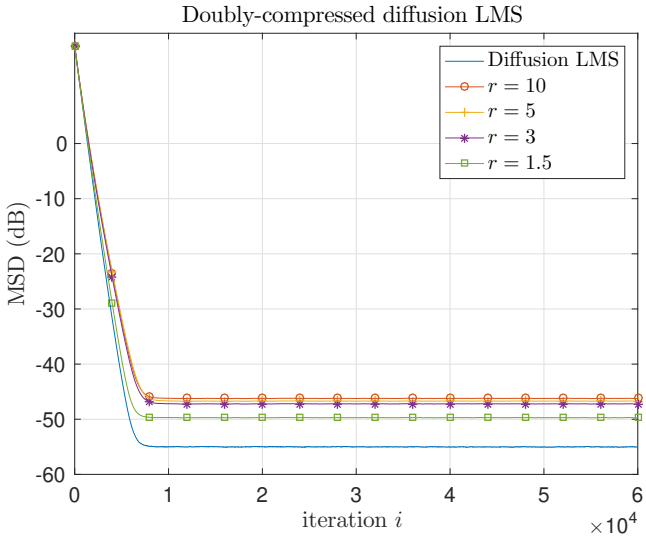

Figure 3.5: Evolution of the MSD as a function of the compression ratio for *doubly-compressed diffusion LMS*. We use the same parameters as the CD algorithm described in Figure 3.4.

We shall now discuss the simulation results. Figure 3.7 shows that the sleep phase duration decreases as the amount of harvested energy increases, and conversely. Also note that, for all the algorithms, the sleep phase is longer at the beginning as a consequence of the limited amount of stored energy that is available. Next, the sleep phase duration drops down until it reaches the minimal sleep duration $T_{s_{\min}}$ if possible. The less energy an algorithm consumes, the faster the super capacitors charge, and the faster the sleep phase duration of the agents drop down. As a consequence, nodes can process larger amounts of data, which makes the convergence of the algorithm faster as confirmed in Figure 3.8. This can be observed with the diffusion LMS and the DC algorithm, which are outperformed by the other algorithms. Let us now focus on the partial diffusion LMS and the DCD algorithm ($\boldsymbol{A} \neq \boldsymbol{I}_L$). As their compression ratio r was set to same value for comparison purposes, and their consumed energy during the active phase is almost the same, their sleep phases in Figure 3.8 (centre) are superimposed. The DCD algorithm however outperformed the partial diffusion LMS, in particular because it is endowed with a gradient sharing mechanism. Both algorithms outperformed the reduced-communication diffusion LMS.

3.4 Conclusion

In this chapter, we tackled one of the most critical challenges brought up with the rise of the Internet of Things and WSN, to wit: energy efficiency. To address this challenge, we investigated a technique for *diffusion LMS* that consists of sharing partial data. We carried out an analysis of the stochastic behaviour of the proposed two algorithms in the mean and mean-square sense. Furthermore, in the last section we provided simulation results to illustrate the accuracy of the theoretical models and considered a realistic simulation where sensor nodes alternate between active and inactive phases. This experiment confirmed the efficiency of the proposed strategy.

Algorithm 3.3 Local updates at node k for the modified DCD

for $i = 1, \ldots$ **do**
  randomly generate $\boldsymbol{H}_{k,i}$ and $\boldsymbol{Q}_{k,i}$
  for $\ell \in \mathcal{N}_k \setminus \{k\}$ **do**
    send $\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i}$ to node ℓ
    receive from node ℓ the partial gradient vector:

$$\boldsymbol{Q}_{\ell,i}\hat{\nabla}_w J_\ell(\boldsymbol{H}_{k,i}\boldsymbol{w}_{k,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{k,i})\boldsymbol{w}_{\ell,i-1})$$

    complete the missing entries using those available at node k, which results in $\boldsymbol{g}_{\ell,i}$ defined in (3.79)
  end for
  update the intermediate estimate:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k}\boldsymbol{g}_{\ell,i}$$

  calculate the local estimate:

$$\boldsymbol{w}_{k,i} = a_{kk}\boldsymbol{\psi}_{k,i} + \sum_{\ell \in \mathcal{N}_k \setminus \{k\}} a_{\ell k}\left[\boldsymbol{H}_{\ell,i}\,\boldsymbol{w}_{\ell,i-1} + (\boldsymbol{I}_L - \boldsymbol{H}_{\ell,i})\,\boldsymbol{\psi}_{k,i}\right]$$

  switch and stay in sleep mode for $T_{s_k,i}$ seconds
end for

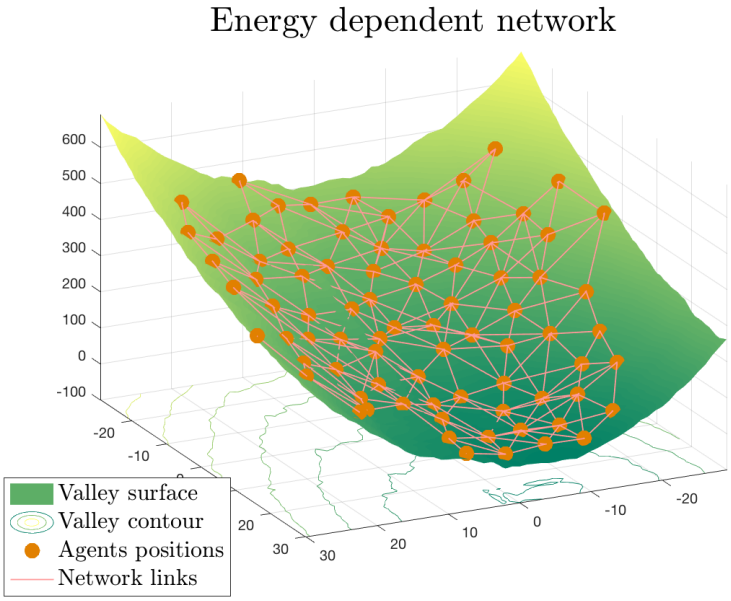

Figure 3.6: Network topology for WSN experiment. We have a network of $N = 80$ agents scattered of a valley. Under such setting, the agents receive different levels of solar energy.

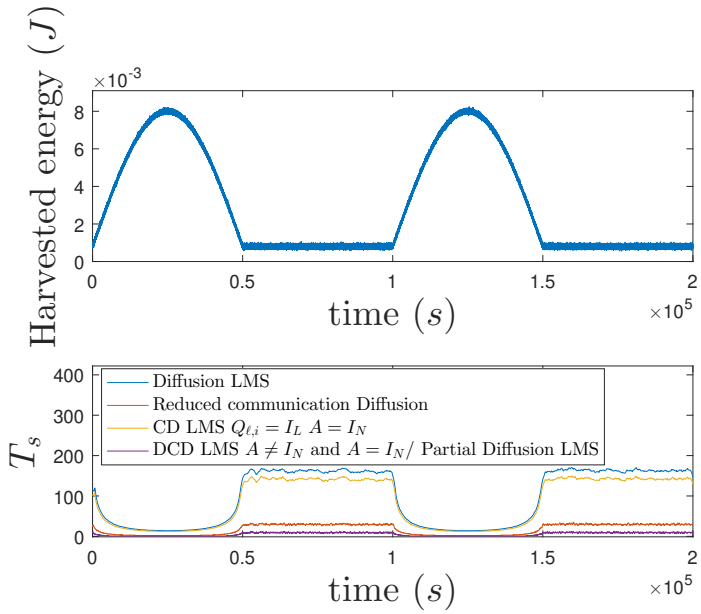

Figure 3.7: Harvested energy and sleep periods during the experimentations. On the upper half, we illustrate the profile of the harvested energy. We considered a simplistic model of the solar energy $E_{\text{harv},k,i} = \max(0, E_0 \sin(2\pi f i) + n(i))$. On the bottom half, we depict the sleep periods duration for all of the compared algorithms.

Table 3.3: List of symbols defined throughout the performance analysis chapter 3

Symbol	Equation
$\widetilde{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i}$	(3.36)
$\widetilde{\boldsymbol{w}}_i = \text{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\}$	(3.37)
$\boldsymbol{\mathcal{M}} = \text{diag}\{\mu_1 \boldsymbol{I}_L, \mu_2 \boldsymbol{I}_L, \ldots, \mu_N \boldsymbol{I}_L\}$	(3.38)
$\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_L$	(3.41)
$\boldsymbol{\mathcal{C}}_2 = \boldsymbol{\mathcal{C}} \odot \boldsymbol{\mathcal{C}}$	(3.110)
$\boldsymbol{\mathcal{R}}_i = \text{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell,1}\boldsymbol{R}_{u_\ell,i}, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell,N}\boldsymbol{R}_{u_\ell,i}\right\}$	(3.39)
$\boldsymbol{\mathcal{R}}_{u,i} = \text{diag}\{\boldsymbol{R}_{u_1,i}, \boldsymbol{R}_{u_2,i}, \ldots, \boldsymbol{R}_{u_N,i}\}$	(3.42)
$\boldsymbol{R}_{u_\ell,i} = \boldsymbol{u}_{\ell,i}\boldsymbol{u}_{\ell,i}^\top$	(3.43)
$[\boldsymbol{\mathcal{R}}_{m,i}]_{k\ell} = c_{\ell k}\boldsymbol{R}_{u_\ell,i}(\boldsymbol{I}_{NL} - \boldsymbol{H}_{k,i})$	(3.44)
$\boldsymbol{\mathcal{H}}_i = \text{diag}\{\boldsymbol{H}_{1,i}, \boldsymbol{H}_{2,i}, \ldots, \boldsymbol{H}_{N,i}\}$	(3.40)
$\boldsymbol{\mathcal{H}} = \frac{M}{L}\boldsymbol{I}_{NL}$	(3.50)
$\boldsymbol{\mathcal{B}}_{CD_i} = \boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_i\boldsymbol{\mathcal{H}}_i - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{m,i}$	(3.46)
$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top$	(3.47)
$\boldsymbol{s}_i = \text{col}\{\boldsymbol{u}_{1,i}v_1(i), \boldsymbol{u}_{2,i}v_2(i), \ldots, \boldsymbol{u}_{N,i}v_N(i)\}$	(3.48)
$\boldsymbol{\mathcal{S}} = \text{diag}(\sigma_{v,1}^2\boldsymbol{R}_{u,1}, \ldots, \sigma_{v,N}^2\boldsymbol{R}_{u,N})$	(3.58)
$\boldsymbol{\mathcal{R}} = \mathbb{E}\{\boldsymbol{\mathcal{R}}_i\} = \text{diag}\{\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N\}$	(3.51)
$\boldsymbol{\mathcal{R}}_u = \mathbb{E}\{\boldsymbol{\mathcal{R}}_{u,i}\} = \text{diag}\{\boldsymbol{R}_{u_1}, \boldsymbol{R}_{u_2}, \ldots, \boldsymbol{R}_{u_N}\}$	(3.52)
$\boldsymbol{R}_k = \sum_{\ell\in\mathcal{N}_k} c_{\ell,k}\boldsymbol{R}_{u_\ell}$	(3.53)
$\boldsymbol{\mathcal{R}}'_{u_m} = \boldsymbol{I}_L \otimes \boldsymbol{R}_{u_m}$	(3.131)
$\boldsymbol{\mathcal{I}}_m = \text{diag}\{0, 0, \ldots, \boldsymbol{I}_L, 0, \ldots, 0\}$	(3.132)
$\boldsymbol{\mathcal{R}}_{Q,i} = \text{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i}, \sum_{\ell\in\mathcal{N}_2} c_{\ell 2}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i}, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}\boldsymbol{Q}_{\ell,i}\boldsymbol{R}_{u_\ell,i}\right\}$	(3.82)
$\boldsymbol{\mathcal{Q}}'_i = \text{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i}), \sum_{\ell\in\mathcal{N}_2} c_{\ell 1}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i}) \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}(\boldsymbol{I}_L - \boldsymbol{Q}_{\ell,i})\right\}$	(3.83)
$\boldsymbol{\mathcal{Q}}_i = \text{diag}\{\boldsymbol{Q}_{1,i}, \boldsymbol{Q}_{2,i}, \ldots, \boldsymbol{Q}_{N,i}\}$	(3.84)
$\boldsymbol{\mathcal{Q}}_i = \text{diag}\{\boldsymbol{Q}_{1,i}, \boldsymbol{Q}_{2,i}, \ldots, \boldsymbol{Q}_{N,i}\}$	(3.84)
$\boldsymbol{\mathcal{B}}_{DCD_i} = \boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q,i}\boldsymbol{\mathcal{H}}_i - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}'_i\boldsymbol{\mathcal{R}}_{u,i} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{Q(I-H),i}$	(3.90)
$\boldsymbol{\mathcal{G}}_{DCD_i} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{Q}}_i + \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{Q}}'_i$	(3.91)
$\boldsymbol{\mathcal{B}}_{DCD} = \boldsymbol{I}_{NL} - \frac{MM_\nabla}{L^2}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}} - \left(1 - \frac{M_\nabla}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_u - \frac{M_\nabla}{L}\left(1 - \frac{M}{L}\right)\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top\boldsymbol{\mathcal{R}}_u$	(3.92)
$\boldsymbol{\mathcal{F}}_{CD}$	(3.67)
$\boldsymbol{\mathcal{F}}_{DCD}$	(3.145)

Table 3.4: Summary of the parameters used to determine the duration of sleeping phase T_s

parameter	description	value
C_s	super capacitor capacity	0.09 F
P_{leak}	super capacitor leakage power	$3.3 \cdot 10^{-6}$ W
P_{sleep}	consumed power for sleep mode	$3.01 \cdot 10^{-5}$ W
$T_{s_{\min}}$	minimal sleep time duration	1 s
$T_{s_{\max}}$	maximal sleep time duration	300 s
V_{ref}	minimal required voltage	3.5 V
$e_{a,\text{diff}}$	consumed energy for diffusion LMS	$8.58 \cdot 10^{-2}$ J
$e_{a,\text{RCD}}$	consumed energy for red. comm. LMS	$1.61 \cdot 10^{-2}$ J
$e_{a,\text{PM}}$	consumed energy for part. dif. LMS	$5.4 \cdot 10^{-3}$ J
$e_{a,\text{cmp}}$	consumed energy for CD LMS	$7.51 \cdot 10^{-2}$ J
$e_{a,\text{dcmp}}$	consumed energy for DCD LMS	$5.4 \cdot 10^{-3}$ J

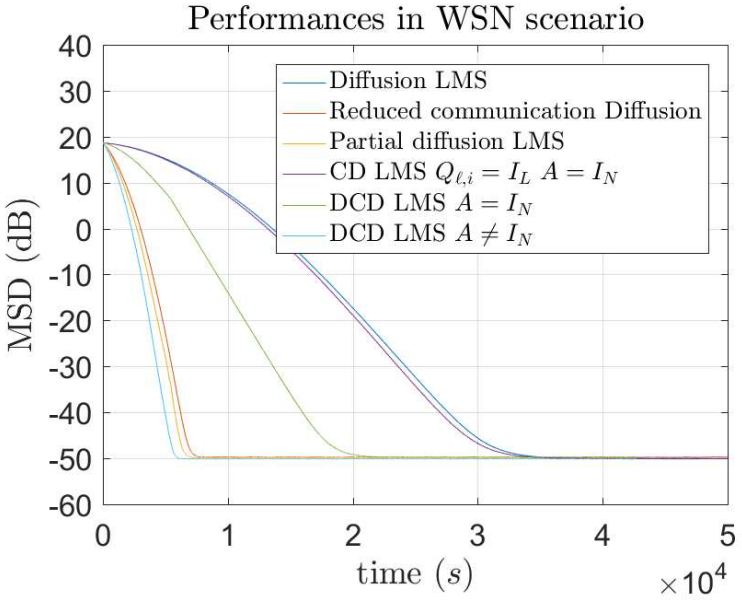

Figure 3.8: Simulated Mean square deviation under a realistic scenario. We consider a network of $N = 80$ agents and data dimension $L = 540$. The step-size parameters are summed up in Table 3.5. We use a similar data profile as the previous experiments.

Table 3.5: Step-size and compression settings for the different tested algorithms.

Algorithm	Step-size μ_k	Comp. ratio
Diffusion LMS	$5.4\,10^{-3}$	/
Reduced communication diffusion [Arablouei et al., 2015]	$1.14 \cdot 10^{-2}$	20
Partial diffusion LMS [Arablouei et al., 2014a]	$4.4 \cdot 10^{-3}$	20
Compressed diffusion LMS	$4.8 \cdot 10^{-2}$	$\frac{80}{65}$
Doubly-compressed diffusion LMS	$6 \cdot 10^{-3}$	20

4 Privacy aware diffusion LMS

Contents

Previously, we explored the compression as a way to reduce the network load. In the present chapter, we are interested in securing the transmitted data through projection onto a smaller subspace. Indeed, adaptive networks may raise significant privacy concerns about the observations that are collected and shared by the agents. Privacy preservation became an important issue in data mining with the advent of social networks and recommender systems [Ramakrishnan et al., 2001]. In order to prevent the disclosure of sensitive information during the learning process, privacy preservation aims at protecting the individual data by making their reconstruction difficult if impossible [Friedman, 2011, Agrawal and Srikant, 2000].

The work presented in this chapter was published in:

- Harrane, I. E. K., Flamary, R., and Richard, C. (2016b). Toward privacy-preserving diffusion strategies for adaptation and learning over networks. In *Proc. EUSIPCO*, Budapest, Hungary.

4.1 Introduction

Adaptive networks take advantage of their communication ability to collaborate and enhance their estimation performances. However, this ability could be a double edged sword when considering data privacy. Indeed, communicating information between multiple agents highly increases the risk of data privacy breaches. There are several privacy preserving data mining techniques. They can be classified according to the following five criterions [Verykios et al., 2004]: (*i*) availability of the data (centralized, distributed); (*ii*) sanitization procedure applied to the data (encryption, corruption, etc.); (*iii*) learning algorithm which the privacy preservation technique is designed for; (*iv*) data type (raw data or aggregated data); (*v*) privacy preservation technique used for the selective modification of the data.

Table 4.1: List of the symbols and notations used in chapter 4

Symbol	Definition
L	length of the parameter vectors
N	network agent count
$\mathcal{N}_k$	neighbourhood of the agent k including it self
$\boldsymbol{w}_{k,i}$	instantaneous estimate at the agent k
$\boldsymbol{w}^o$	optimum parameter vector
$d_k(i)$	reference signal for the agent k at the time instant i
$\boldsymbol{u}_{k,i}$	regression vector of the agent k
$v_k(i)$	additive noise at the agent k
$\boldsymbol{H}$	Wishart matrix used to alter the transmitted data
M_x	degree of freedom of the Wishart matrix

It is important to note that data modification results in degradation of the database performance. This study explores a sanitization procedure for privacy preservation over adaptive networks that consists of corrupting local raw data. Perturbation techniques include the use of additive noise to preserve data privacy while making sure that information can still be exploited by the data mining algorithms. It is worth mentioning that this principle was indirectly studied with diffusion LMS in the case where the additive noise that corrupts the data is caused by noisy transmission channels [Sayed, 2013a, Nassif et al., 2016a]. Nevertheless, it was demonstrated that in many cases, random additive distortion preserves very little data privacy [Kargupta et al., 2005]. Efficient alternatives that provide guarantees against privacy breaches via linear transformations exploit multiplicative perturbations [Chen and Liu, 2011, Chen and Liu, 2005, Chen and Liu, 2008]. Finally, an important step in the design of privacy-preserving algorithms is the identification of appropriate evaluation criteria. Recently, ϵ-differential privacy has been recognized as a meaningful criterion. It guarantees that presence or absence of an individual in a database does not affect the output of a data mining algorithm significantly. For what concerns us here, this criterion was considered in an online learning setting with random additive distortions [Jain et al., 2011] and in a distributed learning setting from finite distributed datasets [Rajkumar and Agarwal, 2012].

This work is a the stepping stone for building and deriving privacy-preserving diffusion strategies to address distributed inference problems in the case where agents are interested in preserving the privacy of local measurements. We introduce a diffusion LMS algorithm that corrupts the local measurements by multiplicative noise at each agent while ensuring the convergence of the algorithm to an unbiased solution. As for the previous chapters, we will theoretically analyse the algorithm in the mean and mean-square sense.

4.2 Diffusion LMS with privacy-preserving capabilities

Privacy preservation has become an important issue in many data mining applications. It aims at protecting the privacy of individual data in order to prevent the disclosure of sensitive information during the learning process. Taking inspiration from differential privacy, a possible strategy is to

use the data patterns locally without directly sharing the original data, and to guarantee that the process does not provide sufficient information to recover the original data.

This version of diffusion LMS ensures the data privacy by corrupting the local measurements $\{d_\ell(i), \boldsymbol{u}_{\ell,i}\}$ in (2.4) while ensuring the algorithm convergence towards an unbiased estimate of the solution of problem (3.8).

Let us consider the same setting as for the previous chapter 3, namely, a connected network of N nodes. The aim is to estimate an $L \times 1$ unknown vector from collected measurements. Each node k has access to temporal measurement sequences $\{d_k(i), \boldsymbol{u}_{k,i}\}$, with $d_k(i)$ denoting a reference signal, and $\boldsymbol{u}_{k,i}$ denoting an $L \times 1$ regression vector with covariance matrix $\boldsymbol{R}_{u,k} > 0$. The data at node k are assumed to be related via the linear regression model at time i:

$$d_k(i) = \boldsymbol{u}_{k,i}^\top \boldsymbol{w}^o + v_k(i) \tag{4.1}$$

To limit the amount of shared information, we shall assume that the weighting matrix in the combination step (2.4) $\boldsymbol{A}$ is set to the identity matrix. Diffusion LMS defined by the equations (2.4) and (2.5) reduces to:

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu_k \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1}) \tag{4.2}$$

where $\hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1}) = -\boldsymbol{u}_{\ell,i}[d_\ell(i) - \boldsymbol{u}_{\ell,i}^\top \boldsymbol{w}_{k,i-1}]$ denotes, the instantaneous approximation at time instant i of the gradient vector $\nabla_w J_\ell(\boldsymbol{w})$ evaluated at the point $\boldsymbol{w}_{k,i-1}$ by node ℓ.

In [Sayed, 2013a, Nassif et al., 2016a], an additive noise component is introduced into each step of the diffusion strategy to model noisy links between nodes. We shall not explore this strategy for privacy protection even though it is frequently used. It has been shown that in many situations the original data can be closely estimated from perturbed data using spectral filtering [Kargupta et al., 2005, Liu et al., 2008a]. We therefore propose to substitute $\hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1})$ in (4.2) by:

$$\boldsymbol{H}_{\ell,i} \hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1}) \tag{4.3}$$

before that node ℓ sends this information to node k, with $\boldsymbol{H}_{\ell,i}$ an $L \times L$ matrix defined as:

$$\boldsymbol{H}_{\ell,i} = \boldsymbol{X}_{\ell,i}^\top \boldsymbol{X}_{\ell,i} \tag{4.4}$$

where $\boldsymbol{X}_{\ell,i}$ is an $M_x \times L$ matrix. Each row of matrix $\boldsymbol{X}_{\ell,i}$ is independently drawn from an L-variate Gaussian distribution with zero mean and covariance $\boldsymbol{R}_{x,\ell}$. If $M_x \geq L$, $\boldsymbol{H}_{\ell,i}$ is said to be drawn from a Wishart distribution with M_x degrees of freedom and scale matrix $\boldsymbol{R}_{x,\ell}$. Otherwise, if $M_x < L$, then the Wishart no longer has a proper density. It is a singular distribution with values in a lower-dimension subspace of the space of $M_x \times M_x$ matrices.

Our motivations for exploring transformation (4.3) are two-fold. Firstly, $\boldsymbol{H}_{\ell,i}$ is a nonnegative matrix. Therefore, the conditional expectation of (4.3) given $\boldsymbol{H}_{\ell,i}$ and $\boldsymbol{w}_{k,i-1}$, namely,

$$\mathbb{E}\{\boldsymbol{H}_{\ell,i} \hat{\nabla}_w J_\ell(\boldsymbol{w}_{k,i-1}) | \boldsymbol{H}_{\ell,i}, \boldsymbol{w}_{k,i-1}\} = \boldsymbol{H}_{\ell,i} \nabla_w J_\ell(\boldsymbol{w}_{k,i-1}) \tag{4.5}$$

Table 4.2: List of symbols defined throughout the performance analysis chapter 4

Symbol	Equation
$\widetilde{\boldsymbol{w}}_i = \boldsymbol{w}^o - \boldsymbol{w}_{k,i}$	(4.10)
$\widetilde{\boldsymbol{w}}_{k,i} = \text{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\}$	(4.11)
$\boldsymbol{\mathcal{M}} = \text{diag}\{\mu_1 \boldsymbol{I}_M, \mu_2 \boldsymbol{I}_M, \ldots, \mu_N \boldsymbol{I}_M\}$	(4.12)
$\boldsymbol{\mathcal{R}}_i = \text{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\boldsymbol{u}_{\ell,i}\boldsymbol{u}_{\ell,i}^\top, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}\boldsymbol{u}_{\ell,i}\boldsymbol{u}_{\ell,i}^\top\right\}$	(4.13)
$\boldsymbol{\mathcal{H}}_{p,i} = \text{diag}\{\boldsymbol{H}_{1,i}, \boldsymbol{H}_{2,i}, \ldots, \boldsymbol{H}_{N,i}\}$	(4.14)
$\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_M$	(4.15)
$\boldsymbol{\mathcal{B}}_{p,i} = \boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{H,i}$	(4.17)
$\boldsymbol{\mathcal{G}}_{p,i} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{H}}_{p,i}\boldsymbol{\mathcal{C}}^\top$	(4.18)
$\boldsymbol{\mathcal{R}}_{H,i} = \text{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\boldsymbol{u}_{\ell,i}\boldsymbol{H}_{1,i}\boldsymbol{u}_{\ell,i}^\top, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}\boldsymbol{u}_{\ell,i}\boldsymbol{H}_{\ell,i}\boldsymbol{u}_{\ell,i}^\top\right\}$	(4.19)
$\boldsymbol{\mathcal{S}} = \text{diag}(\sigma_{v,1}^2\boldsymbol{R}_{u,1}, \ldots, \sigma_{v,N}^2\boldsymbol{R}_{u,N})$	(4.31)
$\boldsymbol{s}_i = \text{col}\{\boldsymbol{u}_{1,i}v_1(i), \boldsymbol{u}_{2,i}v_2(i), \ldots, \boldsymbol{u}_{N,i}v_N(i)\}$	(4.20)
$\boldsymbol{\mathcal{H}} = \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\} = \text{diag}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_N)$	(4.22)
$\boldsymbol{\mathcal{H}}_p = \mathbb{E}\{\boldsymbol{\mathcal{H}}_{p,i}\} = M_x\,\sigma_x^2\,\boldsymbol{I}_{NL}$	(4.24)
$\boldsymbol{\mathcal{R}} = \mathbb{E}\{\boldsymbol{\mathcal{R}}_i\} = \text{diag}(\boldsymbol{R}_{u,1}, \ldots, \boldsymbol{R}_{u,N})$	(4.23)
$\boldsymbol{\mathcal{K}}_i = \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{H}}_{p,i}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\,\boldsymbol{\mathcal{H}}_{p,i}\boldsymbol{\mathcal{C}}^\top$	(4.27)

is a descent direction [Sayed, 2003] provided that $\nabla_w J_\ell(\boldsymbol{w}_{k,i-1})$ is nonzero and does not lie in the null space of $\boldsymbol{H}_{\ell,i}$. Secondly, the parameter M_x allows to fix the rank of $\boldsymbol{H}_{\ell,i}$. This allows to balance the trade-off between privacy, in the case where $\boldsymbol{H}_{\ell,i}$ is rank-deficient, and convergence rate.

4.3 Theoretical analysis

In this section, we shall study the stochastic behaviour of the privacy-preserving diffusion LMS defined as:

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \sum_{\ell\in\mathcal{N}_k} c_{\ell k}\boldsymbol{H}_{\ell,i}\boldsymbol{u}_{\ell,i}[d_\ell(i) - \boldsymbol{u}_{\ell,i}^\top\boldsymbol{w}_{k,i-1}] \tag{4.6}$$

We first summarize some useful properties. For the sake of conciseness and simplicity, we shall consider that $\boldsymbol{R}_{x,\ell} = \sigma_x^2\boldsymbol{I}_M$ for all ℓ.

4.3.1 Preliminary properties of Wishart matrices

In order to analyse the algorithm, we need to recall the first and second-order moments of $\boldsymbol{H}_{\ell,i}$. For clarity, we drop the subscripts ℓ and i. Let $\boldsymbol{H}$ a random matrix drawn from a Wishart distribution of M_x degrees of freedom and a scale matrix $\boldsymbol{R}_x = \sigma_x^2\boldsymbol{I}_L$. We then have the mean of $\boldsymbol{H}$ formulated

as:

$$\mathbb{E}\{\boldsymbol{H}\} = M_x \boldsymbol{R}_x = M_x \sigma_x^2 \boldsymbol{I}_L \tag{4.7}$$

Consider now two independent matrices $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ drawn from Wishart distributions with M_x degrees of freedom and scale matrices $\boldsymbol{R}_x = \sigma_x^2 \boldsymbol{I}_L$. We have:

$$\operatorname{cov}\{(\boldsymbol{H}_1)_{ij}, (\boldsymbol{H}_2)_{k\ell}\} = M_x^2 \, \sigma_x^4 \, \delta_{ij} \, \delta_{k\ell} \tag{4.8}$$

where δ_{ij} stands for the Kronecker delta function. Finally, in the case $\boldsymbol{H} = \boldsymbol{H}_1 = \boldsymbol{H}_2$, by Isserlis' theorem we have:

$$\operatorname{cov}\{(\boldsymbol{H})_{ij},(\boldsymbol{H})_{k\ell}\} = M_x \, \sigma_x^4 \big(M_x \delta_{ij}\delta_{k\ell} + \delta_{ik}\delta_{j\ell} + \delta_{i\ell}\delta_{jk}\big) \tag{4.9}$$

4.3.2 Convergence in the mean

Before proceeding with the analysis of the algorithm, let us recall some important notations:

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{w}^o - \boldsymbol{w}_{k,i} \tag{4.10}$$

$$\widetilde{\boldsymbol{w}}_{k,i} = \operatorname{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\} \tag{4.11}$$

$$\boldsymbol{\mathcal{M}} = \operatorname{diag}\{\mu_1 \boldsymbol{I}_M, \mu_2 \boldsymbol{I}_M, \ldots, \mu_N \boldsymbol{I}_M\} \tag{4.12}$$

$$\boldsymbol{\mathcal{R}}_i = \operatorname{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\boldsymbol{u}_{\ell,i}\boldsymbol{u}_{\ell,i}^\top, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}\boldsymbol{u}_{\ell,i}\boldsymbol{u}_{\ell,i}^\top\right\} \tag{4.13}$$

$$\boldsymbol{\mathcal{H}}_{p,i} = \operatorname{diag}\{\boldsymbol{H}_{1,i}, \boldsymbol{H}_{2,i}, \ldots, \boldsymbol{H}_{N,i}\} \tag{4.14}$$

$$\boldsymbol{\mathcal{C}} = \boldsymbol{C} \otimes \boldsymbol{I}_M \tag{4.15}$$

and introduce the following assumptions:

Assumption 4.1. *The regression vectors $\boldsymbol{u}_{k,i}$ arise from a zero-mean random process that is temporally white and spatially independent. A direct consequence of this assumption is that $\boldsymbol{u}_{k,i}$ is independent of $\boldsymbol{w}_{\ell,j}$ for all ℓ and $j < i$.*

Assumption 4.2. *The additive noise signals $v_k(i)$ are temporally white and spatially independent zero-mean random variables.*

Assumption 4.3. *The rows of matrices $\boldsymbol{X}_{\ell,i}$ arise from zero-mean Gaussian processes that are temporally white, mutually independent, and independent of any other process.*

Using the definitions (4.10), (4.11) and the recursion (4.6) we get:

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{B}}_{p,i}\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{G}}_{p,i}\boldsymbol{s}_i \tag{4.16}$$

where

$$\boldsymbol{\mathcal{B}}_{p,i} = \boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{H,i} \tag{4.17}$$

$$\boldsymbol{\mathcal{G}}_{p,i} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{H}}_{p,i}\boldsymbol{\mathcal{C}}^{\top} \tag{4.18}$$

$$\boldsymbol{\mathcal{R}}_{H,i} = \text{diag}\left\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\boldsymbol{u}_{\ell,i}\boldsymbol{H}_{1,i}\boldsymbol{u}_{\ell,i}^{\top}, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}\boldsymbol{u}_{\ell,i}\boldsymbol{H}_{\ell,i}\boldsymbol{u}_{\ell,i}^{\top}\right\} \tag{4.19}$$

$$\boldsymbol{s}_i = \text{col}\{\boldsymbol{u}_{1,i}v_1(i), \boldsymbol{u}_{2,i}v_2(i), \ldots, \boldsymbol{u}_{N,i}v_N(i)\} \tag{4.20}$$

Under 4.1, $\boldsymbol{u}_{k,i}$ is independent of $\boldsymbol{w}_{\ell,j}$ for $i \geq j$ and for all ℓ. This assumption is commonly used in the adaptive filtering literature because it helps simplify the analysis. The performance results obtained under this assumption match well the actual performance of stand-alone filters for sufficiently small step-sizes.

Taking expectation of both sides of recursion (4.16), using Assumptions 4.1 – 4.3, and $\mathbb{E}\{\boldsymbol{s}_i\} = 0$, we find that:

$$\begin{aligned}\mathbb{E}\{\widetilde{\boldsymbol{w}}_i\} &= \left(\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\,\mathbb{E}\{\boldsymbol{\mathcal{R}}_{H,i}\}\right)\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}\} \\ &= \left(\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{H}}_p\boldsymbol{\mathcal{R}}\right)\mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}\}\end{aligned} \tag{4.21}$$

where

$$\boldsymbol{\mathcal{H}}_p = \mathbb{E}\{\boldsymbol{\mathcal{H}}_i\} = \text{diag}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_N) \tag{4.22}$$

$$\boldsymbol{\mathcal{R}} = \mathbb{E}\{\boldsymbol{\mathcal{R}}_i\} = \text{diag}(\boldsymbol{R}_{u,1}, \ldots, \boldsymbol{R}_{u,N}) \tag{4.23}$$

Let us now evaluate $\boldsymbol{\mathcal{H}}_p$. Since it is a block diagonal matrix, we can use the result (4.7) from the previous section:

$$\boldsymbol{\mathcal{H}}_p = \mathbb{E}\{\boldsymbol{\mathcal{H}}_{p,i}\} = M_x\,\sigma_x^2\,\boldsymbol{I}_{NL} \tag{4.24}$$

From (2.31), the algorithm asymptotically converges in the mean to $\boldsymbol{w}^o$ if and only if matrix $(\boldsymbol{I}_{NL} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{H}}_p\boldsymbol{\mathcal{R}})$ is stable, meaning that all its eigenvalues lie strictly inside the unit disc. This leads to the following condition on the step-size parameters μ_ℓ:

$$\mu_\ell < \frac{2}{\lambda_{\max}(M_x\,\sigma_x^2\,\boldsymbol{R}_{u,\ell})} \tag{4.25}$$

where $\lambda_{\max}(\cdot)$ stands for the maximum eigenvalue of its matrix argument [Sayed et al., 2013].

4.3.3 Mean-square stability

To analyse the mean-square-error stability, we evaluate the weighted mean-square deviation $\mathbb{E}\|\widetilde{\boldsymbol{w}}\|_{\boldsymbol{\Sigma}}^2$ where $\boldsymbol{\Sigma}$ denotes a nonnegative definite matrix with $L \times L$ block entries $[\boldsymbol{\Sigma}]_{k\ell}$. The freedom in selecting $\boldsymbol{\Sigma}$ allows us to extract various types of information about the network. From relation

(4.16) and using the Assumptions 4.1 – 4.3, we get:

$$\mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|_{\boldsymbol{\Sigma}}^2 = \mathbb{E}\{\widetilde{\boldsymbol{w}}_{i-1}^\top \boldsymbol{\mathcal{B}}_{p,i}^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_{p,i} \widetilde{\boldsymbol{w}}_{i-1}\} + \mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{G}}_{p,i}^\top \boldsymbol{\Sigma}\, \boldsymbol{\mathcal{G}}_{p,i} \boldsymbol{s}_i\} \tag{4.26}$$

Observe that the analysis of (4.26) is not a direct extension of the analysis of the diffusion LMS algorithm because of the presence of the stochastic matrix $\boldsymbol{\mathcal{H}}_{p,i}$.

Let us evaluate the last term in the right-hand side of (4.26). We introduce the following notations:

$$\boldsymbol{\mathcal{K}}_i = \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i} \boldsymbol{\mathcal{C}}^\top \tag{4.27}$$

The last expectation of (4.26) is given by:

$$\begin{aligned}
\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{K}}_i \boldsymbol{s}_i\} &= \mathrm{trace}\big(\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{K}}_i \boldsymbol{s}_i\}\big) && (4.28)\\
&= \mathrm{trace}\big(\mathbb{E}\{\boldsymbol{\mathcal{K}}_i\}\, \mathbb{E}\{\boldsymbol{s}_i \boldsymbol{s}_i^\top\}\big) && (4.29)\\
&= \mathrm{trace}\big(\mathbb{E}\{\boldsymbol{\mathcal{K}}_i\}\, \boldsymbol{\mathcal{S}}\big) && (4.30)
\end{aligned}$$

with

$$\boldsymbol{\mathcal{S}} = \mathrm{diag}(\sigma_{v,1}^2 \boldsymbol{R}_{u,1}, \ldots, \sigma_{v,N}^2 \boldsymbol{R}_{u,N}) \tag{4.31}$$

To evaluate $\mathbb{E}\{\boldsymbol{\mathcal{K}}_i\}$, we consider $\mathbb{E}\{\boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i}\}$ because the matrix $\boldsymbol{\mathcal{C}}$ is constant. Its (k,ℓ)-th block is given by:

$$\mathbb{E}\{[\boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i}]_{k\ell}\} = \mu_k \mu_\ell\, \mathbb{E}\{\boldsymbol{H}_{k,i} [\boldsymbol{\Sigma}]_{k\ell} \boldsymbol{H}_{\ell,i}\} \tag{4.32}$$

since $\boldsymbol{\mathcal{H}}_{p,i}$ is a block diagonal matrix, see (4.14). In this expression, $[\cdot]_{k\ell}$ denotes the (k,ℓ)-th block of its matrix argument. We start by expanding the matrix product:

$$\mathbb{E}\{(\boldsymbol{H}_{k,i} [\boldsymbol{\Sigma}]_{k\ell} \boldsymbol{H}_{\ell,i})_{pq}\} = \sum_{m=1}^{L} \sum_{n=1}^{L} ([\boldsymbol{\Sigma}]_{k\ell})_{mn}\, \mathbb{E}\{(\boldsymbol{H}_{k,i})_{pm} (\boldsymbol{H}_{\ell,i})_{nq}\} \tag{4.33}$$

Let us now evaluate the expectation on the right-hand side. We have to consider the two cases $(k \neq \ell)$ and $(k = \ell)$ separately. If $k \neq \ell$, we obtain from (4.8):

$$\mathbb{E}\{[\boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i}]_{k\ell}\} = \mu_k \mu_\ell\, M_x^2\, \sigma_x^4\, [\boldsymbol{\Sigma}]_{k\ell} \tag{4.34}$$

If $k = \ell$, we obtain from (4.9):

$$\mathbb{E}\{[\boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i}]_{kk}\} = \mu_k^2\, M_x\, \sigma_x^4 \Big((M_x + 1)[\boldsymbol{\Sigma}]_{kk} + \mathrm{trace}([\boldsymbol{\Sigma}]_{kk}) \boldsymbol{I}_M \Big) \tag{4.35}$$

We denote $\mathbb{E}\{\boldsymbol{\mathcal{K}}_i\}$ by $\boldsymbol{\mathcal{K}}$. The (k,ℓ)-th block of the argument of the trace operator in (4.30) reduces to:

$$[\boldsymbol{\mathcal{K}} \boldsymbol{\mathcal{S}}]_{k\ell} = \sigma_{v,\ell}^2 [\boldsymbol{\mathcal{K}}]_{k\ell} \boldsymbol{R}_{u,\ell} \tag{4.36}$$

since $\boldsymbol{\mathcal{S}}$ is a block diagonal matrix. We conclude that the last expectation in the right-hand side of (4.26) is given by:

$$\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{K}} \boldsymbol{s}_i\} = \sum_{k,\ell,m=1}^{N} c_{mk}\, c_{m\ell}\, \sigma_{v,m}^2\, \mathrm{trace}\big(\mathbb{E}\{[\boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i}]_{k\ell}\}\boldsymbol{R}_{u,m}\big)$$

where the expectation is given by (4.34)–(4.35). This expression can be simplified making further assumptions. For example, if the matrix $\boldsymbol{\Sigma}$ is block diagonal, it becomes:

$$\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{K}} \boldsymbol{s}_i\} = \sum_{k,\ell=1}^{N} c_{k\ell}^2\, \sigma_{v,k}^2\, \mathrm{trace}\big(\mathbb{E}\{[\boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i}]_{\ell\ell}\}\boldsymbol{R}_{u,k}\big) \tag{4.37}$$

where the expectation is given by (4.35). With regards to the first expectation on the right-hand side of (4.26), we have:

$$\mathbb{E}(\widetilde{\boldsymbol{w}}_{i-1}^\top \boldsymbol{\mathcal{B}}_{p,i}^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_{p,i} \widetilde{\boldsymbol{w}}_{i-1}) = \mathbb{E}\|\widetilde{\boldsymbol{w}}_{i-1}\|_{\boldsymbol{\Sigma}'}^2 \tag{4.38}$$

where we introduced the weighting matrix

$$\begin{aligned}\boldsymbol{\Sigma}' &= \mathbb{E}(\boldsymbol{\mathcal{B}}_{p,i}^\top \boldsymbol{\Sigma}\, \boldsymbol{\mathcal{B}}_{p,i}) \\ &= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{H}}_p\boldsymbol{\mathcal{R}} - \boldsymbol{\mathcal{R}}^\top \boldsymbol{\mathcal{H}}_p^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma} + \mathcal{O}(\boldsymbol{\mathcal{M}}^2)\end{aligned} \tag{4.39}$$

where

$$\mathcal{O}(\boldsymbol{\mathcal{M}}^2) = \mathbb{E}\{\boldsymbol{\mathcal{R}}_{H,i}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{R}}_{H,i}\} \tag{4.40}$$

The above expectation depends on higher order moments of the regression data, which makes its calculation complicated. Following [Sayed, 2013a], we focus on the case of sufficiently small step sizes $\{\mu_k\}$ where the effect of terms involving higher powers of the step-sizes can be ignored. A reasonable approximation for $\mathcal{O}(\boldsymbol{\mathcal{M}}^2)$ for sufficiently small step sizes is:

$$\mathcal{O}(\boldsymbol{\mathcal{M}}^2) = \boldsymbol{\mathcal{R}}^\top \mathbb{E}\{\boldsymbol{\mathcal{H}}_{p,i}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\, \boldsymbol{\mathcal{H}}_{p,i}\}\boldsymbol{\mathcal{R}} \tag{4.41}$$

where the expectation on the right-hand side was calculated earlier in (4.32)–(4.35).

Following a similar reasoning as in the previous chapter, we combine the results (4.37) and (4.39) and applying the vec(.) operator we can extract several information about the network performance such as the mean square-error of a single agent or the whole network by setting the proper metric $\boldsymbol{\Sigma}$.

4.4 Privacy preserving diffusion LMS numerical analysis

We shall now carry out a numerical analysis of the algorithm. We commence the study by testing the accuracy of the theoretical model. We considered a connected network consisting of $N = 10$ nodes. The optimal parameter vector $\boldsymbol{w}^o$ of length $L = 5$ was randomly selected from a zero-mean Gaussian distribution with covariance $\boldsymbol{I}_5$. The regression inputs $\boldsymbol{u}_{k,i}$ were zero-mean random

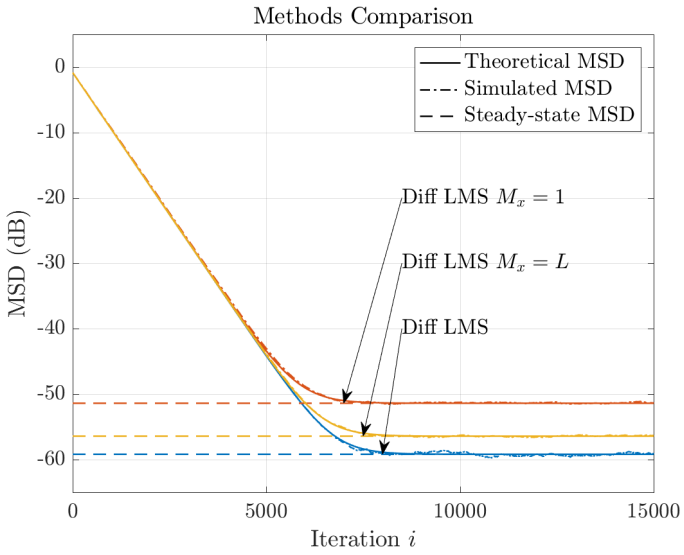

FIGURE 4.1: Mean square deviation (MSD) comparison between *diffusion LMS* and its privacy-preserving version for i.i.d. regression data. We consider a network of $N = 10$ agents, data dimension $L = 5$ and a randomly drawn objective vector $\boldsymbol{w}^o \sim \mathcal{N}(0, \boldsymbol{I}_5)$. The regression data vectors and the additive noise are also draw from a Gaussian distribution $\boldsymbol{u}_{k,i} \sim \mathcal{N}(0, \boldsymbol{I}_5)$, $v_k(i) \sim \mathcal{N}(0, 10^{-3})$. Finally the step-size parameters are set to $\mu_k = \frac{1}{M_x}$.

vectors drawn from a Gaussian distribution with covariance $\boldsymbol{R}_{u,k} = \boldsymbol{I}_5$ in a first experiment where we considered independent entries for the regression vectors $\boldsymbol{u}_{k,i}$, and

$$\boldsymbol{R}_{u,k} = \begin{pmatrix} 1 & a & a^2 & a^3 & a^4 \\ a & 1 & a & a^2 & a^3 \\ a^2 & a & 1 & a & a^2 \\ a^3 & a^2 & a & 1 & a \\ a^4 & a^3 & a^2 & a & 1 \end{pmatrix} \tag{4.42}$$

in a second experiment with $a = 0.3$ where the entries are correlated. The background noises $v_k(i)$ were i.i.d. zero-mean Gaussian random variables of variance $\sigma^2_{v,k} = 10^{-3}$, independent of any other signals. The matrix $\boldsymbol{C}$ was generated using the Metropolis rule [Sayed, 2013a]. The combination matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ were set to the identity for all the algorithms. The step-sizes were set to $\mu_k = 5 \cdot 10^{-3}$. We set the parameter $\sigma_x^2 = \sqrt{1/M_x}$ so as to keep the same convergence rate for all methods. P

The simulation results were obtained by averaging 100 Monte-Carlo runs. It can be observed in Figure 4.1 that the models accurately fit the simulated results for i.i.d. regression data.

In Figure 4.1, we compared the privacy-preserving diffusion LMS in the case of a singular $\boldsymbol{H}_{\ell,i}$ where $M_x = 1$ a full-rank $\boldsymbol{H}_{\ell,i}$ with $M_x = 5$, with the diffusion LMS algorithm. As expected, the diffusion LMS algorithm outperformed its privacy-preserving counterparts. Such result is due to the corrupted nature of the shared data.

To study the influence of M_x on the performance of the *privacy-preserving diffusion LMS*, we

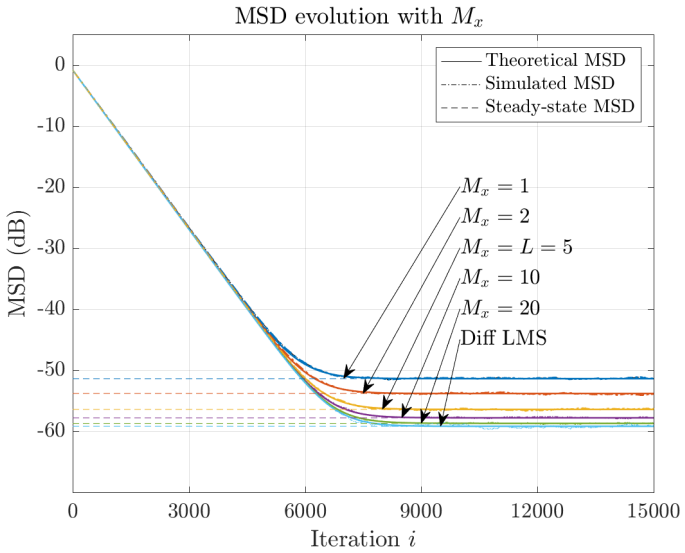

Figure 4.2: Mean square deviation (MSD) comparison between *diffusion LMS* and its privacy-preserving version for correlated input data. For this final experiment, we use the same parameters as the first one with different values of the parameter M_x.

simulated different algorithm runs with different values of the said parameter. Figure 4.2 shows that the performance increases with M_x. Note that as M_x increases, the transformation matrices $\boldsymbol{H}_{\ell,i}$ converge to $\boldsymbol{I}_L$ and the algorithm degenerates to diffusion LMS and loose its privacy-preserving property. On the contrary, small parameter values $M_x < L$ lead to low-rank transformations that ensure privacy. Finally, in Figure 4.3, we report the performance obtained for correlated input data, and confirms the models accuracy.

4.5 Conclusion

In this chapter we focus our attention towards a very sensitive aspect of distributed algorithms. We are referring to the privacy aspect. As a first step towards a privacy preserving diffusion algorithm, we took inspiration from differential privacy to build a distributed algorithm where agents are not only interested in cooperating to enhance their performances but also require data privacy. Then we theoretically analysed the proposed algorithm and verified the fitness of the models numerically. We concluded that, a trade-off between a privacy and performance is necessary as the more secure the algorithm the less accurate.

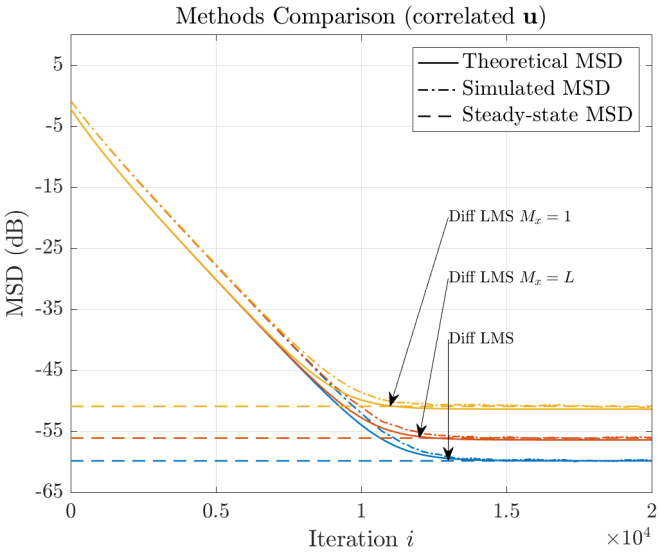

Figure 4.3: Mean square deviation (MSD) comparison between *diffusion LMS* and its privacy-preserving version for correlated input data. We use a similar setting as the previous experiment. However, we consider the covariance matrix $\boldsymbol{R}_{u,k}$ defined in (4.42) for the regression data.

5 Unsupervised Clustered Multitask Learning

Contents

In the previous chapters, we mainly focused on the single task diffusion LMS where all agents are interested in estimating a common parameter. However, agents within the same network often seek to estimate different parameters. Such setting is known as multitask learning. In the present chapter, we start by a brief overview of multitask learning literature. Then we formulate the multitask optimisation problem. Next, we introduce a clustered multitask estimation algorithm. It relies on two types of agents, cluster agents seeking to estimate the clusters centroids as well as the graph structure, and regular agents estimating their respective objective vectors. We then, theoretically analyse the algorithm performance in both mean and mean square sense after the clustering convergence. We also study it under two different scenarios. First, we consider the algorithm under optimal conditions where the tasks are correctly clustered. Under the second scenario, we consider a less favourable setting where some agents are miss-clustered. Such scenario could occur, for instance, when the number of tasks is unknown. For this particular scenario, the number of cluster agents would not match the quantity of tasks to estimate. Such setting will result

in a subset of agents communicating with the wrong cluster agent. Finally, we perform numerical analysis to verify the accuracy of the theoretical model under the previously discussed scenarios.

5.1 Introduction

We previously, mainly focused on the single task diffusion LMS [Sayed, 2003, Sayed, 2008, Sayed, 2013a, Sayed, 2013a]. In this scenario, the whole network aims to minimise an aggregate cost function. Every agent, through local computation and communication, seeks to estimate the minimiser of its cost function. However, in many cases, agents belonging to the same network are interested in estimating different parameters. Such setting is common in mobile networks where each agent seeks to estimate the power of the surrounding signals to determine the optimal base station to link to. Target tracking, is another example of agents seeking different minimisers.

This chapter is structured in four parts. We start by an overview of the multitask learning literature. Then, in the second part, we recall the optimisation problem of multitask diffusion LMS algorithm. Next, we propose an unsupervised clustered multitask algorithm. The information exchange between agents is carried out through a regularisation parameter promoting a clustering of agents. In the third part, we theoretically analyse the algorithm stochastic behaviour in the mean and mean square sense under the previously introduced scenarios. Finally, we carry out a numerical analysis to validate the theoretical analysis and measure its performance.

5.2 Multitask Learning

In this section, we start by giving a brief examination of multitask learning literature. Then, we discuss its extension to a multi-agent setting and its different variants. However, before delving into the literature, it is important to introduce some basic notions. A multitask network composed of N agents seeks to minimise N cost functions. In order to enhance the performance, agents sharing similar objectives should collaborate, form a cluster and limit collaboration with other agents. Depending on the setting, agents either do or do not have access to information about their respective clusters. In the latter case the nodes need to learn their clusters to guarantee an unbiased solution.

Multitask learning in machine learning Multitask learning aims to learn multiple classifiers simultaneously. Such procedure takes advantage of the shared information between classifiers to improve the learning process as argued in [Caruana, 1997]. In a multitask setting, in contrast with single task learning, where the input signals are task specific, they are available for all the tasks. As a consequence, using a shared representation of those signals, related tasks are jointly learned. In [Caruana, 1997], the authors consider a neural network learning to solve multiple tasks simultaneously. The information is shared through a hidden layer in the network connecting all of the tasks outputs. They have also considered multitask K-nearest neighbours and kernel regression. For all considered learning methods, multitask learning outperformed its single task counterpart.

While [Caruana, 1997] used a common representation to share data between tasks, other authors [Evgeniou and Pontil, 2004a, Argyriou et al., 2008] proposed a regularisation approach. They considered Q tasks, for each task q there are M input/output examples $(\boldsymbol{x}_{q1}, y_{t1}), \ldots, (\boldsymbol{x}_{qM}, y_{tM}) \in \mathbb{R}^d \times \mathbb{R}$. They used a regularisation cost function defined as:

$$\min_{\boldsymbol{W}} \sum_{m=1}^{Q} \sum_{n=1}^{M} J(y_{mn}, \boldsymbol{w}_m^\top \boldsymbol{x}_{mn}) + \gamma \Omega(\boldsymbol{W}) \tag{5.1}$$

where $J(.)$ is a loss function, $\Omega(.)$ is a regularisation function, $\boldsymbol{w}_m$ are the estimated vectors and $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_Q\}$. In [Evgeniou and Pontil, 2004a], the authors considered the regularisation function defined as:

$$\Omega(\boldsymbol{W}) = \sum_{m=1}^{Q} ||\boldsymbol{w}_m - \tfrac{1}{Q} \textstyle\sum_{n=1}^{Q} \boldsymbol{w}_n||^2 \tag{5.2}$$

Such regularisation function promotes similarity between tasks by minimising their variances. The authors [Argyriou et al., 2008] used a similar method. However, they used the regularisation as a mean to perform a joint feature selection through a Group Lasso approach [Yuan and Lin, 2006]. The authors considered the following regularised cost function:

$$\min_{\boldsymbol{W}} \sum_{m=1}^{Q} \sum_{n=1}^{M} J(y_{mn}, \boldsymbol{w}_m^\top \boldsymbol{x}_{mn}) + \gamma ||\boldsymbol{W}||_{\ell_1, \ell_2} \tag{5.3}$$

where $\boldsymbol{w}_m$ is the m-th column of the matrix $\boldsymbol{W}$. The (ℓ_1, ℓ_2)-norm is obtained by first calculating the 2-norm of the rows of the matrix $\boldsymbol{W}$ then the 1-norm of the resulting vector such as $||\boldsymbol{W}||_{\ell_1,\ell_2} = \sum_{k}^{d} ||[\boldsymbol{W}]_k||_2$, where $[\boldsymbol{W}]_k$ denotes the k-th row of the matrix $\boldsymbol{W}$. The (ℓ_1, ℓ_2) norm combines the tasks and ensures the joint feature selection while promoting sparsity. In [Rakotomamonjy et al., 2011], the authors investigated a larger class of mixed-norm penalty based on ℓ_p - ℓ_q norm with $p \leq 1$ and $1 \leq q \leq 2$ for linear and non-linear multitask learning. It is worth mentioning that, when $p \leq 1$, the optimisation problem is no longer convex. In order to overcome this difficulty, the authors fitted the problem into the Majorization - Minimisation framework. The authors also provided numerical results proving the superiority of the ℓ_p - ℓ_q norm compared to ℓ_1 - ℓ_2 norm.

Still through regularisation, the authors in [Flamary et al., 2014] considered learning a graph of tasks relation $\boldsymbol{P}$. They considered the following regularisation term:

$$\Omega(\boldsymbol{W}, \{\lambda_q\}, \boldsymbol{P}) = \sum_{q=1}^{Q} \lambda_q ||\boldsymbol{w}_q||_2^2 + \sum_{q,r=1}^{Q} p_{qr} ||\boldsymbol{w}_q - \boldsymbol{w}_r||_2^2 \tag{5.4}$$

where Q is the number of tasks and p_{qr} is the (q, r)-th entry of the adjacency matrix $\boldsymbol{P}$. The first term is a ridge regularisation term and the second promotes pairwise similarity between tasks as dictated by the symmetric adjacency matrix $\boldsymbol{P}$. The authors proposed to learn the adjacency matrix of the task relation graph $\boldsymbol{P}$ alongside the task decision function parameters. They introduced a bilevel optimisation problem dealing with the generalisation error and the optimisation of the task parameters. They have also provided a numerical analysis proving the efficiency of the proposed

approach.

To solve a similar problem, in other words, learning the relationship between tasks as well as the tasks decision functions, authors in [Jacob et al., 2009] considered a convex relaxation of the objective function defined as:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{d\times Q},\boldsymbol{\Sigma}\in\mathcal{S}_r} J(\boldsymbol{W}) + \lambda \text{trace}(\boldsymbol{W}\boldsymbol{\Sigma}^{-1}\boldsymbol{W}^\top) \tag{5.5}$$

where Q is the number of tasks, d the data dimension, $\boldsymbol{W}$ a $d \times Q$ matrix formed by Q vectors to be estimated, $\boldsymbol{\Sigma}$ encodes the graph structure and $\mathcal{S}_r$ is a finite set of semi-definite matrices. The positive coefficient λ controls the trade-off between the two terms. In contrast with the other approaches, the authors considered tasks clustered into r groups and the weight vectors $\boldsymbol{w}$ within the same group are similar.

Multitask diffusion LMS Multitask learning has been extended to a multi-agent setting in [Chen et al., 2015b]. The authors analysed diffusion LMS algorithm when the single task hypothesis is violated. They also proposed a clustering method through adaptive combination weights. In a similar approach as [Evgeniou and Pontil, 2004b, Flamary et al., 2014], the authors in [Chen et al., 2014b], used a regularisation function that measures the similarity between tasks formulated for an agent j as:

$$\Omega_j(\boldsymbol{W}) = \sum_{k\in\mathcal{C}_j} \sum_{\ell\in\mathcal{N}_k\backslash\mathcal{C}_j} \rho_{k\ell} ||\boldsymbol{w}_{\mathcal{C}_{(k)}} - \boldsymbol{w}_{\mathcal{C}_{(\ell)}}||_2^2 \tag{5.6}$$

where $\mathcal{C}_j$ denotes the cluster j, $\mathcal{C}(k)$ stands for the cluster to which node k belongs, and the notation $\mathcal{N}_k\backslash\mathcal{C}_k$ denotes the set of neighbouring nodes of k that are not in the same cluster as k. The non-negative coefficients $\rho_{k\ell}$ locally control the strength of the regularisation. They are similar to the adjacency matrix $\boldsymbol{P}$ in [Flamary et al., 2014]. However, the coefficients $\rho_{k\ell}$ are not learned but set dynamically following a weighting rule depending on the cluster size. Note that this regularisation function still promotes similarity between clusters. Similarly to [Evgeniou et al., 2005], although in a multi-agent setting, authors in [Chen et al., 2014a] considered estimating $\boldsymbol{w}_k^o$ in two parts: $\boldsymbol{u}$ and ϵ_k^o:

$$\boldsymbol{w}_k^o = \boldsymbol{\Theta}\boldsymbol{u} + \epsilon_k^o \tag{5.7}$$

where the part $\boldsymbol{\Theta}\boldsymbol{u}$ is common for all agents, ϵ_k^o is node specific and the matrix $\boldsymbol{\Theta}$ is assumed to be known and full rank. The authors also considered using an approximation of the mean square deviation between agents in the regularisation function in [Chen et al., 2015a].

In [Bertrand and Moonen, 2010a, Bertrand and Moonen, 2010b], the authors studied the performance of a fully connected sensor network. Agents have access to multi-channel observations and the objective vectors are assumed to share a common latent signal subspace. The authors extended the analysis to a tree network topology in [Bertrand and Moonen, 2011] and to heterogeneous and mixed-topology wireless networks in [Szurley et al., 2015].

Some authors addressed the multitask problem in a different way. Instead of using regularisation for conditioning the objective vectors, they proposed to add a clustering step before performing the

diffusion LMS algorithm based on the estimated clusters as it is proposed in [Plata-Chaves et al., 2016]

Regularized Multitask Diffusion In the previously mentioned works, regularisation was mainly used to promote similarity between tasks. However, regularisation can be used for other purposes. For instance, in [Nassif et al., 2015] ℓ_1-norm regularisation was used to promote similarity cluster-wise. They considered two regularisation function defined as:

$$\Omega(\boldsymbol{W}) = \sum_{k=1}^{N} \sum_{\ell \in \mathcal{N}_k \setminus \mathcal{C}_k} (\rho_{k\ell} + \rho_{\ell k}) ||\boldsymbol{w}_{\mathcal{C}(k)} - \boldsymbol{w}_{\mathcal{C}(\ell)}||_1 \tag{5.8}$$

$$\Omega_\alpha(\boldsymbol{W}) = \sum_{k=1}^{N} \sum_{\ell \in \mathcal{N}_k \setminus \mathcal{C}_k} (\rho_{k\ell} + \rho_{\ell k}) \sum_{m=1}^{L} \alpha_m |[\boldsymbol{w}_{\mathcal{C}(k)} - \boldsymbol{w}_{\mathcal{C}(\ell)}]_m| \tag{5.9}$$

where $\mathcal{C}(k)$ denotes the cluster to which node k belongs, $\rho_{\ell k}$ are non-negative coefficients used to locally adjusting the regularisation strength, L is the data dimension and α_m are positive weights to be dynamically adjusted. Note that the second regularisation function $\Omega_\alpha(.)$ is a weighted formulation of the ℓ_1-norm designed to enhance the penalization of the non-zero components of its parameter vector.

In [Nassif et al., 2016d], the authors considered minimisers sharing a large number of similar entries. They introduced the multitask diffusion LMS with Forward-Backward splitting where they also used an ℓ_1-norm regularisation.

Multitask diffusion through correlation Often, in a multitask setting, the tasks are similar or share common parameters. Some authors proposed to use such correlation to build more suited algorithms. For instance, in [Plata-Chaves et al., 2015], the authors considered the parameters to be optimised as three parts: a common part to all the agents in the network, a shared part for a subset of agents and a local part. Some authors proposed to use the inherent physical correlation between tasks as they used the spatiotemporal correlation between the measurements of the nodes [Abdolee et al., 2014]. In [Nassif et al., 2016d] the authors considered minimiser vectors sharing entries without any prior knowledge, and in [Nassif et al., 2016b, Nassif et al., 2017b] they considered linearly related tasks.

5.3 Unsupervised Clustered Multitask estimation

Let us consider an interconnected network of N agents. The objective is to estimate N parameters of interest. We suppose that some agents share similar tasks. These agents ought to form a cluster to enhance the estimation performance. In this work, we consider $Q < N$ clusters. At each time instant i every agent k collects zero mean quantities: the scalar reference signal $d_k(i)$ and the L dimensioned regression vector $\boldsymbol{u}_{k,i}$ with a covariance matrix $\boldsymbol{R}_{u_k} \succ 0$. The data at an agent k are

Table 5.1: List of the symbols and notations used in chapter 5

Symbol	Definition
L	length of the parameter vectors
N	network agent count
Q	number of tasks
$\mathcal{N}_k$	neighbourhood of the agent k including itself
$\boldsymbol{w}_{k,i}$	instantaneous estimate at the agent k
$\bar{\boldsymbol{w}}_{q,i}$	instantaneous estimate of the cluster agent q
$\boldsymbol{w}^o$	optimum parameter vector
$d_k(i)$	reference signal for the agent k at the time instant i
$\boldsymbol{u}_{k,i}$	regression vector of the agent k
$v_k(i)$	additive noise at the agent k

assumed to be related to the unknown parameter vector $\boldsymbol{w}_k^o$ with a linear model:

$$d_k(i) = \boldsymbol{u}_{k,i}^\top \boldsymbol{w}_k^o + v_k(i) \tag{5.10}$$

where $v_k(i)$ a zero mean additive noise with variance $\sigma_{v_k}^2$. Each agent seeks to minimise the local objective cost function J_k:

$$J_k(\boldsymbol{w}) = E\{|d_k(i) - \boldsymbol{u}_{k,i}^\top \boldsymbol{w}|^2\} \tag{5.11}$$

We further assume that some agents share the same tasks i.e. cluster. As a consequence, the N agents form $Q < N$ groups.

Optimisation problem We propose to solve the following optimisation problem in a distributed manner:

$$\min_{\boldsymbol{w}_k, \bar{\boldsymbol{w}}_q, \boldsymbol{\alpha} \in \Delta_Q^N} \sum_k^N \mathbb{E}\{J_k(d_k - \boldsymbol{u}_k^\top \boldsymbol{w}_k)\} + \gamma \sum_k^N \sum_q^Q \alpha_{kq} ||\boldsymbol{w}_k - \bar{\boldsymbol{w}}_q||_2^2 \tag{5.12}$$

where Δ_Q^N is a simplex defined as:

$$\Delta_Q^N = \{\alpha_{k\ell} \geq 0, \sum_\ell^Q \alpha_{k\ell} = 1, \forall k = 1, \ldots, N, \forall \ell = 1, \ldots, Q\} \tag{5.13}$$

The first term is a data related term where $J_k(.)$ is a convex cost function. In the scope of this work, we consider the mean square criterion. The second term is the regularisation term. It is controlled through the non-negative parameter γ. The regularisation term promotes similarity between tasks which results in a clustering effect per task. Each cluster q has a cluster node tasked with estimating two parameters: $\bar{\boldsymbol{w}}_q$ the cluster centroid and α_{kq} positive weighting coefficients encoding the graph structure (cluster membership of each agent).

In the following we consider two types of agents depicted in Figure 5.1. The cluster agents depicted by filled dots in Figure 5.1. They are in charge of the second term of the cost function (5.12). They estimate their respective centroids $\bar{\boldsymbol{w}}_q$ and the relationship between them and the regular agents. This relationship is encoded in the $N \times Q$ matrix $\boldsymbol{\alpha}$. The centroids $\bar{\boldsymbol{w}}_q$ and $\boldsymbol{\alpha}$ are then relayed

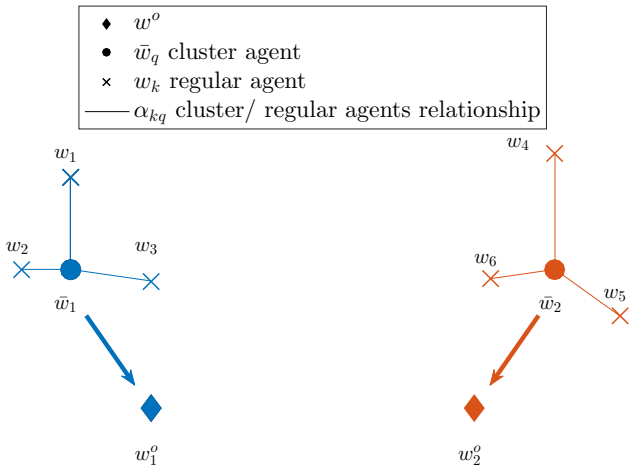

FIGURE 5.1: We depict a network with $N = 6$ agents, $Q = 2$ task agents. The 6 regular agents estimate their local estimates $\boldsymbol{w}_k$ and cluster around the 2 cluster agents. The relationship between the cluster agents and regular agents is conveyed through $\boldsymbol{\alpha}$. The cluster centroids $\bar{\boldsymbol{w}}_q$ and $\boldsymbol{\alpha}$ are estimated by the cluster agents.

to the regular agents. These regular agents depicted by the symbols "$\times$" in Figure 5.1, are in charge of the first term of the cost function (5.12). In other words: they estimate their respective objective vectors using the information relayed by the cluster agents. Through this regularisation, regular agents sharing the same tasks, cluster around their respective centroids. The group formed by cluster and regular agents then converges to the minimiser.

In this work, we simultaneously estimate the objective vectors $\boldsymbol{w}^o$ and the centroids $\bar{\boldsymbol{w}}_q$. It is worth mentioning that this problem is similar to [Flamary et al., 2014] in a distributed setting. However instead of learning the relationship graph between all of the tasks resulting in a $N \times N$ adjacency matrix, we only learn the relationships between each Q cluster agent and the N regular agents. As a result, we obtain a smaller $Q \times N$ matrix.

Figure 5.2 represents the relationships between the regular agents and cluster agents. Starting from the left hand-side, we depict the evolution of the clusters along the algorithm iterations. At the beginning, the coefficients α_{kq} are uniformly selected as we wish that the cluster agents communicate with all of the regular ones. Note the link width is used to represent the coefficients α_{kq}. In the centre, we represent an intermediate step where some links have been severed and the coefficients α_{kq} have been adjusted. For example, the links between the three top agents and the red cluster agent are stronger compared to the green one. Finally, the far-right figure is a representation of the final clusters where all of the regular agents only communicate with one cluster agent. Note that after the convergence of the weights α_{kq} or in the case where $Q = 1$ the problem is equivalent to [Evgeniou and Pontil, 2004a] where they minimise the variance of the estimates per cluster.

Observe that the cost function (5.12) is not convex. However, for a fixed $\boldsymbol{\alpha}$, the function becomes convex with respect to $\bar{\boldsymbol{w}}_q$ or $\boldsymbol{w}_k$. The cost function is also convex with respect to $\boldsymbol{\alpha}$ when both $\boldsymbol{w}_k$ and $\boldsymbol{w}_k$ are fixed. In addition, as the weights α_{kq} are required to belong to the simplex Δ_Q^N, the proposed optimisation problem is constrained with respect to $\boldsymbol{\alpha}$.

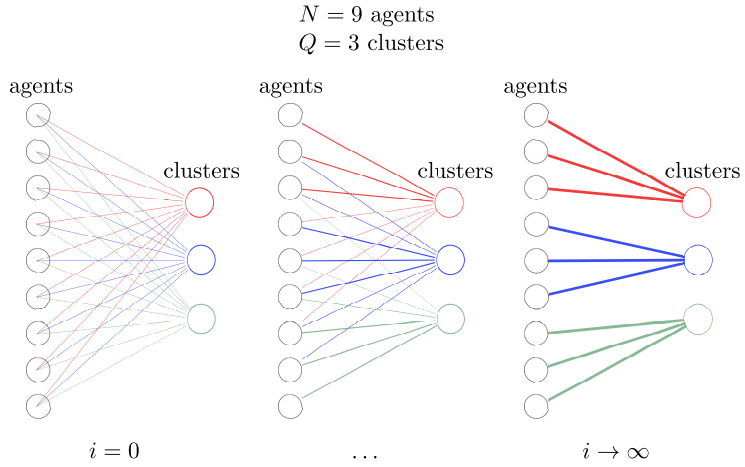

FIGURE 5.2: A representation of the relationships between each of the $N = 9$ agents and the Q cluster agents. We represent the coefficients α_{kq} through the width of the links. On the left-hand side, we illustrate the links as in the beginning of the learning. At this stage all the agents communicate with all of the cluster nodes, thus the thinness of the links. The centre figure represents an intermediate step where the cluster agents start to adjust their connections with the regular agents. For instance, the link between the top regular agent and green cluster agent was severed as they do not share similar tasks. On the right-hand side, we represent the links after the convergence of the adjacency parameters α_{kq} where all the unnecessary links have been cut off. Note that at this stage, the entries of $\boldsymbol{\alpha}_i$ are on the corners of the simplex Δ_Q^N.

Optimisation algorithm To solve the optimisation problem (5.12), we propose a distributed stochastic approach. We consider a network of N regular agents and Q cluster agents. The proposed algorithm run is described in 5.1. In the following we detail its run

- During the first step the Q cluster agents stochastically estimate their respective objective vectors $\bar{\boldsymbol{w}}_{q,i}$ as described in the following recursion:

$$\bar{\boldsymbol{w}}_{q,i} = \bar{\boldsymbol{w}}_{q,i-1} - \gamma\bar{\mu}\sum_{k=1}^{N}\alpha_{kq}(i)(\bar{\boldsymbol{w}}_{q,i-1} - \boldsymbol{w}_{k,i-1}) \tag{5.14}$$

  where $\bar{\mu}$ is a non-negative step-size parameter.

- Based on those estimates, they estimate the coefficients $\alpha_{kq}(i)$ where

$$\widetilde{\alpha}_{kq}(i) = \alpha_{kq}(i-1) - \gamma\mu_\alpha||\bar{\boldsymbol{w}}_{q,i-1} - \boldsymbol{w}_{k,i-1}||_2^2 \tag{5.15}$$

  where μ_α is a non-negative adaptation step-size parameter. Each row $\widetilde{\boldsymbol{\alpha}}_{k,i}$ of the temporary matrix $\widetilde{\boldsymbol{\alpha}}_i$ is then projected on the simplex Δ_Q^N

$$\boldsymbol{\alpha}_{k,i} = \mathrm{P}_{\Delta_Q^N}(\widetilde{\boldsymbol{\alpha}}_{k,i}) \tag{5.16}$$

  where $\mathrm{P}_{\Delta_Q^N}(.)$ is the simplex projection operator.

Algorithm 5.1 Unsupervised Clustered Multitask learning algorithm

initialise $\boldsymbol{w}_i$, $\bar{\boldsymbol{w}}_i$, $\boldsymbol{\alpha}$,
for $i = 1, \dots$ **do**
  for $q = 1, 2, \dots, Q$ **do**
    update $\bar{\boldsymbol{w}}_{q,i}$

$$\bar{\boldsymbol{w}}_{q,i} = \bar{\boldsymbol{w}}_{q,i-1} - \gamma\bar{\mu} \sum_{k=1}^{N} \alpha_{kq}(i)(\bar{\boldsymbol{w}}_{q,i-1} - \boldsymbol{w}_{k,i-1})$$

    update $\widetilde{\alpha}_{kq}(i)$

$$\widetilde{\alpha}_{kq}(i) = \alpha_{kq}(i-1) - \gamma\mu_\alpha ||\bar{\boldsymbol{w}}_{q,i} - \boldsymbol{w}_{k,i-1}||_2^2$$

    project the vector $\widetilde{\boldsymbol{\alpha}}_{k,i}$ onto the simplex Δ^Q as

$$\boldsymbol{\alpha}_{k,i} = \mathrm{P}_{\Delta_Q^N}(\widetilde{\boldsymbol{\alpha}}_{k,i})$$

    send $\bar{\boldsymbol{w}}_q$ and $\boldsymbol{\alpha}_i$ to all agents
  end for
  for $k = 1, 2, \dots, N$ **do**
    update the local estimate $\boldsymbol{w}_{k,i}$

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \left[\boldsymbol{u}_{k,i}(d_k(i) - \boldsymbol{u}_{k,i}^\top \boldsymbol{w}_{k,i-1}) - \gamma \sum_{q=1}^{Q} \alpha_{kq}(i)(\boldsymbol{w}_{k,i-1} - \bar{\boldsymbol{w}}_{q,i}) \right]$$

  end for
end for

- Finally the regular agents estimate their objective vectors $\boldsymbol{w}_{k,i}$ using the following recursion

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu_k \left[\boldsymbol{u}_{k,i}(d_k(i) - \boldsymbol{u}_{k,i}^\top \boldsymbol{w}_{k,i-1}) - \underbrace{\gamma \sum_{q=1}^{Q} \alpha_{kq}(i)(\boldsymbol{w}_{k,i-1} - \bar{\boldsymbol{w}}_{q,i-1})}_{\text{Regularisation term}} \right] \tag{5.17}$$

Observe that the three parameters $\boldsymbol{w}_k$, $\bar{\boldsymbol{w}}_q$ and $\boldsymbol{\alpha}$ are estimated through a stochastic gradient descent. Furthermore, as the estimates updates are carried out over three stages: centroids $\bar{\boldsymbol{w}}_q$, local estimates $\boldsymbol{w}_k$ then $\boldsymbol{\alpha}$, the algorithm structure resembles the block coordinate descent.

Although extensively studied, stochastic gradient descent based algorithms, because of non convexity, their convergence properties for non convex optimisation remains unknown. However, for smooth cost functions, these algorithms are guaranteed to converge to first-order optimal solutions [Liu et al., 2018]. In addition, some authors [Zeng and Yin, 2018, Bianchi and Jakubowicz, 2013] successfully implemented projected stochastic gradient descent for non convex optimisation.

Table 5.2: List of symbols defined throughout the performance analysis chapter 5

Symbol	Equation
$\widetilde{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i}$	(5.23)
$\widehat{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \bar{\boldsymbol{w}}_{k,i}$	(5.24)
$\widetilde{\boldsymbol{w}}_i = \text{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\}$	(5.25)
$\widehat{\boldsymbol{w}}_i = \text{col}\{\widehat{\boldsymbol{w}}_{1,i}, \widehat{\boldsymbol{w}}_{2,i}, \ldots, \widehat{\boldsymbol{w}}_{Q,i}\}$	(5.26)
$\widetilde{\boldsymbol{w}}_{e,i} = \begin{bmatrix} \widetilde{\boldsymbol{w}}_i \\ \widehat{\boldsymbol{w}}_i \end{bmatrix}$	(5.27)
$\boldsymbol{s}_{k,i} = v_k(i)\boldsymbol{u}_{k,i}$	(5.29)
$\boldsymbol{s}_i = \text{col}\{\boldsymbol{s}_{1,i}, \boldsymbol{s}_{2,i}, \ldots, \boldsymbol{s}_{N,i}\}$	(5.32)
$\boldsymbol{S} = \mathbb{E}\{\boldsymbol{s}_i \boldsymbol{s}_i^\top\} = \text{diag}\{\sigma_{v_1}^2 \boldsymbol{R}_{u,1}, \sigma_{v_2}^2 \boldsymbol{R}_{u,2}, \ldots, \sigma_{v_N}^2 \boldsymbol{R}_{u,N}, \boldsymbol{0}_{L\times L}\}$	(5.49)
$\boldsymbol{\mathcal{B}}_{c,i} = \begin{bmatrix} \boldsymbol{I}_{NL} - \gamma\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{u,i} & \gamma\boldsymbol{\mathcal{M}}(\boldsymbol{1}_{N\times 1} \otimes \boldsymbol{I}_L) \\ \bar{\mu}(\boldsymbol{1}_{1\times N} \otimes \boldsymbol{I}_L) & (\boldsymbol{I}_L - \bar{\mu}N) \end{bmatrix}$	(5.37)
$\boldsymbol{\mathcal{B}}_c = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{c,i}\} = \begin{bmatrix} \boldsymbol{I}_{NL} - \gamma\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{u} & \gamma\boldsymbol{\mathcal{M}}(\boldsymbol{1}_{N\times 1} \otimes \boldsymbol{I}_L) \\ \bar{\mu}(\boldsymbol{1}_{1\times N} \otimes \boldsymbol{I}_L) & (\boldsymbol{I}_L - \bar{\mu}N) \end{bmatrix}$	(5.40)
$\boldsymbol{\Sigma}' = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{c,i}^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_{c,i}\}$	(5.46)
$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{M}}\boldsymbol{S}\boldsymbol{\mathcal{M}}$	(5.50)
$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{B}}_c^\top \otimes \boldsymbol{\mathcal{B}}_c^\top$	(5.57)
$\boldsymbol{x} = \text{col}\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$	(5.62)
$\boldsymbol{x}_k = \boldsymbol{w}_k^o - w^*$	(5.63)
$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{\mathcal{M}}\boldsymbol{x} \\ -\bar{\mu}(\boldsymbol{1}_{1\times N} \otimes \boldsymbol{I}_L)\boldsymbol{x} \end{bmatrix}$	(5.74)

5.4 Theoretical analysis with perfect clustering

In this section we shall analyse the algorithm performance under optimal conditions, in other words, when all the agents are assigned to the correct cluster. However, before proceeding with the analysis, let us introduce the following assumption on the regression data.

Assumption 5.1. *The regression vectors $\boldsymbol{u}_{k,i}$ arise from a zero-mean random process that is temporally white and spatially independent.*

Let us assume that the local estimates have converged to a close enough estimates $\boldsymbol{w}_{k,\infty}$ and $\bar{\boldsymbol{w}}_{q,\infty}$ of their respective objectives $\boldsymbol{w}_k^o$, $\boldsymbol{w}_q^o$ as:

$$\boldsymbol{w}_{k,i} \to \boldsymbol{w}_{k,\infty} \tag{5.18}$$

$$\bar{\boldsymbol{w}}_{q,i} \to \bar{\boldsymbol{w}}_{q,\infty} \tag{5.19}$$

and

$$||\bar{\boldsymbol{w}}_{q,\infty} - \boldsymbol{w}_{k,\infty}||_2^2 < ||\bar{\boldsymbol{w}}_{q,\infty} - \boldsymbol{w}_{\ell,\infty}||_2^2 \;\; \forall q = 1, \ldots, Q;\, k \in \mathcal{C}_q;\, \ell \notin \mathcal{C}_q \tag{5.20}$$

As a result, we have:

$$\min_{\alpha_{kq}\in\Delta_Q^N} \gamma \sum_{k=1}^{N}\sum_{q=1}^{Q} \alpha_{kq}||\boldsymbol{w}_{k,\infty} - \boldsymbol{w}_q||_2^2 \tag{5.21}$$

From (5.21) it is straightforward that

$$\alpha_{kq} = \begin{cases} 1 & \text{if } k \in \mathcal{C}_q \\ 0 & \text{otherwise} \end{cases} \tag{5.22}$$

where $\mathcal{C}_q$ denotes the q-th cluster. Such result means that, after convergence, each cluster behaves independently of the others, thus allowing us to considerably facilitate the algorithm analysis by considering $Q = 1$.

Let us now begin the algorithm analysis by introducing the error vectors

$$\widetilde{\boldsymbol{w}}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i} \tag{5.23}$$

$$\widehat{\boldsymbol{w}}_{q,i} = \boldsymbol{w}^o - \bar{\boldsymbol{w}}_{k,i} \tag{5.24}$$

We also introduce their concatenated versions

$$\widetilde{\boldsymbol{w}}_i = \text{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\} \tag{5.25}$$

$$\widehat{\boldsymbol{w}}_i = \text{col}\{\widehat{\boldsymbol{w}}_{1,i}, \widehat{\boldsymbol{w}}_{2,i}, \ldots, \widehat{\boldsymbol{w}}_{Q,i}\} \tag{5.26}$$

and finally, the whole error vector $\widetilde{\boldsymbol{w}}_{e,i}$ defined as:

$$\widetilde{\boldsymbol{w}}_{e,i} = \begin{bmatrix} \widetilde{\boldsymbol{w}}_i \\ \widehat{\boldsymbol{w}}_i \end{bmatrix} \tag{5.27}$$

From the definition (5.23) we get:

$$\widetilde{\boldsymbol{w}}_{k,i} = \widetilde{\boldsymbol{w}}_{k,i-1} - \mu_k[\boldsymbol{u}_{k,i}\boldsymbol{u}_{k,i}^\top \widetilde{\boldsymbol{w}}_{k,i-1} + \boldsymbol{s}_{k,i} - \gamma \sum_{q=1}^{Q} \alpha_{kq}(i)(\boldsymbol{w}_{k,i-1} - \bar{\boldsymbol{w}}_{q,i-1})] \tag{5.28}$$

where

$$\boldsymbol{s}_{k,i} = v_k(i)\boldsymbol{u}_{k,i} \tag{5.29}$$

Since $Q = 1$ we can write:

$$\widetilde{\boldsymbol{w}}_{k,i} = \widetilde{\boldsymbol{w}}_{k,i-1} - \mu_k[\boldsymbol{R}_{u_k,i}\widetilde{\boldsymbol{w}}_{k,i-1} + \boldsymbol{s}_{k,i} - \gamma(\boldsymbol{w}^o - \widetilde{\boldsymbol{w}}_{k,i-1} - \bar{\boldsymbol{w}}_{i-1})] \tag{5.30}$$

The equation (5.30) can be formulated for the whole network as:

$$\widetilde{\boldsymbol{w}}_i = [\boldsymbol{I}_{NL} - \gamma\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{u,i}]\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{M}}\boldsymbol{s}_i + \gamma\boldsymbol{\mathcal{M}}(\boldsymbol{1}_{N\times 1} \otimes \boldsymbol{I}_L)\widehat{\boldsymbol{w}}_{i-1} \tag{5.31}$$

where the $L \times N$ vector $\boldsymbol{s}_i$ is defined as:

$$\boldsymbol{s}_i = \text{col}\{\boldsymbol{s}_{1,i}, \boldsymbol{s}_{2,i}, \ldots, \boldsymbol{s}_{N,i}\} \tag{5.32}$$

From the definition (5.24) the centroid of the cluster is expressed as:

$$\widehat{\boldsymbol{w}}_i = \widehat{\boldsymbol{w}}_{i-1} + \gamma\bar{\mu} \sum_{\ell=1}^{N} (\bar{\boldsymbol{w}}_{i-1} - \boldsymbol{w}_{\ell,i-1}) \tag{5.33}$$

Using the definition (5.23) we get $\boldsymbol{w}_{\ell,i} = \boldsymbol{w}^o - \widetilde{\boldsymbol{w}}_{\ell,i}$. Substituting this expression leads to:

$$\widehat{\boldsymbol{w}}_i = \widehat{\boldsymbol{w}}_{i-1} + \gamma\bar{\mu} \sum_{\ell=1}^{N} (\underbrace{\bar{\boldsymbol{w}}_{i-1} - \boldsymbol{w}^o}_{-\widehat{\boldsymbol{w}}_{i-1}} + \widetilde{\boldsymbol{w}}_{\ell,i-1}) \tag{5.34}$$

Using the definition (5.23) we find:

$$\widehat{\boldsymbol{w}}_i = (\boldsymbol{I}_L - \gamma\bar{\mu}N)\widehat{\boldsymbol{w}}_{i-1} + \gamma\bar{\mu} \sum_{\ell=1}^{N} \widetilde{\boldsymbol{w}}_{\ell,i-1} \tag{5.35}$$

Combining (5.31) and (5.35) we get:

$$\boldsymbol{w}_{e,i} = \boldsymbol{\mathcal{B}}_{c,i}\boldsymbol{w}_{e,i-1} - \begin{bmatrix} \boldsymbol{\mathcal{M}}\boldsymbol{s}_i \\ \boldsymbol{0}_{L\times 1} \end{bmatrix} \tag{5.36}$$

where

$$\boldsymbol{\mathcal{B}}_{c,i} = \begin{bmatrix} \boldsymbol{I}_{NL} - \gamma\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{u,i} & \gamma\boldsymbol{\mathcal{M}}(\boldsymbol{1}_{N\times 1} \otimes \boldsymbol{I}_L) \\ \gamma\bar{\mu}(\boldsymbol{1}_{1\times N} \otimes \boldsymbol{I}_L) & (1 - \gamma\bar{\mu}N)\boldsymbol{I}_L \end{bmatrix} \tag{5.37}$$

5.4.1 Mean error stability analysis

Let us now study the algorithm stability in the mean error sense. From (5.36) we have:

$$\mathbb{E}\{\boldsymbol{w}_{e,i}\} = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{c,i}\boldsymbol{w}_{e,i-1}\} - \begin{bmatrix} \boldsymbol{\mathcal{M}}\mathbb{E}\{\boldsymbol{s}_i\} \\ \boldsymbol{0}_{L\times 1} \end{bmatrix} \tag{5.38}$$

Using the independence 5.1 and since $\mathbb{E}\{\boldsymbol{s}_i\} = 0$, we find:

$$\mathbb{E}\{\boldsymbol{w}_{e,i}\} = \boldsymbol{\mathcal{B}}_c\mathbb{E}\{\boldsymbol{w}_{e,i-1}\} \tag{5.39}$$

where

$$\boldsymbol{\mathcal{B}}_c = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{c,i}\} = \begin{bmatrix} \boldsymbol{I}_{NL} - \gamma\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_u & \gamma\boldsymbol{\mathcal{M}}(\boldsymbol{1}_{N\times 1} \otimes \boldsymbol{I}_L) \\ \gamma\bar{\mu}(\boldsymbol{1}_{1\times N} \otimes \boldsymbol{I}_L) & (1 - \gamma\bar{\mu}N)\boldsymbol{I}_L \end{bmatrix} \tag{5.40}$$

From (5.39), the algorithm asymptotically converges in the mean to $\boldsymbol{w}^o$ if and only if the matrix $\boldsymbol{\mathcal{B}}_c$ is stable. In other words, all of its eigenvalues strictly lie inside the unite disc. We have:

$$\begin{aligned}\rho(\boldsymbol{\mathcal{B}}_c) &\leq ||\boldsymbol{\mathcal{B}}_c||_{b,\infty} \\ &\leq (N+1)\max_{\ell k} ||[\boldsymbol{\mathcal{B}}_c]_{\ell k}||\end{aligned} \tag{5.41}$$

which results in the following condition on the step-size parameters:

$$0 < \mu_k < \tfrac{1}{(N+1)(\gamma+\lambda_{\max}(R_{u_k}))} \tag{5.42}$$

$$0 < \bar{\mu} < \tfrac{1}{\gamma N(N+1)} \tag{5.43}$$

Observe that the regularisation parameter γ does not only control the regularisation strength but also ensures the algorithm stability.

5.4.2 Mean square error analysis

For the sake of generality, we chose to analyse the weighted mean-square deviation $\mathbb{E}||\widetilde{\boldsymbol{w}}_i||^2_{\boldsymbol{\Sigma}}$ with the metric $\boldsymbol{\Sigma}$ being a nonnegative definite $(N+1)L \times (N+1)L$ matrix. Such approach allows different types of studies. For instance, $\boldsymbol{\Sigma}$ can be set to extract the mean-square deviation of the whole network, one single agent or the centroid error vector $\bar{\boldsymbol{w}}_i$. The advantages of this method are not only limited to mean-square deviation, for instance, $\boldsymbol{\Sigma}$ can be set the covariance matrix $\text{diag}\{\boldsymbol{\mathcal{R}}_u, \boldsymbol{0}_{L\times 0}\}$ to extract the excess mean-square deviation (EMSE).

From (5.36) we have:

$$\mathbb{E}||\boldsymbol{w}_{e,i}||^2_{\boldsymbol{\Sigma}} = \mathbb{E}\left\{\left[\boldsymbol{\mathcal{B}}_{c,i}\boldsymbol{w}_{e,i-1}\} - \begin{bmatrix}\boldsymbol{\mathcal{M}}\mathbb{E}\{\boldsymbol{s}_i\} \\ \boldsymbol{0}_{L\times 1}\end{bmatrix}\right]^\top \boldsymbol{\Sigma}\left[\boldsymbol{\mathcal{B}}_{c,i}\boldsymbol{w}_{e,i-1}\} - \begin{bmatrix}\boldsymbol{\mathcal{M}}\mathbb{E}\{\boldsymbol{s}_i\} \\ \boldsymbol{0}_{L\times 1}\end{bmatrix}\right]\right\} \tag{5.44}$$

Using the independence 5.1 we find:

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{\boldsymbol{\Sigma}} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i-1}||^2_{\boldsymbol{\Sigma}'} + \mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i\} \tag{5.45}$$

where

$$\boldsymbol{\Sigma}' = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{c,i}^\top \boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}_{c,i}\} \tag{5.46}$$

Let us start with the last term of the RHS (5.45)

$$\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i\} = \text{trace}(\mathbb{E}\{\boldsymbol{s}_i\boldsymbol{s}_i^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\}) \tag{5.47}$$

using the trace(.) properties we have:

$$\mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i\} = \text{trace}(\boldsymbol{\mathcal{Y}}\boldsymbol{\Sigma}) \tag{5.48}$$

where

$$\boldsymbol{S} = \mathbb{E}\{\boldsymbol{s}_i \boldsymbol{s}_i^\top\} = \text{diag}\{\sigma_{v_1}^2 \boldsymbol{R}_{u,1}, \sigma_{v_2}^2 \boldsymbol{R}_{u,2}, \ldots, \sigma_{v_N}^2 \boldsymbol{R}_{u,N}, \boldsymbol{0}_{L\times L}\} \tag{5.49}$$

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{M}}\boldsymbol{S}\boldsymbol{\mathcal{M}} \tag{5.50}$$

The analysis of the term $\boldsymbol{\Sigma}'$ appearing in the RHS (5.45) is similar to the classical diffusion LMS and can be approximated for small enough step-size by

$$\boldsymbol{\Sigma}' \simeq \boldsymbol{\mathcal{B}}_c^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_c \tag{5.51}$$

Indeed, replacing the matrix $\boldsymbol{\mathcal{B}}_{c,i}$ from (5.37) in (5.46) would result in the same terms plus an additional depending on the squared step-size parameters μ_k. For sufficiently small step-sizes, this additional term will have a negligible effect. In this way, we shall assume that

$$\boldsymbol{\Sigma}' = \boldsymbol{\mathcal{B}}_c^\top \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_c \tag{5.52}$$

Replacing (5.48) in (5.45) leads to

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||_{\boldsymbol{\Sigma}}^2 = \mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i-1}||_{\boldsymbol{\Sigma}'}^2 + \text{trace}(\boldsymbol{\mathcal{Y}}\boldsymbol{\Sigma}) \tag{5.53}$$

Proceeding as in [Sayed, 2013a], we use the vec(.) operator on (5.52)

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma} \tag{5.54}$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ are the vectorized versions of the metric matrices $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}'$, respectively

$$\boldsymbol{\sigma} = \text{vec}(\boldsymbol{\Sigma}) \tag{5.55}$$

$$\boldsymbol{\sigma}' = \text{vec}(\boldsymbol{\Sigma}') \tag{5.56}$$

and $\boldsymbol{\mathcal{F}}$ a $[(N+1)L]^2 \times [(N+1)L]^2$ weighing matrix defined as

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{B}}_c^\top \otimes \boldsymbol{\mathcal{B}}_c^\top \tag{5.57}$$

Replacing (5.54) in (5.45) and applying the vec(.) operator on its last term leads to

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||_{\boldsymbol{\sigma}}^2 = \mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i-1}||_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}^2 + \text{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top \boldsymbol{\sigma} \tag{5.58}$$

The algorithm is stable in the mean-square sense if and only if the matrix $\boldsymbol{\mathcal{F}}$ defined in (5.57) is stable i.e. all of its eigenvalues strictly lie inside the unite disc. Such condition is achieved for

sufficiently small step-sizes μ_k and $\bar{\mu}$ such as:

$$0 < \mu_k < \frac{1}{(N+1)(\gamma+\lambda_{\max}(R_{u_k}))} \tag{5.59}$$

$$0 < \bar{\mu} < \frac{1}{\gamma N(N+1)} \tag{5.60}$$

The equation (5.58) allows a versatile analysis of the algorithm. Depending on the metric $\boldsymbol{\sigma}$, it is possible to extract the mean square error of a single agent the whole network or the cluster estimate $\bar{\boldsymbol{w}}$. In addition, it is also possible to extract other performance measurements such as the excess mean square error (EMSE).

Note that the steady and transient states of the Mean-Square Deviation can be studied following the same procedure as in [Sayed, 2013a] as reported in chapter 2.

5.5 Algorithm performance for imperfect clustering

In this part, we study the algorithm behaviour when some agents are miss-clustered. As a consequence, one or more agents would collaborate with the wrong cluster agents. To model this setting, we shall consider two objective vectors $\boldsymbol{w}_1^o$ and $\boldsymbol{w}_2$. However, we consider a network with $Q = 1$. Under such setting, the network has only one cluster agent. Let us consider the arbitrary vector $\boldsymbol{w}^*$ chosen from $\{\boldsymbol{w}_1^o, \boldsymbol{w}_2^o\}$ and study the estimation error.

We redefine the cluster error vector as:

$$\widehat{\boldsymbol{w}}_i = \boldsymbol{w}^* - \bar{\boldsymbol{w}}_i \tag{5.61}$$

We also introduce the vector $\boldsymbol{x}_k$ and its concatenated version $\boldsymbol{x}$

$$\boldsymbol{x} = \text{col}\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \tag{5.62}$$

$$\boldsymbol{x}_k = \boldsymbol{w}_k^o - \boldsymbol{w}^* \tag{5.63}$$

From the definition (5.23) we have:

$$\widetilde{\boldsymbol{w}}_{k,i} = \underbrace{\boldsymbol{w}_k^o - \boldsymbol{w}_{k,i-1}}_{\widetilde{\boldsymbol{w}}_{k,i-1}} - \mu_k \left[\boldsymbol{u}_{k,i}(d_k(i) - \boldsymbol{u}_{k,i}^\top \boldsymbol{w}_{k,i-1}) - \gamma(\boldsymbol{w}_{k,i-1} - \bar{\boldsymbol{w}}_{i-1})\right] \tag{5.64}$$

Replacing $d_k(i)$ by its definition (5.10) results in:

$$\widetilde{\boldsymbol{w}}_{k,i} = \widetilde{\boldsymbol{w}}_{k,i-1} - \mu_k \left[\boldsymbol{R}_{u_k,i}\widetilde{\boldsymbol{w}}_{k,i-1} + \boldsymbol{s}_k(i) - \gamma(\boldsymbol{w}_{k,i-1} - \bar{\boldsymbol{w}}_{i-1})\right] \tag{5.65}$$

Using the vector error definition (5.23) we find:

$$\widetilde{\boldsymbol{w}}_{k,i} = \widetilde{\boldsymbol{w}}_{k,i-1} - \mu_k \left[\boldsymbol{R}_{u_k,i}\widetilde{\boldsymbol{w}}_{k,i-1} + \boldsymbol{s}_k(i) - \gamma(\underbrace{\boldsymbol{w}_k^o - \boldsymbol{w}^*}_{\boldsymbol{x}_k} - \widetilde{\boldsymbol{w}}_{k,i-1} + \underbrace{\boldsymbol{w}^* - \bar{\boldsymbol{w}}_{i-1}}_{\widehat{\boldsymbol{w}}_i})\right] \tag{5.66}$$

Similarly to the previous analysis, we rewrite the error vector under a compact form

$$\widetilde{\boldsymbol{w}}_i = \underbrace{(I_{NL} - \gamma\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{R}}_{u,i})\,\widetilde{\boldsymbol{w}}_{i-1} - \boldsymbol{\mathcal{M}}\boldsymbol{s}_i + \gamma\boldsymbol{\mathcal{M}}(\mathbf{1}_{N\times 1}\otimes \boldsymbol{I}_L)\widehat{\boldsymbol{w}}_{i-1}}_{\text{weights convergence term}} + \underbrace{\gamma\boldsymbol{\mathcal{M}}\boldsymbol{x}}_{\text{Bias term}} \tag{5.67}$$

In the case where all agents share the same task i.e. $\boldsymbol{w}_k^o = \boldsymbol{w}^* \,\forall\, k$, we retrieve the error term (5.31) where the bias term is equal to zero. It is noteworthy that the bias term depends on the regularisation parameter γ. Such result allows control over the estimation error through the said parameter.

From (5.61) the newly redefined cluster error vector is expressed as:

$$\widehat{\boldsymbol{w}}_i = \boldsymbol{w}^* - \bar{\boldsymbol{w}}_i \tag{5.68}$$

which results in the following expression:

$$\widehat{\boldsymbol{w}}_i = \widehat{\boldsymbol{w}}_{i-1} + \gamma\bar{\mu}\sum_{\ell=1}^{N}(\bar{\boldsymbol{w}}_{i-1} - \boldsymbol{w}_{\ell,i-1}) \tag{5.69}$$

From the definition (5.23) we have $\boldsymbol{w}_{\ell,i} = \boldsymbol{w}^* - \widetilde{\boldsymbol{w}}_{\ell,i}$. Using this expression leads to:

$$\widehat{\boldsymbol{w}}_i = \widehat{\boldsymbol{w}}_{i-1} + \gamma\bar{\mu}\sum_{\ell=1}^{N}(\underbrace{\bar{\boldsymbol{w}}_{i-1} - \boldsymbol{w}^*}_{-\widehat{\boldsymbol{w}}_{i-1}} + \widetilde{\boldsymbol{w}}_{\ell,i-1} + \underbrace{\boldsymbol{w}^* - \boldsymbol{w}_k^o}_{-\boldsymbol{x}_k}) \tag{5.70}$$

Using the definition (5.68) we have:

$$\widehat{\boldsymbol{w}}_i = \underbrace{(\boldsymbol{I}_L - \gamma\bar{\mu}N)\widehat{\boldsymbol{w}}_{i-1} + \gamma\bar{\mu}\sum_{\ell=1}^{N}\widetilde{\boldsymbol{w}}_{\ell,i-1}}_{\text{clustering term}} - \underbrace{\gamma\bar{\mu}\sum_{\ell=1}^{N}\boldsymbol{x}_k}_{\text{Bias term}} \tag{5.71}$$

Compared to (5.35), there is an additional term that only depends on the difference between the shared minimiser $\boldsymbol{w}^*$ and local minimisers $\boldsymbol{w}_k^o$. This term is equal to zero when all agents share the same task.

Combining the results (5.67) and (5.71) we find:

$$\boldsymbol{w}_{e,i} = \underbrace{\boldsymbol{\mathcal{B}}_{c,i}\boldsymbol{w}_{e,i-1}}_{\text{Data term}} - \underbrace{\begin{bmatrix}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i\\ \mathbf{0}_{L\times 1}\end{bmatrix}}_{\text{Noise term}} + \underbrace{\gamma\begin{bmatrix}\boldsymbol{\mathcal{M}}\boldsymbol{x}\\ -\bar{\mu}(\mathbf{1}_{1\times N}\otimes \boldsymbol{I}_L)\boldsymbol{x}\end{bmatrix}}_{\text{bias term}} \tag{5.72}$$

where the matrix $\boldsymbol{\mathcal{B}}_{c,i}$ remains unchanged, as defined in (5.37). As in (5.36), we have a data term and a noise term. However in addition, there is a third term depending the regularisation term γ and the difference between the common minimiser $\boldsymbol{w}^*$ and agents minimisers $\boldsymbol{w}_k^o$. This term is equal to zero if all the agents seek the same objective vector i.e. $\boldsymbol{w}_k^o = \boldsymbol{w}^* \,\forall\, k$.

5.5.1 Mean error analysis

Similarly to the previous theoretical analysis, in this part we study the algorithm in the mean error sense in the case where some agents are miss-clustered. From (5.72) we have:

$$\mathbb{E}\{\boldsymbol{w}_{e,i}\} = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{c,i}\}\mathbb{E}\{\boldsymbol{w}_{e,i-1}\} - \begin{bmatrix}\boldsymbol{\mathcal{M}}\mathbb{E}\{\boldsymbol{s}_i\}\\ \boldsymbol{0}_{L\times 1}\end{bmatrix} + \underbrace{\gamma\mathbb{E}\left\{\begin{bmatrix}\boldsymbol{\mathcal{M}}\boldsymbol{x}\\ -\bar{\mu}(\boldsymbol{1}_{1\times N}\otimes\boldsymbol{I}_L)\boldsymbol{x}\end{bmatrix}\right\}}_{\boldsymbol{z}} \tag{5.73}$$

where

$$\boldsymbol{z} = \begin{bmatrix}\boldsymbol{\mathcal{M}}\boldsymbol{x}\\ -\bar{\mu}(\boldsymbol{1}_{1\times N}\otimes\boldsymbol{I}_L)\boldsymbol{x}\end{bmatrix} \tag{5.74}$$

We have $\mathbb{E}\{\boldsymbol{s}_i\} = 0$ and using the independence 5.1, we find:

$$\mathbb{E}\{\boldsymbol{w}_{e,i}\} = \boldsymbol{\mathcal{B}}_c\,\mathbb{E}\{\boldsymbol{w}_{e,i-1}\} + \gamma\boldsymbol{z} \tag{5.75}$$

where $\boldsymbol{\mathcal{B}}_c$ is defined in (5.40). Note that the convergence condition remains the same, namely:

$$0 < \mu_k < \tfrac{1}{(N+1)(\gamma+\lambda_{\max}(R_{u_k}))} \tag{5.76}$$

$$0 < \bar{\mu} < \tfrac{1}{\gamma N(N+1)} \tag{5.77}$$

5.5.2 Network Mean Performance

Let us study the algorithm mean performance. As the algorithm is stable for sufficiently small step-sizes we can write:

$$\lim_{i\to\infty}\mathbb{E}\{\boldsymbol{w}_{e,i}\} = \lim_{i\to\infty}\boldsymbol{\mathcal{B}}_c\,\mathbb{E}\{\boldsymbol{w}_{e,i-1}\} + \gamma\boldsymbol{z} \tag{5.78}$$

Grouping terms leads to:

$$\lim_{i\to\infty}\mathbb{E}\{\boldsymbol{w}_{e,i}\} = \gamma(\boldsymbol{I}-\boldsymbol{\mathcal{B}}_c)^{-1}\boldsymbol{z} \tag{5.79}$$

From (5.79), the error vector does not converge towards the vector $\boldsymbol{0}_{(N+1)L}$ but towards a bias term proportional to the vector $\boldsymbol{z}$. It is worth mentioning that this error term can be controlled through the regularisation parameter γ. In the case where $\gamma = 0$ we retrieve an unbiased solution, this case corresponds to single agent LMS.

5.5.3 Mean square error analysis

Similarly to the previous mean square error analysis, we shall consider a general case by using a weighting $(N+1)L\times(N+1)L$ matrix $\boldsymbol{\Sigma}$. From (5.72) we have:

$$\mathbb{E}||\boldsymbol{w}_{e,i}||^2_{\boldsymbol{\Sigma}} = \mathbb{E}\left\{\left[\boldsymbol{\mathcal{B}}_{c,i}\boldsymbol{w}_{e,i-1} - \begin{bmatrix}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i\\ \boldsymbol{0}_{L\times 1}\end{bmatrix} + \gamma\boldsymbol{z}\right]^{\top}\boldsymbol{\Sigma}\left[\boldsymbol{\mathcal{B}}_{c,i}\boldsymbol{w}_{e,i-1} - \begin{bmatrix}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i\\ \boldsymbol{0}_{L\times 1}\end{bmatrix} + \gamma\boldsymbol{z}\right]\right\} \tag{5.80}$$

with $\boldsymbol{\Sigma}'$ defined in (5.52). Using the independence 5.1 we get:

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{\boldsymbol{\Sigma}} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i-1}||^2_{\boldsymbol{\Sigma}'} + \mathbb{E}\{\boldsymbol{s}_i^\top \boldsymbol{\mathcal{M}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{M}}\boldsymbol{s}_i\} + \gamma^2||\boldsymbol{z}||^2_{\boldsymbol{\Sigma}} + 2\,\gamma\mathbb{E}\{\boldsymbol{w}_{e,i-1}^\top\}\boldsymbol{\mathcal{B}}_c^\top\boldsymbol{\Sigma}\boldsymbol{z} \tag{5.81}$$

where $\boldsymbol{\mathcal{B}}_c$ is defined in (5.40).

The first two terms of the RHS (5.81) remain unchanged and are calculated in the previous section (5.52) and (5.48), respectively and the third term is constant. The final term depends on the difference between the shared objective vector $\boldsymbol{x}^*$ and each agent minimiser $\boldsymbol{w}_k^o$.

Proceeding as we did in the previous section, we rewrite (5.81) using the vectorised version of the weighting matrix $\boldsymbol{\Sigma}$ leading to

$$\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{\boldsymbol{\sigma}} = \underbrace{\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i-1}||^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}}}_{\text{Data term}} + \underbrace{\text{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top\boldsymbol{\sigma}}_{\text{Noise term}} + \underbrace{\gamma^2||\boldsymbol{z}||^2_{\boldsymbol{\sigma}} + 2\,\gamma(\boldsymbol{z}^\top \otimes \mathbb{E}\{\boldsymbol{w}_{e,i-1}^\top\}\boldsymbol{\mathcal{B}}_c^\top)\boldsymbol{\sigma}}_{\text{Bias term}} \tag{5.82}$$

where $\boldsymbol{\sigma}$, $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{Y}}$ are defined in (5.52), (5.57) and (5.50), respectively. Compared to (5.58), we have two additional terms that would rather be equal to zero if all the agents were to estimate the same vector $\boldsymbol{w}^*$.

It is worth mentioning that the bias term is depending on the regularisation parameter. Such result allows control over the mean square deviation.

The algorithm stability in the mean-square sense is achieved if and only if the matrix $\boldsymbol{\mathcal{F}}$ defined in (5.57) is stable i.e. all of its eigenvalues strictly lie inside the unite disc. Such condition is verified for sufficiently small step-sizes μ_k and $\bar{\mu}$ such as:

$$\begin{aligned} &0 < \mu_k < \tfrac{1}{(N+1)(\gamma+\lambda_{\max}(R_{u_k}))} \\ &0 < \bar{\mu} < \tfrac{1}{\gamma N(N+1)} \end{aligned} \tag{5.83}$$

Note that the algorithm stability conditions remain the same under this scenario. However, its Mean Square Deviation (MSD) is greater as we shall demonstrate in the sequel.

5.5.4 Network Mean-Square Performance

Using the equation (5.82), we can evaluate the network as well as every single agent accuracy. As the algorithm is stable for sufficiently small step-sizes (5.83) we can take the limits of the equation (5.82) as:

$$\lim_{i\to\infty}\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{\boldsymbol{\sigma}} = \lim_{i\to\infty}\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i-1}||^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}} + \text{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top\boldsymbol{\sigma} + \gamma^2||\boldsymbol{z}||^2_{\boldsymbol{\sigma}} + 2\gamma\lim_{i\to\infty}(\boldsymbol{z}^\top \otimes \mathbb{E}\{\boldsymbol{w}_{e,i-1}^\top\}\boldsymbol{\mathcal{B}}_c^\top)\boldsymbol{\sigma} \tag{5.84}$$

Grouping terms leads to:

$$\lim_{i\to\infty}\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{(\boldsymbol{I}-\boldsymbol{\mathcal{F}})\boldsymbol{\sigma}} = \text{vec}(\boldsymbol{\mathcal{Y}}^\top)^\top\boldsymbol{\sigma} + \gamma^2||\boldsymbol{z}||^2_{\boldsymbol{\sigma}} + 2\gamma\lim_{i\to\infty}(\boldsymbol{z}^\top \otimes \mathbb{E}\{\boldsymbol{w}_{e,i-1}^\top\}\boldsymbol{\mathcal{B}}_c^\top)\boldsymbol{\sigma} \tag{5.85}$$

Replacing the term $\lim_{i\to\infty} \mathbb{E}\{\boldsymbol{w}_{e,i-1}^{\top}\}$ by (5.79) results in:

$$\lim_{i\to\infty} \mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{(\boldsymbol{I}-\boldsymbol{\mathcal{F}})\boldsymbol{\sigma}} = \underbrace{\text{vec}(\boldsymbol{\mathcal{Y}}^{\top})^{\top}\boldsymbol{\sigma}}_{\text{Noise term}} + \underbrace{\gamma^2||\boldsymbol{z}||^2_{\boldsymbol{\sigma}} + 2\gamma\,(\boldsymbol{z}^{\top}\otimes(\boldsymbol{I}-\boldsymbol{\mathcal{B}}_c)^{-1}\boldsymbol{z}^{\top}\boldsymbol{\mathcal{B}}_c^{\top})\boldsymbol{\sigma}}_{\text{Bias term}} \tag{5.86}$$

It is worth mentioning that, compared to (5.45), an additional bias term is introduced leading to a greater MSD. The bias term is highly dependent on the difference between the vectors to estimate and the regularisation parameter γ. As a consequence, in the case where all agents are estimating the right task, the vector $\boldsymbol{z}$ would be equal to $\boldsymbol{0}_{(N+1)L}$ which leads to the same unbiased MSD expressed in (5.45).

5.5.5 Transient State Analysis

The algorithm transient behaviour can be easily studied under optimal conditions i.e. when all the agents are properly clustered by following the same procedure as [Sayed et al., 2013]. However, the analysis is slightly different when the some of agents are not. To analyse the algorithm under this unfavourable condition, we iterate the equation (5.82) from $i = 0$ to find:

$$\begin{aligned}\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{\boldsymbol{\sigma}} = \mathbb{E}||\widetilde{\boldsymbol{w}}_{e,-1}||^2_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}} + \text{vec}(\boldsymbol{\mathcal{Y}}^{\top})^{\top}\sum_{j=0}^{i}\boldsymbol{\mathcal{F}}^{j}\boldsymbol{\sigma} + \gamma^2\sum_{j=0}^{i}||\boldsymbol{z}||^2_{\boldsymbol{\mathcal{F}}^{j}\boldsymbol{\sigma}} \\ + 2\gamma\sum_{j=0}^{i}(\boldsymbol{z}^{\top}\otimes\mathbb{E}\{\boldsymbol{w}_{e,j-1}^{\top}\}\boldsymbol{\mathcal{B}}_c^{\top})\boldsymbol{\mathcal{F}}^{i-j}\boldsymbol{\sigma}\end{aligned} \tag{5.87}$$

From (5.75) we have:

$$\mathbb{E}\{\widetilde{\boldsymbol{w}}_{e,i}\} = \boldsymbol{\mathcal{B}}_c^{i+1}\mathbb{E}\{\widetilde{\boldsymbol{w}}_{e,-1}\} + \gamma\sum_{j=1}^{i}\boldsymbol{\mathcal{B}}_c^{j}\boldsymbol{z} \tag{5.88}$$

Replacing $\widetilde{\boldsymbol{w}}_{e,i-1}$ by its recursion (5.88) leads to:

$$\begin{aligned}\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,i}||^2_{\boldsymbol{\sigma}} = \overbrace{\mathbb{E}||\widetilde{\boldsymbol{w}}_{e,-1}||^2_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}}}^{\text{Data term}} + \overbrace{\text{vec}(\boldsymbol{\mathcal{Y}}^{\top})^{\top}\sum_{j=0}^{i}\boldsymbol{\mathcal{F}}^{j}\boldsymbol{\sigma}}^{\text{Noise term}} \\ \underbrace{+\gamma^2\sum_{j=0}^{i}||\boldsymbol{z}||^2_{\boldsymbol{\mathcal{F}}^{j}\boldsymbol{\sigma}} + 2\gamma\sum_{j=1}^{i+1}(\boldsymbol{z}^{\top}\otimes\mathbb{E}\{\boldsymbol{w}_{e,-1}^{\top}\boldsymbol{\mathcal{B}}_c^{\top\,j}\})\boldsymbol{\mathcal{F}}^{i-j+1}\boldsymbol{\sigma} + 2\gamma\sum_{j=1}^{i}\sum_{k=0}^{j-1}(\boldsymbol{z}^{\top}\otimes\boldsymbol{z}^{\top}\boldsymbol{\mathcal{B}}_c^{\top k})\boldsymbol{\mathcal{F}}^{i-j}\boldsymbol{\sigma}}_{\text{Bias term}}\end{aligned} \tag{5.89}$$

From (5.89) we observe that the mean square error is degraded compared to the case where all the agents are correctly labelled. This is due to the introduced bias term that depends on the distance between the miss-assigned objective vectors and the correct ones, and the regularisation parameter γ. It is interesting to notice that the bias term not only affects the steady state performance as stated in (5.86) but also the transient behaviour of the algorithm.

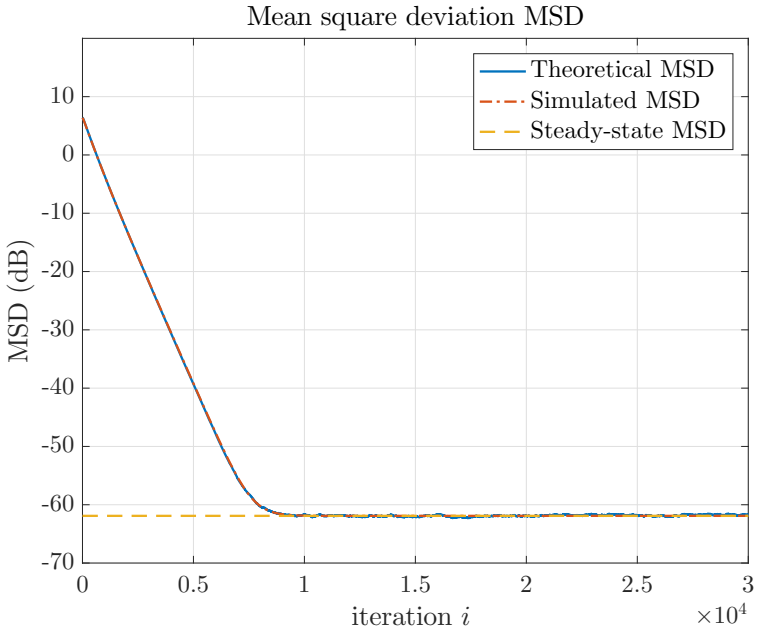

Figure 5.3: Theoretical and simulated mean-square deviation (MSD). For this experiment, we consider a network of $N = 10$ agents, data dimension $L = 2$, Gaussian regression data $\boldsymbol{u}_{k,i} \sim \mathcal{N}(0, I_L)$ and additive noise $v_k(i) \sim \mathcal{N}(0, 10^{-3})$. The step-size are set to $\mu_k = 10^{-3}$, $\bar{\mu} = 10^{-4}$ and the regularisation parameter $\gamma = 0.5$.

5.6 Numerical analysis

This section shall be divided into three parts. The first part consists of confirming the theoretical models fitness for both previously described scenarios. In the second part, we shift the focus to the algorithm performance compared to a single agent LMS and assess the effect of the miss-clustered agents for various ratios of clustered/miss-clustered agents. Finally, we study the effect of the step-size parameters on the algorithm convergence.

5.6.1 Theoretical model accuracy

For this first experiment we consider a network of $N = 10$ agents and an $L = 2$ objective vector $\boldsymbol{w}^o$. The regression data is drawn from a Gaussian distribution $\boldsymbol{u}_{k,i} \sim \mathcal{N}(0, I_L)$ and so are the noise scalars $v_k(i) \sim \mathcal{N}(0, 10^{-3})$. We set the step sizes to $\mu_k = 10^{-3}$, $\bar{\mu} = 10^{-4}$ and the regularisation parameter $\gamma = 0.5$. The simulated results are averaged over 100 Monte-Carlo runs. The Figure 5.3 clearly confirms the theoretical model accuracy.

In the previous experiment we considered the algorithm under optimal conditions where every agent is assigned the right cluster. Let us now consider the case where some agents are wrongly assigned. We consider a network of $N = 5$ agents with one cluster agent. However, instead of estimating one objective vector, we estimate two $\boldsymbol{w}_1^0 \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ and $\boldsymbol{w}_2^0 = \boldsymbol{w}_1^0 + 10\mathbf{1}_L$. We assign the first one to $N_1 = 3$ agents and the second task to the two remaining agents ($N_2 = 2$). We used the same step-size parameters and data profile as the previous experiment. Note that in the absence of a second cluster agent, all the weights α_{kq} are set to 1 leading to the formation of one cluster instead

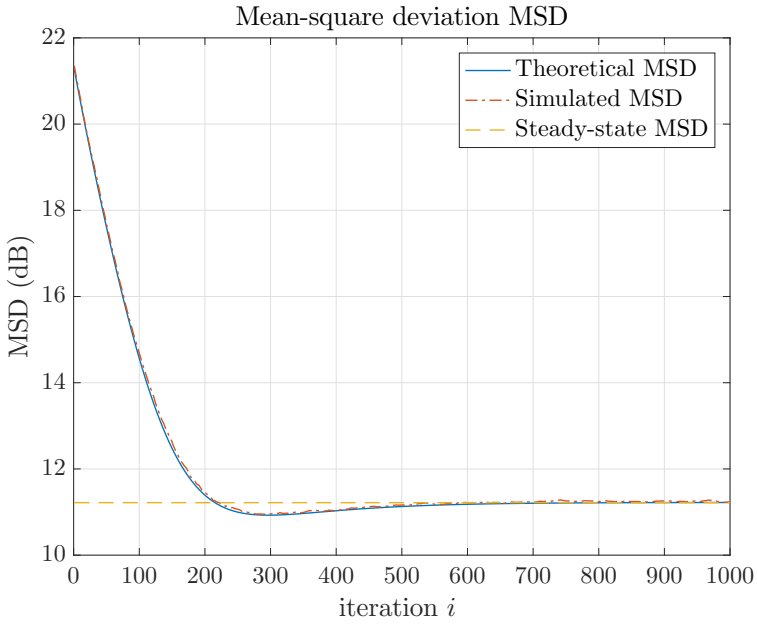

Figure 5.4: Theoretical and simulated mean-square deviation (MSD) when a subset of agents is miss-assigned. We consider two objective vectors $\boldsymbol{w}_1^0 \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ and $\boldsymbol{w}_2^0 = \boldsymbol{w}_1^0 + 10\,\mathbf{1}_L$, a network of $N = 5$ agents $N_1 = 3$ tasked with estimating $\boldsymbol{w}_1^o$ and $N_2 = 2$ estimating $\boldsymbol{w}_2^o$. The remaining parameters remains the same as in the previous experiment.

of two. In the Figure 5.4, we can observe that the theoretical model correctly fits the simulated results. Note that the theoretical model correctly fits the inertia resulting from a high step-size parameter μ_k.

5.6.2 Multitask performance

Now that the theoretical models are verified, let us assess the algorithm performance. We first compare it with the single agent LMS. For this part of the experiment we considered a larger network of $N = 100$, step-sizes $\mu_k = 10^{-2}$. Note that we did not carry out the theoretical analysis because of its high numerical complexity. As it is illustrated in Figure 5.5 the proposed algorithm outperforms its single agent counterpart as it takes advantage of the shared data through the regularisation term.

Next, we investigate the algorithm weights convergence trajectories. To do so, we consider a network of $N = 30$ and $Q = 3$ tasks with two sets of objective vectors $\boldsymbol{w}_1^o = \text{col}\{-1.5, -1\}$, $\boldsymbol{w}_2^o = \text{col}\{1.5, -1\}$ and $\boldsymbol{w}_3^o = \text{col}\{0, 1.5\}$ and $\boldsymbol{w}_{1'}^o = \text{col}\{-1.5, 1\}$, $\boldsymbol{w}_{2'}^o = \text{col}\{1.5, 1\}$ and $\boldsymbol{w}_{3'}^o = \text{col}\{0, -1.5\}$. We set the adaptation step-size parameters to $\mu = \mu_\alpha = \bar{\mu} = 10^{-3}$ and $\gamma = 10^{-1}$. Figure 5.6 depicts the trajectories followed by the local estimates $\boldsymbol{w}_{k,i}$ for different objective vectors and initial local estimates. We can indeed confirm that the local estimates converge to their respective objective vectors $\boldsymbol{w}_q^o$. Furthermore, we can observe the clustering effect as the trajectories get tighter the more they approach the objective vector. Note that having uniform initial weights α_{kq} tends to shrink the cluster and regular agents estimates to the centre. This behaviour is expected as such initial condition drives the estimate towards a common minimiser for all tasks.

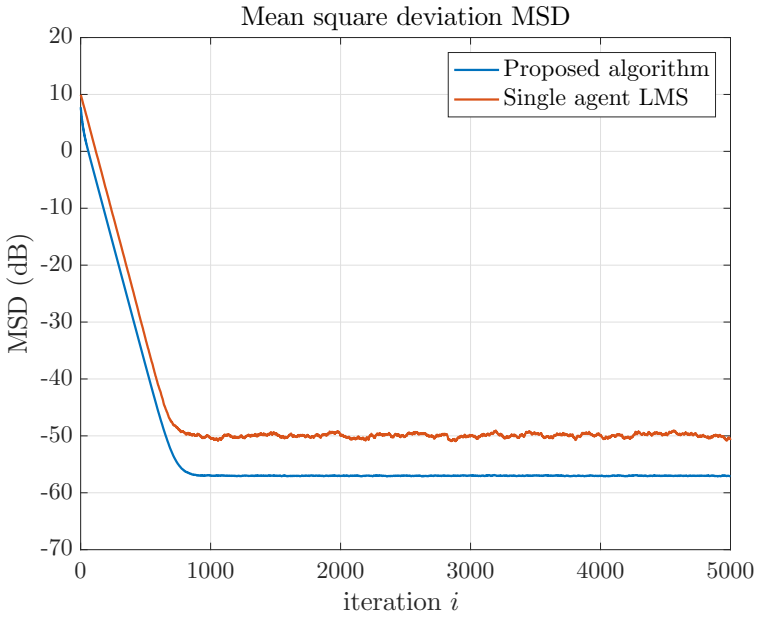

Figure 5.5: Algorithm comparison with single task LMS. We consider a network of $N = 100$ and a data dimension of $L = 2$. The objective vector is drawn from a Gaussian distribution $\boldsymbol{w}^o \sim \mathcal{N}(0, \boldsymbol{I}_L)$ and the step-size parameters, for both algorithms, are set to $\mu_k = 10^{-2}$. The regression data is drawn from a Gaussian distribution $\boldsymbol{u}_{k,i} \sim \mathcal{N}(0, \boldsymbol{I}_L)$ and so are the noise scalars $v_k(i) \sim \mathcal{N}(0, 10^{-3})$.

Let us assess the effect of miss-labelled agents. We consider a network of $N = 30$ agents with one cluster agent. We affect two different objective vectors $\boldsymbol{w}_1^0$ and $\boldsymbol{w}_2^0$ to two subsets of agents N_1 and N_2 of varying sizes, where

$$\boldsymbol{w}_1^o = \boldsymbol{w}_t \tag{5.90}$$

$$\boldsymbol{w}_2^o = \boldsymbol{w}_t + K\mathbf{1}_2 \tag{5.91}$$

and $\boldsymbol{w}_t \sim \mathcal{N}(0, \boldsymbol{I}_2)$. The bigger subset determines the main objective vector $\boldsymbol{w}^*$. For instance, if the vector $\boldsymbol{w}_1^o$ is assigned to $N_1 = 20$ agents then $\boldsymbol{w}^* = \boldsymbol{w}_1^o$. We considered various ratios $\frac{N_1}{N_2}$. The parameter K is used to control the distance between the two objective vectors. We used different values of K ranging from $K = 0$ to $K = 10$. The Figure 5.7, illustrates the mean square deviation for various ratios of miss-labelled agents. Note that the ratio $\frac{N_1}{N_2}$ can reach a maximum of only 50% due to the definition of $\boldsymbol{w}^*$.

We can observe a rise of the mean-square deviation with the ratio of miss-labelled agents as it reaches a peak at 50% and a minimum when $N_1 = N$ or $N_2 = N$. These two cases correspond to the correct behaviour of the algorithm or when the parameter $k = 0$. Note that the performance suffers the most for high values of the coefficient K. This result was expected as the bias term is highly dependent on the distance between the objective vectors.

In this final part, we investigate the effect of the step-size parameters on the algorithm convergence. We consider the same setting as in the previous experiment, namely a network of $N = 30$ agents and $Q = 3$, we also consider the same data profile. In Figure 5.8, we depict the trajectories of the local (left hand-side) and centroids (right hand-side) estimates, for a large cluster step-size parameter

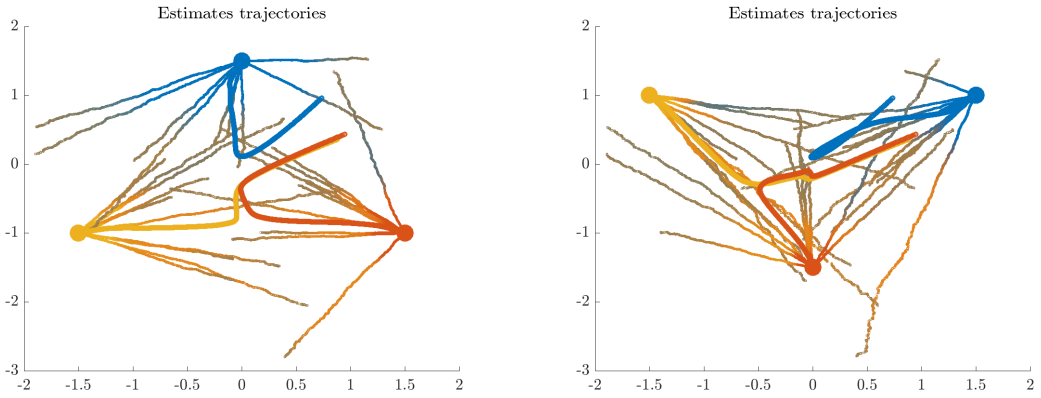

Figure 5.6: Local estimates trajectories and cluster agents estimates (thick lines) and their respective objective vectors. For these experiments, we considered a network of $N = 30$ agents, $Q = 3$ tasks, step-sizes $\mu_k = 10^{-2}$, $\bar{\mu} = 10^{-3}$, $\mu_\alpha = 10^{-4}$ and a regularisation parameter $\gamma = 10^{-1}$. The $(L \times 1)$ objective vector $\boldsymbol{w}_q^0$ are set to $\boldsymbol{w}_1^o = \text{col}\{-1.5, -1\}$, $\boldsymbol{w}_2^o = \text{col}\{1.5, -1\}$ and $\boldsymbol{w}_3^o = \text{col}\{0, 1.5\}$ for the left-hand side figure and $\boldsymbol{w}_{1'}^o = \text{col}\{-1.5, 1\}$, $\boldsymbol{w}_{2'}^o = \text{col}\{1.5, 1\}$ and $\boldsymbol{w}_3^o = \text{col}\{0, -1.5\}$ for the right-hand one. The regression data $\boldsymbol{u}_{k,i}$ and the noise scalars $v_k(i)$ are, as in the previous experiments, drawn from Gaussian distributions.

$\bar{\mu} = 10^{-2}$. Surprisingly, the algorithm does converge to the right objective vectors. However, the regular agents reach consensus quicker compared to the previous experiment. Such behaviour is due to the quick convergence of the centroids which end up leading the agents towards the minimisers.

Note that a larger centroid step-size $\bar{\mu}$ would eventually lead to the algorithm divergence as it can be observed in Figure 5.9. Only the blue cluster converged to the right vector. The remaining agents were not clustered and could not reach their respective objectives.

We now repeat the same experiment. However, this time we use a larger step-size for the coefficients α_{kq}. Considering a similar setting as previously except for $\bar{\mu}$ that is set this time to $\bar{\mu} = 10^{-3}$ and $\mu_\alpha = 10^{-2}$. As expected, the algorithm does not converge to the right minimisers as the clustering task failed to estimate the clusters. Note that, in contrast with the previous experiment, all the agents were assigned to a cluster but not necessarily the correct one. In order to converge properly, the algorithm must be correctly set up. The local estimates must have a head start over the centroids estimate, and the centroids estimates over the clustering coefficients.

With the previously discussed step-size parameter in mind, in Figure 5.11, we consider a two-stage estimation strategy. To do so, we use two different regularisation parameters. We begin with a very small one $\gamma = 10^{-8}$. Such small value allows the agents to converge towards an accurate estimate of their objective vectors, which reduces the chances of miss-clustering agents. In the next phase, we use a larger regularisation parameter $\gamma = 8\ 10^{-1}$. Such increase of the regularisation parameter γ allows more collaboration which leads to a more accurate estimate. Note that such strategy also allows a more accurate estimation of the clusters.

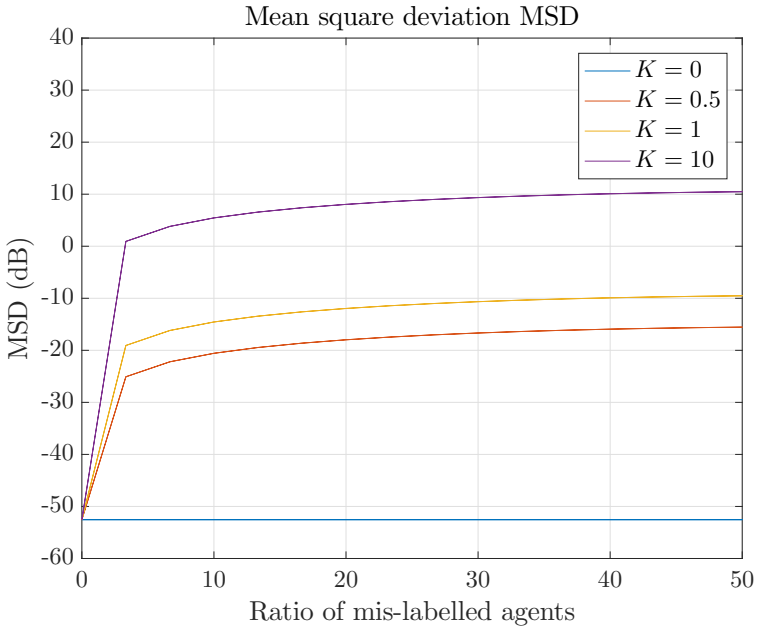

Figure 5.7: The mean square deviation (MSD) for different numbers of miss-labelled tasks. We consider a Network of $N = 30$ agents, L dimensioned objective vectors $\boldsymbol{w}_1^0$ and $\boldsymbol{w}_2^0$, K is a scalar varying from $K = 0$ to $K = 10$. We kept the same step-sizes, regression data and noise parameters as the previous experiment.

5.7 Conclusion

As a result of data segmentation over networks, multitask optimisation has become a necessary tool to infer information. In order to solve this sort of optimisation problems, in an unsupervised manner, we first proposed an unsupervised clustered multitask algorithm where the information exchange is carried out through a stochastically estimated regularisation parameter. Then we performed a theoretical analysis in the mean and mean-square sense under two scenarios. One considering a perfect clustering of the agents and a second where some of them were miss-clustered. Finally, we performed several numerical experiments to validate the theoretical models for both scenarios and to quantify the algorithm performance.

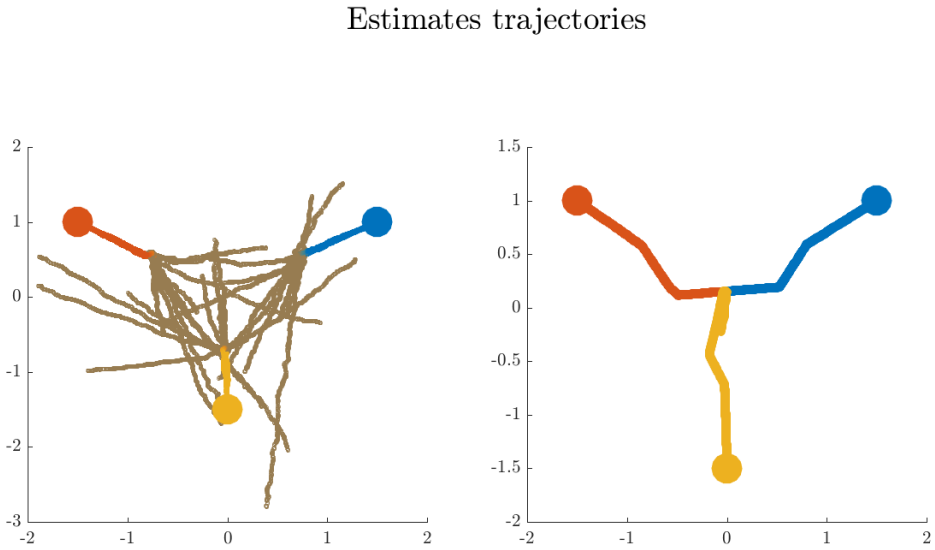

FIGURE 5.8: Local and cluster agents estimates trajectories and their respective objective vectors. For this experiment, we consider a network of $N = 30$ agents, $Q = 3$ tasks, step-sizes $\mu_k = 10^{-3}$, $\bar{\mu} = 10^{-2}$, $\mu_\alpha = 10^{-4}$ and a regularisation parameter $\gamma = 10^{-1}$. The $(L \times 1)$ objective vector $\boldsymbol{w}_q^0$ are set to $\boldsymbol{w}_1^o = \text{col}\{-1.5, 1\}$, $\boldsymbol{w}_2^o = \text{col}\{1.5, 1\}$ and $\boldsymbol{w}_3^o = \text{col}\{0, -1.5\}$. The regression data $\boldsymbol{u}_{k,i}$ and the noise scalars $v_k(i)$ are, as in the previous experiments, drawn from Gaussian distributions.

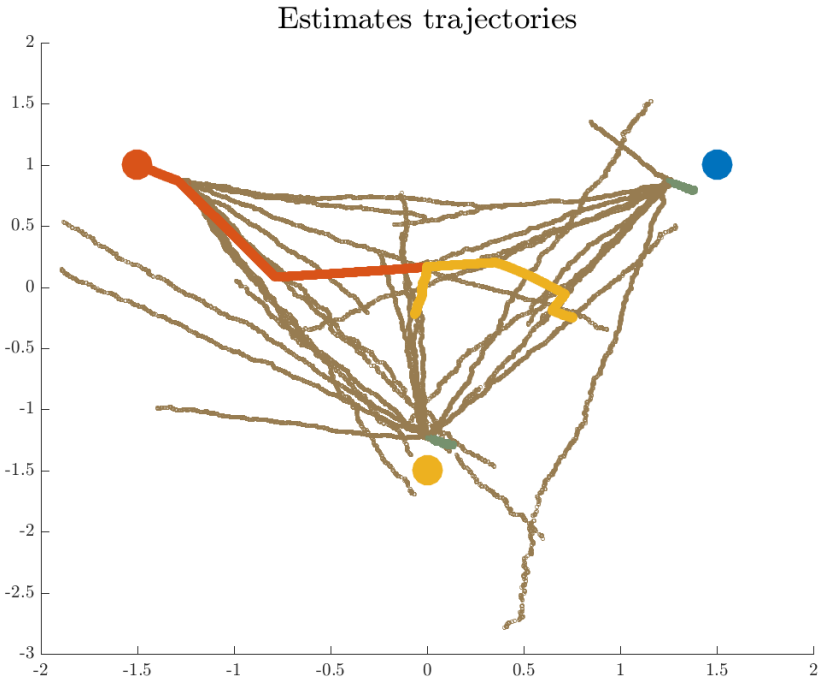

Figure 5.9: Local and cluster agents estimates trajectories and their respective objective vectors. For this experiment, we consider a network of $N = 30$ agents, $Q = 3$ tasks, step-sizes $\mu_k = 10^{-3}$, $\bar{\mu} = 10^{-1}$, $\mu_\alpha = 10^{-4}$ and a regularisation parameter $\gamma = 10^{-1}$. The $(L \times 1)$ objective vector $\boldsymbol{w}_q^0$ are set to $\boldsymbol{w}_1^o = \text{col}\{-1.5, 1\}$, $\boldsymbol{w}_2^o = \text{col}\{1.5, 1\}$ and $\boldsymbol{w}_3^o = \text{col}\{0, -1.5\}$. The regression data $\boldsymbol{u}_{k,i}$ and the noise scalars $v_k(i)$ are, as in the previous experiments, drawn from Gaussian distributions.

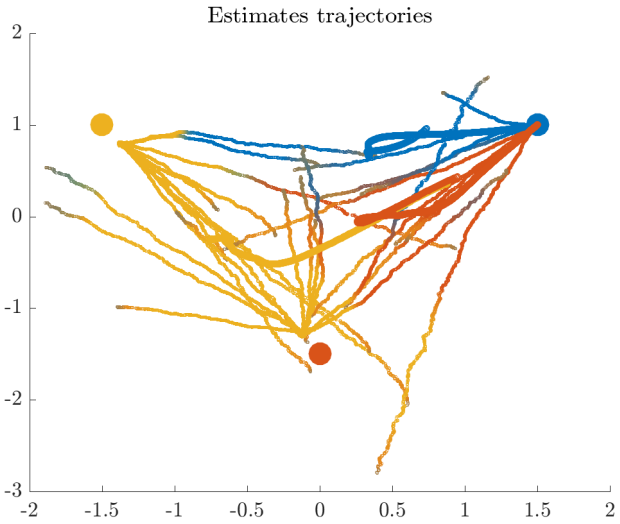

Figure 5.10: Local estimates trajectories and their respective objective vectors. We consider a network of $N = 30$ agents, $Q = 3$ tasks, step-sizes $\mu_k = 10^{-3}$, $\bar{\mu} = 10^{-3}$, $\mu_\alpha = 10^{-2}$ and a regularisation parameter $\gamma = 10^{-1}$. The $(L \times 1)$ objective vector $\boldsymbol{w}_q^0$ are set to $\boldsymbol{w}_1^o = \text{col}\{-1.5, 1\}$, $\boldsymbol{w}_2^o = \text{col}\{1.5, 1\}$ and $\boldsymbol{w}_3^o = \text{col}\{0, -1.5\}$. The regression data $\boldsymbol{u}_{k,i}$ and the noise scalars $v_k(i)$ are drawn from Gaussian distributions.

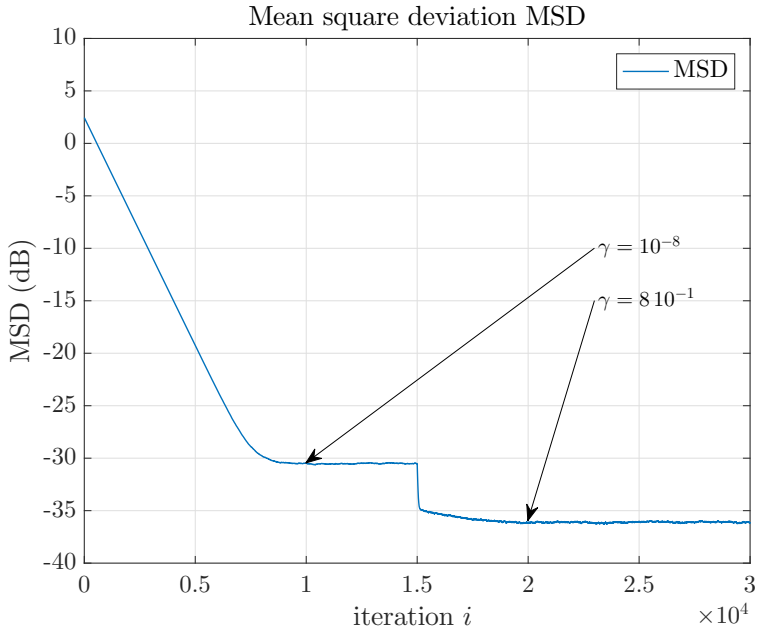

Figure 5.11: The mean square deviation (MSD) for a two-stage estimation strategy. We consider the same setting as in the previous experiment except for the step-size and the regularisation parameters. In the beginning, the regularisation parameter is set to $\gamma = 10^{-8}$ to limit the collaboration between the cluster regular agents. After convergence of the regular agents, we set $\gamma = 8\ 10^{-1}$ to improve the accuracy. Note that we set the step-size parameters to $\mu_k = \frac{10^{-3}}{\gamma}$, $\bar{\mu} = \frac{10^{-3}}{\gamma}$ and $\mu_\alpha = \frac{5\ 10^{-4}}{\gamma}$.

6 Conclusion

Contents

As the world gets more connected and networked, distributed estimation becomes more and more attractive, as it takes advantage of this interconnection to enhance its performance. However, as a result of this high connectivity, we are now confronted by unprecedented data quantities. With such quantities, new challenges were raised. First, transferring such colossal amounts of data without risking a network congestion. Next, processing the data without breaching its privacy. As a consequence of this growth, more parameters need to be inferred requiring autonomous multitask networks able to find their respective clusters without any supervision. Throughout this dissertation, we tried to address each of these concerns. In this ultimate chapter, we shall summarise and discuss the main contributions and solutions we proposed to tackle these challenges. We then follow up that summary by a reflection on the findings and discuss future research directions.

6.1 Summary of results

Diffusion strategies offer many advantages compared to their centralised counterparts such as link and agent failures. However, these advantages come with a substantial burden on network resources. Indeed, diffusion strategies require constant communication between neighbouring agents. As these agents usually communicate wirelessly, it is sometimes hard to maintain such constant communication especially for agents with limited energy budgets as it is the case for most WSN networks. As many applications require some aspects of privacy preservation such as in hospitals and banks, data privacy also became a critical aspect for any distributed strategy. In the following, we sum up the methods and algorithms proposed to tackle these highly critical challenges.

6.1.1 Energy and network resources management

As the radio communication is the bottleneck for both energy and network resources management, we decided to tackle these challenges from two different angles: through compression and network

sparsification. We then proposed an algorithm where the gradient vectors are fully transmitted while the local estimates are partially shared. The missing entries are compensated by the local ones. In [Arablouei et al., 2014b], at every iteration, due to the entry selection method, the number of transmitted entries could potentially reach L (data dimension). In this case the network must be over-dimensioned to avoid congestions. In contrast, with CD we guarantee a set number M of transmitted entries which allows a more efficient network resources allocation. The theoretical analysis of this first algorithm served as a basis for the rather challenging analysis of the fully compressed version of the algorithm. This fully compressed version added a second compression layer as the gradient vectors are also compressed.

As both algorithms rely on a random selection matrix, we preceded the algorithms theoretical analysis by a study of the selection matrix first and second order moments. Using these moments, we analysed both algorithms in the mean and mean-square sense. Finally, we conducted an extensive numerical analysis. We began by testing the fitness of the theoretical models. Then we tested the algorithms performance for different compression ratios for large networks and high dimensional data. As expected, high compression ratios yield less accurate estimates. This is due to the limited quantity of shared information. We concluded the numerical analysis with a scenario considering the scarcity of the energy resources. We considered agents endowed with batteries and solar energy harvesting capabilities. As the batteries sizes are limited, the nodes turn on and off for a pre-calculated duration to save energy. The newly introduced version of the algorithm outperformed all of its state of the art counterparts.

While the first approach considers compression in a single task setting, for this second one, we take advantage of the multitask setting to limit the communication load and thus reducing energy consumption. In this scenario, it is beneficial to sparisify the network graph in a way where only agents sharing similar tasks are connected. For this approach, we considered a slightly different network structure with two types of agents: cluster agents and regular ones. The cluster agents seek to estimate two parameters: the cluster centroids and the network structure. The regular agents estimate their local estimates through a regularised cost function. The quantities estimated by the cluster agents are used in the regularisation function.

In order to quantify the algorithm performance, we first theoretically analysed it under two scenarios. First, we considered the algorithm under optimal conditions where all the agents are correctly clustered. In this case, the algorithm converges toward the correct minimiser provided some conditions on the step-sizes. The second scenario considered the case where some agents are assigned to the wrong clusters. Under this scenario, the algorithm converged towards a biased solution. The bias depends on two factors: the distances between each wrongly assigned objective vector and the correct ones. In contrast with the first factor, the second one, the regularisation parameter γ, can be controlled to limit the estimation error. We also derived stability conditions in both mean and mean-square sense.

In addition to the theoretical study, we numerically analysed the algorithm under the two previously mentioned scenarios. We first verified both theoretical models accuracies. Then we tested the algorithm performance for different ratios of miss-clustered agents. We also studied the effect of

different adaptation step-sizes of the estimates on the algorithm convergence. Based on these studies, we proposed a two-stage estimation strategy to enhance the algorithm performance.

More details about the compression and network sparsification methods can be found in the Compressed Diffusion LMS and Unsupervised clustered multitask learning chapters.

6.1.2 Privacy preservation

While network and energy resources management is critical for diffusion strategies, data privacy may be even more critical for some applications. To address this concern, we proposed to limit the amount of shared information by setting the weighing matrix $\boldsymbol{A}$ to the identity. To secure the information carried through the gradients, we took inspiration from the *differential privacy* principle. The main idea behind this technic is to corrupt the information so the statistical structure of data is conserved while the individual information privacy is preserved. To do so, we used random Wishart matrices to corrupt the transmitted gradient vectors while preserving the descent direction.

Similarly to the previous work on the compression aspect, we started by studying different moments of the Wishart matrix to ease the theoretical analysis of the algorithm. We then went on to theoretically analysing the algorithm both in the mean and mean-square sense. This analysis allowed us to derive convergence conditions for the first and second order moments of the error vectors. Unsurprisingly, both these conditions depend on the eigenvalues and rank of the Wishart matrix used to corrupt the data. We finally performed a numerical analysis. We first confirmed the theoretical model accuracy. Then, we tested the algorithm performance for various degrees of freedom of the Wishart matrix. We noticed that higher orders of corruption matrix lead to a low mean-square deviation. However, this same condition facilitates the reconstruction of the data. We also found out that the most secure option is to consider a singular Wishart matrix to corrupt the data as such matrix is irreversible. However, this option yields the largest mean square deviation as the data are projected onto a one dimensional space. We detail the method and its analysis in Privacy aware diffusion LMS chapter.

6.2 Future works and discussion

In this very last section, we first discuss some ways to improve and build upon the proposed methods and algorithms. Then we explore some new research directions related to the field.

6.2.1 Improvements related to the presented work

Starting with Compressed diffusion LMS, we set the weighing matrix $\boldsymbol{A}$ to identity matrix to limit the communication cost. However, it would be interesting to formulate a general theoretical model that would include the *partial diffusion LMS* introduced in [Arablouei et al., 2014b]. Furthermore, even if the matrix $\boldsymbol{A}$ is set to the identity, the local estimate vector would still be transmitted for the estimation of the stochastic gradients. Although we numerically studied a version of the

algorithm that takes advantage of this information in the numerical analysis, it would be of interest to carry out an in-depth theoretical analysis. Such analysis would give an insight into the algorithm behaviour and help derive important quantities such convergence conditions.

In Privacy aware diffusion LMS, we took inspiration from the *differential privacy* principle that is usually applied to finite data-sets to a stream of data. While it is possible to quantify an algorithm privacy for finite data-sets, we were not able to achieve such quantification in the case of the proposed distributed algorithm. Studying this aspect of the algorithm would be a good future direction. As a mean to reduce the quantity of exchanged information, we set the weighting matrix $\boldsymbol{A}$ to the identity. It would be interesting to consider an ATC version of the algorithm and derive a global theoretical model. It would be also of interest to consider different matrices to alter the data instead of a Wishart matrix and quantify their induced privacy.

The Unsupervised clustered multitask learning chapter dealt with a multi-task optimisation problem. We proposed to use two types of agents: cluster and regular agents. The regular agents only communicated with the cluster ones. It would be interesting to study the case where the agents are allowed to communicate with their neighbours within the same cluster. As the algorithm learns the clusters structure, it could be used a pre-processing phase for *multi-task diffusion LMS* to estimate the clusters. Such study would be an interesting lead.

In the entirety of this dissertation, we considered that the data are related to the collected measurement through a linear regression model. However, most phenomenon cannot be linearly modelled. It would therefore be important to consider non-linear models such as kernel-based and polynomial models. It would be also interesting to consider a graph-signal approach study for the future diffusion algorithms such as [Nassif et al., 2017a].

6.2.2 New research directions

Random matrix theory was used in various applications [Couillet et al., 2015, Couillet and McKay, 2014, Couillet et al., 2016]. However, it has not been applied in a distributed online setting. Although It is relatively simple to theoretically study diffusion strategies, they might be extremely challenging to analyse when considering large networks and high dimensional data. We are currently working on using random matrix theory to theoretically analyse diffusion LMS algorithm for large networks and large scale data.

While the theoretical results developed in [Sayed, 2013a] remain valid for large networks and high dimensional data, they happen to be tricky to implement. They require enormous amount of memory and a phenomenal computational power. The traditional theoretical analysis as it is featured in [Sayed, 2013a] has a numerical complexity of $\mathcal{O}((L \times N)^4)$, albeit, with a more sophisticated implementation, it can be reduced to $\mathcal{O}(N \times L^2 \times \overline{\mathcal{N}_m}^2)$, where $\overline{\mathcal{N}_m}$ denotes the mean number of neighbours. Random matrix theory allows an even better improvement by L folds utilizing the diagonal nature of the blocks constructing the metric $\boldsymbol{\mathcal{F}}$. It is of course possible to use that same strategy for the traditional method but the cost of the diagonalisation outweighs its benefits as

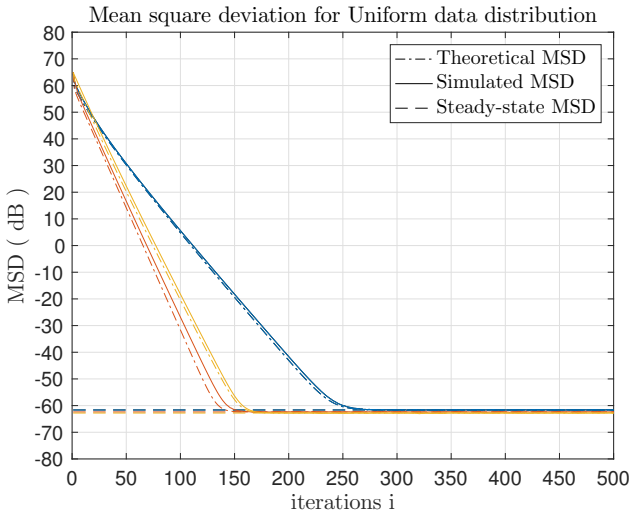

Figure 6.1: Mean square deviation (MSD) using random matrix theory for Uniform data distribution for three covariance matrices. We consider a highly connected network of $N = 4\ 10^4$ agents. We set data dimension to $L = 400$, the step-size $\mu = 10^{-1}$ and the matrix $\boldsymbol{A}$ is set to the identity matrix. We set the noise variance to $\sigma_v^2 = 10^{-3}$. We drew the regression vectors $\boldsymbol{u}_{k,i}$ from a uniform distribution over [-2 2].

it drastically increases the complexity. It is worth mentioning that in the special case where the weighing matrix $\boldsymbol{A} = \boldsymbol{I}_N$ the computational complexity drops to $\mathcal{O}(N \times L)$.

In addition to the computational efficiency enhancement, random matrix theory allows a complete independence of the data model probability law. For instance, Marcenko-Pastur theorem does not have any requirement related to the probability distribution other than the independence condition. It is a very powerful tool as it allows theoretical analysis for numerous data model that would be rather challenging if not impossible.

Considering a very large network and high dimensional data, we have some preliminary results for different data distribution laws, namely: Uniform, Poisson and Bernoulli distributions, depicted respectively, in Figures 6.1, 6.2, 6.3.

While random matrix theory seems promising for diffusion LMS, optimal transport is the way forward in the multitask setting. For instance, in [Courty et al., 2017], the authors use optimal transport to solve a domain adaptation task. Such method could be adapted for a heterogenous multitask network, where agents estimate similar tasks but do not necessarily agree on the data representation. Using domain adaptation in a way where each agent adapts to its neighbours data representation might yield better estimation performance.

While a theoretical study of any algorithm is of high importance, it would be interesting and refreshing to consider physical implementations. For instance, diffusion LMS could be implemented for different drone applications such as target location in a rescue situation [Câmara, 2014, Rabta et al., 2018] or for modelling a flock behaviour as a way to verify the theoretical models. Physical implementations might raise new challenges. For instance, for a fleet of drones, to preserve energy resources, it is beneficial to limits unnecessary movements rather than radio communication.

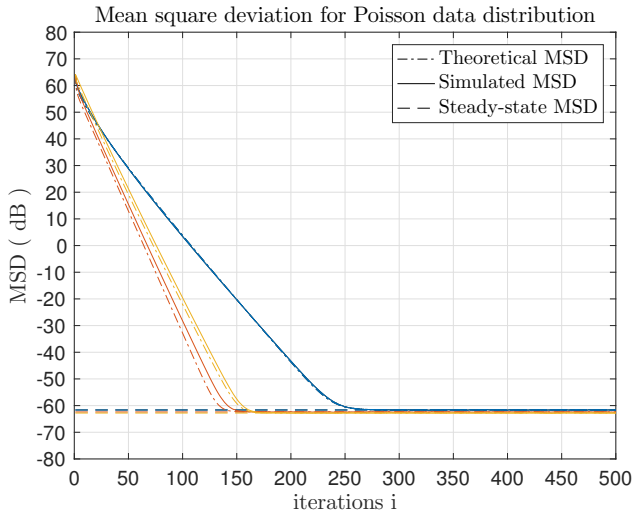

Figure 6.2: Mean Square Error (MSD) using random matrix theory for Poisson data distribution for three covariance matrices. We consider the same setting as for the uniform distribution experiment Figure 6.1. We set the rate parameter λ_p of the distribution to $\lambda_p = 4$ and cantered and normalised the data.

Furthermore, multitask adaptive networks can be used for astrophysics applications such as sky mapping where information is processed on networks clustered on a distance criteria as telescopes in close areas are likely to capture similar signals. Diffusion strategies could also be used in Genomic Signal Processing (GSP) [Anastassiou, 2001]. Indeed, DNA sequences are encoded by four nitrogenated bases: adenine, thymine, cytosine. These bases can be transcribed into a binary code before being numerically processed. One of the most challenging tasks of GSP is finding the relationship between DNA structures and the function of genomic sequences. To do so, it is necessary to find the similarity of DNA sequences. This problem can be formulated as an unsupervised distributed problem where agents need to learn the graph structure based on the similarity with their neighbours.

Evolution tree mapping is also an interesting application. In this case each agent has access to a complete DNA sequence of a species. Agents need to learn the graph structure. However, this time, it is a constrained problem where the graph needs to have a tree structure. This resulting tree is known as the tree of life [Puigbò et al., 2012].

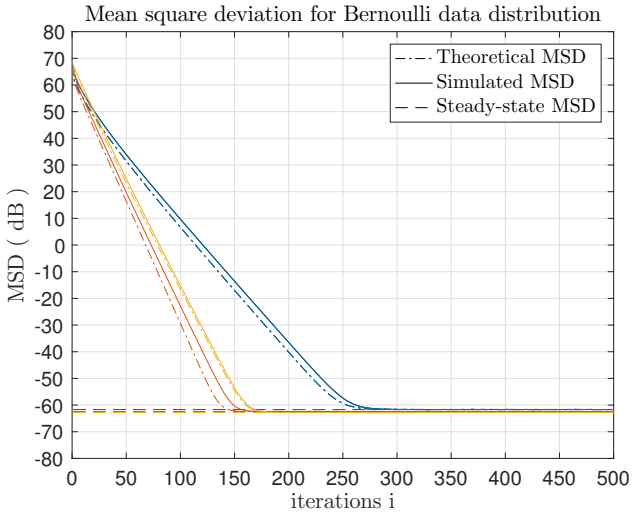

Figure 6.3: Mean square deviation (MSD) using random matrix theory for Bernoulli data distribution for three covariance matrices. We use the same setting as for the previous data distribution. We use two binary values -1 and 1 for the data distribution.

Bibliography

[Abdolee et al., 2014] Abdolee, R., Champagne, B., and Sayed, A. H. (2014). Estimation of space-time varying parameters using a diffusion lms algorithm. *IEEE Transactions on Signal Processing*, 62(2):403–418.

[Abdolee et al., 2016] Abdolee, R., Champagne, B., and Sayed, A. H. (2016). Diffusion adaptation over multi-agent networks with wireless link impairments. *IEEE Transactions on Mobile Computing*, 15(6):1362–1376.

[Agrawal and Srikant, 2000] Agrawal, R. and Srikant, R. (2000). Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450.

[Anastassiou, 2001] Anastassiou, D. (2001). Genomic signal processing. *IEEE signal processing magazine*, 18(4):8–20.

[Arablouei et al., 2014a] Arablouei, R., Doğançay, K., Werner, S., and Huang, Y.-F. (2014a). Adaptive distributed estimation based on recursive least-squares and partial diffusion. *IEEE Transactions on Signal Processing*, 62(14):3510–3522.

[Arablouei et al., 2015] Arablouei, R., Werner, S., Doğançay, K., and Huang, Y.-F. (2015). Analysis of a reduced-communication diffusion LMS algorithm. *Signal Processing*, 117:355–361.

[Arablouei et al., 2014b] Arablouei, R., Werner, S., Huang, Y.-F., and Doğançay, K. (2014b). Distributed least mean-square estimation with partial diffusion. *IEEE Transactions on Signal Processing*, 62(2):472–484.

[Argyriou et al., 2007] Argyriou, A., Evgeniou, T., and Pontil, M. (2007). Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48.

[Argyriou et al., 2008] Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272.

[Bertrand and Moonen, 2010a] Bertrand, A. and Moonen, M. (2010a). Distributed adaptive node-specific signal estimation in fully connected sensor networksâĂŤpart i: Sequential node updating. *IEEE Transactions on Signal Processing*, 58(10):5277–5291.

[Bertrand and Moonen, 2010b] Bertrand, A. and Moonen, M. (2010b). Distributed adaptive node-specific signal estimation in fully connected sensor networksâĂŤpart ii: Simultaneous and asynchronous node updating. *IEEE Transactions on Signal Processing*, 58(10):5292–5306.

[Bertrand and Moonen, 2011] Bertrand, A. and Moonen, M. (2011). Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology. *IEEE Transactions on Signal Processing*, 59(5):2196–2210.

[Bianchi and Jakubowicz, 2013] Bianchi, P. and Jakubowicz, J. (2013). Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, 58(2):391–405.

[Câmara, 2014] Câmara, D. (2014). Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios. In *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*, pages 1–4. IEEE.

[Carmona et al., 2013] Carmona, M., Michel, O., Lacoume, J. L., Sprynski, N., and Nicolas, B. (2013). An analytical solution for the complete sensor network attitude estimation problem. *Signal Processing*, 93(4):652–660.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[Chainais and Richard, 2013] Chainais, P. and Richard, C. (2013). Learning a common dictionary over a sensor network. In *Proc. IEEE Int. Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 1–4, Saint Martin, France.

[Chen et al., 2014a] Chen, J., Richard, C., Hero, A. O., and Sayed, A. H. (2014a). Diffusion lms for multitask problems with overlapping hypothesis subspaces. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE.

[Chen et al., 2014b] Chen, J., Richard, C., and Sayed, A. H. (2014b). Multitask diffusion adaptation over networks. *IEEE Transactions on Signal Processing*, 62(16):4129–4144.

[Chen et al., 2015a] Chen, J., Richard, C., and Sayed, A. H. (2015a). Adaptive clustering for multitask diffusion networks. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 200–204. IEEE.

[Chen et al., 2015b] Chen, J., Richard, C., and Sayed, A. H. (2015b). Diffusion LMS over multitask networks. *IEEE Transactions on Signal Processing*, 63(11):2733–2748.

[Chen et al., 2016] Chen, J., Richard, C., Song, Y., and Brie, D. (2016). Transient performance analysis of zero-attracting lms. *IEEE Signal Processing Letters*, 23(12):1786–1790.

[Chen and Sayed, 2012] Chen, J. and Sayed, A. H. (2012). Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing*, 60(8):4289–4305.

[Chen and Sayed, 2013a] Chen, J. and Sayed, A. H. (2013a). Distributed Pareto optimization via diffusion strategies. *IEEE Journal of Selected Topics in Signal Processing*, 7(2):205–220.

[Chen and Sayed, 2013b] Chen, J. and Sayed, A. H. (2013b). Distributed pareto optimization via diffusion strategies. *IEEE Journal of Selected Topics in Signal Processing*, 7(2):205–220.

[Chen and Liu, 2005] Chen, K. and Liu, L. (2005). Privacy preserving data classification with rotation perturbation. In *Proc. IEEE ICDM*.

[Chen and Liu, 2008] Chen, K. and Liu, L. (2008). A survey of multiplicative perturbation for privacy-preserving data mining. In *Privacy-Preserving Data Mining*, pages 157–181. Springer.

[Chen and Liu, 2011] Chen, K. and Liu, L. (2011). Geometric data perturbation for privacy preserving outsourced data mining. *Knowledge and Information Systems*, 29(3):657–695.

[Chouvardas et al., 2012] Chouvardas, S., Slavakis, K., Kopsinis, Y., and Theodoridis, S. (2012). A sparsity-promoting adaptive algorithm for distributed learning. *IEEE Transactions on Signal Processing*, 60(10):5412–5425.

[Chouvardas et al., 2011] Chouvardas, S., Slavakis, K., and Theodoridis, S. (2011). Adaptive robust distributed learning in diffusion sensor networks. *IEEE Transactions on Signal Processing*, 59(10):4692–4707.

[Couillet et al., 2016] Couillet, R., Benaych-Georges, F., et al. (2016). Kernel spectral clustering of large dimensional data. *Electronic Journal of Statistics*, 10(1):1393–1454.

[Couillet and McKay, 2014] Couillet, R. and McKay, M. (2014). Large dimensional analysis and optimization of robust shrinkage covariance matrix estimators. *Journal of Multivariate Analysis*, 131:99–120.

[Couillet et al., 2015] Couillet, R., Pascal, F., and Silverstein, J. W. (2015). The random matrix regime of maronnaâĂŹs m-estimator with elliptically distributed samples. *Journal of Multivariate Analysis*, 139:56–78.

[Courty et al., 2017] Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2017). Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865.

[Di Lorenzo and Sayed, 2013] Di Lorenzo, P. and Sayed, A. H. (2013). Sparse distributed learning based on diffusion adaptation. *IEEE Transactions on Signal Processing*, 61(6):1419–1433.

[Evgeniou et al., 2005] Evgeniou, T., Micchelli, C. A., and Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(Apr):615–637.

[Evgeniou and Pontil, 2004a] Evgeniou, T. and Pontil, M. (2004a). Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM.

[Evgeniou and Pontil, 2004b] Evgeniou, T. and Pontil, M. (2004b). Regularized multi-task learning. In *Proc. ACM SIGKDD int. Conf. Knowledge Discovery and Data Mining*, Seattle, WA, USA.

[Flamary et al., 2014] Flamary, R., Rakotomamonjy, A., and Gasso, G. (2014). Learning constrained task similarities in graphregularized multi-task learning. *Regularization, Optimization, Kernels, and Support Vector Machines*, 103.

[Friedman, 2011] Friedman, A. (2011). *Privacy preserving data mining*. PhD thesis, Technion-Israel Institute of Technology.

[Gharehshiran et al., 2013] Gharehshiran, O. N., Krishnamurthy, V., and Yin, G. (2013). Distributed energy-aware diffusion least mean squares: Game-theoretic learning. *IEEE Journal of Selected Topics in Signal Processing*, 7(5):1–16.

[Harrane et al., 2016a] Harrane, I. E. K., Flamary, R., and Richard, C. (2016a). Doubly compressed diffusion lms over adaptive networks. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 987–991.

[Harrane et al., 2016b] Harrane, I. E. K., Flamary, R., and Richard, C. (2016b). Toward privacy-preserving diffusion strategies for adaptation and learning over networks. In *Proc. EUSIPCO*, Budapest, Hungary.

[Harrane et al., 2019] Harrane, I. E. K., Flamary, R., and Richard, C. (2019). On reducing the communication cost of the diffusion lms algorithm. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):100–112.

[Hua et al., 2017] Hua, F., Nassif, R., Richard, C., and Wang, H. (2017). Penalty-based multitask estimation with non-local linear equality constraints. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2017 IEEE 7th International Workshop on*, pages 1–5. IEEE.

[Jacob et al., 2009] Jacob, L., Vert, J.-p., and Bach, F. R. (2009). Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, pages 745–752.

[Jain et al., 2011] Jain, P., Kothari, P., and Thakurta, A. (2011). Differentially private online learning. *arXiv preprint arXiv:1109.0105*.

[Kargupta et al., 2005] Kargupta, H., Datta, S., Wang, Q., and Sivakumar, K. (2005). Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information Systems*, 7(4):387–414.

[Khalili et al., 2012a] Khalili, A., Tinati, M. A., Rastegarnia, A., and Chambers, J. A. (2012a). Steady-state analysis of diffusion LMS adaptive networks with noisy links. *IEEE Transaction on Signal Processing*, 60(2):974–979.

[Khalili et al., 2012b] Khalili, A., Tinati, M. A., Rastegarnia, A., and Chambers, J. A. (2012b). Transient analysis of diffusion least-mean squares adaptive networks with noisy channels. *International Journal of Adaptive Control and Signal Processing*, 26(2):171–180.

[Le et al., 2013] Le, T. N., Pegatoquet, A., Berder, O., and Sentieys, O. (2013). Multi-source power manager for super-capacitor based energy harvesting WSN. In *Proc. ACM ENSSys'13*, pages 19:1–19:2, Rome, Italy.

[Liu et al., 2008a] Liu, K., Giannella, C., and Kargupta, H. (2008a). A survey of attack techniques on privacy-preserving data perturbation methods. In *Privacy-Preserving Data Mining*, pages 359–381. Springer.

[Liu et al., 2018] Liu, T., Chen, Z., Zhou, E., and Zhao, T. (2018). Toward deeper understanding of nonconvex stochastic optimization with momentum using diffusion approximations. *arXiv preprint arXiv:1802.05155*.

[Liu et al., 2008b] Liu, W., Pokharel, P. P., and Principe, J. C. (2008b). The kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 56(2):543–554.

[Liu et al., 2012] Liu, Y., Li, C., and Zhang, Z. (2012). Diffusion sparse least-mean squares over networks. *IEEE Transactions on Signal Processing*, 60(8):4480–4485.

[Lopes and Sayed, 2008] Lopes, C. G. and Sayed, A. H. (2008). Diffusion adaptive networks with changing topologies. In *Proc. IEEE ICASSP'08*, pages 3285–3288, Las Vegas, USA.

[Narayan and Peterson, 1981] Narayan, S. S. and Peterson, A. (1981). Frequency domain least-mean-square algorithm. *Proceedings of the IEEE*, 69(1):124–126.

[Nassif et al., 2016a] Nassif, R., Richard, C., Chen, J., Ferrari, A., and Sayed, A. H. (2016a). Diffusion LMS over multitask networks with noisy links. In *Proc. IEEE ICASSP*.

[Nassif et al., 2017a] Nassif, R., Richard, C., Chen, J., and Sayed, A. H. (2017a). A graph diffusion lms strategy for adaptive graph signal processing. In *Signals, Systems, and Computers, 2017 51st Asilomar Conference on*, pages 1973–1976. IEEE.

[Nassif et al., 2015] Nassif, R., Richard, C., Ferrari, A., and Sayed, A. H. (2015). Multitask diffusion lms with sparsity-based regularization. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 3516–3520. IEEE.

[Nassif et al., 2016b] Nassif, R., Richard, C., Ferrari, A., and Sayed, A. H. (2016b). Distributed learning over multitask networks with linearly related tasks. In *Signals, Systems and Computers, 2016 50th Asilomar Conference on*, pages 1390–1394. IEEE.

[Nassif et al., 2016c] Nassif, R., Richard, C., Ferrari, A., and Sayed, A. H. (2016c). Multitask diffusion adaptation over asynchronous networks. *IEEE Transactions on Signal Processing*, 64(11):2835–2850.

[Nassif et al., 2016d] Nassif, R., Richard, C., Ferrari, A., and Sayed, A. H. (2016d). Proximal multitask learning over networks with sparsity-inducing coregularization. *IEEE Transactions on Signal Processing*, 64(23):6329–6344.

[Nassif et al., 2016e] Nassif, R., Richard, C., Ferrari, A., and Sayed, A. H. (2016e). Proximal multitask learning over networks with sparsity-inducing coregularization. *IEEE Transactions on Signal Processing*, 64(23):6329–6344.

[Nassif et al., 2017b] Nassif, R., Richard, C., Ferrari, A., and Sayed, A. H. (2017b). Diffusion lms for multitask problems with local linear equality constraints. *IEEE Transactions on Signal Processing*, 65(19):4979–4993.

[Necoara et al., 2017] Necoara, I., Nesterov, Y., and Glineur, F. (2017). Random block coordinate descent methods for linearly constrained optimization over networks. *Journal of Optimization Theory and Applications*, 173(1):227–254.

[Nedic and Ozdaglar, 2009] Nedic, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Autom. Control*, 54(1):48–61.

[Parreira et al., 2012] Parreira, W. D., Bermudez, J. C. M., Richard, C., and Tourneret, J.-Y. (2012). Stochastic behavior analysis of the gaussian kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 60(5):2208–2222.

[Plata-Chaves et al., 2016] Plata-Chaves, J., Bahari, M. H., Moonen, M., and Bertrand, A. (2016). Unsupervised diffusion-based lms for node-specific parameter estimation over wireless sensor networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4159–4163. IEEE.

[Plata-Chaves et al., 2015] Plata-Chaves, J., Bogdanović, N., and Berberidis, K. (2015). Distributed diffusion-based lms for node-specific adaptive parameter estimation. *IEEE Transactions on Signal Processing*, 63(13):3448–3460.

[Puigbò et al., 2012] Puigbò, P., Wolf, Y. I., and Koonin, E. V. (2012). Genome-wide comparative analysis of phylogenetic trees: the prokaryotic forest of life. In *Evolutionary Genomics*, pages 53–79. Springer.

[Rabta et al., 2018] Rabta, B., Wankmüller, C., and Reiner, G. (2018). A drone fleet model for last-mile distribution in disaster relief operations. *International Journal of Disaster Risk Reduction*, 28:107–112.

[Rajkumar and Agarwal, 2012] Rajkumar, A. and Agarwal, S. (2012). A differentially private stochastic gradient descent algorithm for multiparty classification. In *International Conference on Artificial Intelligence and Statistics*, pages 933–941.

[Rakotomamonjy et al., 2011] Rakotomamonjy, A., Flamary, R., Gasso, G., and Canu, S. (2011). ℓ_p - $\ell - q$ penalty for sparse linear and sparse multiple kernel multitask learning. *IEEE Transactions on Neural Networks*, 22(8):1307–1320.

[Ramakrishnan et al., 2001] Ramakrishnan, N., Keller, B. J., Mirza, B. J., Grama, A. Y., and Karypis, G. (2001). Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54.

[Sayed, 2003] Sayed, A. H. (2003). *Fundamentals of adaptive filtering.* J. Wiley & Sons, Hoboken, NJ.

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

[Sayed, 2013b] Sayed, A. H. (2013b). Diffusion adaptation over networks. *Academic Press Library in Signal Processing*, 3:323–454.

[Sayed, 2014] Sayed, A. H. (2014). Adaptation, learning, and optimization over networks. In *Foundations and Trends in Machine Learning*, volume 7, pages 311–801. NOW Publishers, Boston-Delft.

[Sayed, 2013a] Sayed, A. H. (2014. Also available as arXiv:1205.4220 [cs.MA], May 2013a). Diffusion adaptation over networks. In Chellapa, R. and Theodoridis, S., editors, *Academic Press Libraray in Signal Processing*, pages 322–454. Elsevier.

[Sayed et al., 2013] Sayed, A. H., Tu, S.-Y., Chen, J., Zhao, X., and Towfic, Z. J. (2013). Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior. *IEEE Signal Processing Magazine*, 30(3):155–171.

[Sayin and Kozat, 2014] Sayin, M. O. and Kozat, S. S. (2014). Compressive diffusion strategies over distributed networks for reduced communication load. *IEEE Transactions on Signal Processing*, 62(20):5308–5323.

[Szurley et al., 2015] Szurley, J., Bertrand, A., and Moonen, M. (2015). Distributed adaptive node-specific signal estimation in heterogeneous and mixed-topology wireless sensor networks. *Signal Processing*, 117:44–60.

[Takahashi et al., 2010] Takahashi, N., Yamada, Y., and Sa (2010). Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis. *IEEE Trans. Signal Processing*, 58(9):4795–4810. Diffusion LMS 3: more general, adpative combination.

[Tu and Sayed, 2010] Tu, S.-Y. and Sayed, A. H. (2010). Foraging behavior of fish schools via diffusion adaptation. In *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*, pages 63–68. IEEE.

[Vadidpour et al., 2015] Vadidpour, V., Rastegarnia, A., Khalili, A., and Sanei, S. (2015). Partial-diffusion least mean-square estimation over networks under noisy information exchange. *arXiv preprint arXiv:1511.09044*.

[Verykios et al., 2004] Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y. (2004). State-of-the-art in privacy preserving data mining. *ACM Sigmod Record*, 33(1):50–57.

[Vincent et al., 2014] Vincent, R., Carmona, M., Michel, O., and Lacoume, J.-L. (2014). A lower bound for passive sensor-network auto-localization. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 1965–1969. IEEE.

[Wang et al., 2018] Wang, C., Zhang, Y., Ying, B., and Sayed, A. H. (2018). Coordinate-descent diffusion learning by networked agents. *IEEE Transactions on Signal Processing*, 66(2):352–367.

[Wen and Liu, 2015] Wen, F. and Liu, W. (2015). Diffusion least mean square algorithms with zero-attracting adaptive combiners. In *Proc. IEEE CIT–IUCC–DASC–PICOM'15*, pages 252–256.

[Xi and Khan, 2017] Xi, C. and Khan, U. A. (2017). Distributed subgradient projection algorithm over directed graphs. *IEEE Transactions on Automatic Control*, 62(8):3986–3992.

[Yuan and Lin, 2006] Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

[Zeng and Yin, 2018] Zeng, J. and Yin, W. (2018). On nonconvex decentralized gradient descent. *IEEE Transactions on signal processing*, 66(11):2834–2848.

[Zhao and Sayed, 2015a] Zhao, X. and Sayed, A. H. (2015a). Asynchronous adaptation and learning over networksâĂŤpart i: Modeling and stability analysis. *IEEE Transactions on Signal Processing*, 63(4):811–826.

[Zhao and Sayed, 2015b] Zhao, X. and Sayed, A. H. (2015b). Asynchronous adaptation and learning over networksâĂŤpart ii: Performance analysis. *IEEE Transactions on Signal Processing*, 63(4):827–842.

[Zhao et al., 2012] Zhao, X., Tu, S.-Y., and Sayed, A. H. (2012). Diffusion adaptation over networks under imperfect information exchange and non-stationary data. *IEEE Transactions on Signal Processing*, 60(7):3460–3475.