DANIEL GILIO TIGLEA

Can adaptive diffusion networks do better with less data?

São Paulo
2024

DANIEL GILIO TIGLEA

Can adaptive diffusion networks do better with less data?

Corrected Version

Doctoral Thesis presented to the Polytechnic School of the University of São Paulo to obtain the degree of Doctor of Science.

São Paulo
2024

Nome: TIGLEA, Daniel G.
Título: Can adaptive diffusion networks do better with less data?
Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Ciências.

Aprovado em: 19/06/2024

Banca Examinadora

DANIEL GILIO TIGLEA

Can adaptive diffusion networks do better with less data?

Corrected Version

Doctoral Thesis presented to the Polytechnic School of the University of São Paulo to obtain the degree of Doctor of Science.

Concentration area:
Electronic Systems

Supervisors:
Prof. Dr. Magno Teófilo Madeira da Silva
Dr. Renato Candido

São Paulo
2024

Catalogação-na-publicação

Tiglea, Daniel Gilio
Can adaptive diffusion networks do better with less data? / D. G. Tiglea -- versão corr. -- São Paulo, 2024.
194 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Sistemas distribuídos 2.Filtros elétricos adaptativos 3.Redes de computadores (Sensores) 4.Processamento de sinais (Amostragem) 5.Amostragem em soluções distribuídas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

To my parents,
Paulo and Virgínia

ACKNOWLEDGMENTS

I would like to thank my advisors, Drs. Magno Silva and Renato Candido, without whom I would never have made it this far. Their technical advice was crucial to the elaboration of this dissertation, and to my academic training. I am in their debt because of this, and I sincerely thank them for this. Perhaps even more importantly, after more than six years of everyday interactions with Drs. Magno Silva and Renato Candido, during which they were always very supportive and patient with me, I dare thank them for their *friendship* as well.

I would also like to thank Professors Denis Fantinato and Wallace Alves Martins for their suggestions during the qualification exam. In fact, their recommendations have somehow helped shape this dissertation even before I started my PhD, due to their insightful comments on my Master's thesis. Also, many thanks to prof. Luiz A. Azpicueta-Ruiz for his collaboration and helpful comments, which have also enriched this work.

I also thank the Coordination of Superior Level Staff Improvement (CAPES) for the financial support for this work (project 88887.512247/2020-00).

I am also grateful to a handful of friends, who contributed to this work not through technical advice, but through moments of relaxation that made this journey (and my life as a whole) more pleasant. Thus, my special thanks to Amanda Macedo, Camila Zilio, Felipe Ribeiro, Leonardo Lopes, Leonardo Macedo, Luis Felipe and Luis Henrique Fabbri, Mariana Rabello, Rafael Orsoni, Rodrigo Martins, and Rodrigo Scolaro.

I would also like to thank my girlfriend, Bruna, from the bottom of my heart, for her patience, support, love, and care throughout these years.

Finally, I would also like to thank my parents, Paulo and Virginia, for their love and support, for teaching me so much throughout my life, and for instilling in me the appreciation for Science. It is not that I would never have finished this journey without them. I never would have even started it.

ABSTRACT

TIGLEA, D. G. **Can adaptive diffusion networks do better with less data?** 2024. Thesis (Doctor in Electric Engineering) – Polytechnic School of the University of São Paulo, São Paulo, 2024.

Adaptive diffusion networks consist of a collection of agents that can measure and process locally streaming data, and that can cooperate with one another to improve the overall performance. Since their inception, these networks have consolidated themselves as interesting tools for distributed estimation and learning, and have spun several types of solutions for these problems. To reduce the amount of data measured, processed, and transmitted over these networks, several techniques have been proposed in the literature, which usually affect the performance of the original solutions, but are necessary in order to extend the network lifetime. In this work, in addition to an extensive literature review, we present sampling techniques that eliminate the need to measure and process the data at every node and every time instant. By controlling the sampling of the nodes based on their estimation error, the proposed techniques are able to maintain the convergence rate of the original solutions, while achieving a lower computational cost and better performance in the steady state. This comes at the expense of only a slight increase in the computational cost during the transient phase in comparison with that of the original solutions. Moreover, with slight modifications, the techniques presented can also be used to restrict the number of transmissions between the nodes in the network. Lastly, we conduct a theoretical analysis in order to understand the performance of the proposed solutions, which agrees with the simulation results.

Keywords: Adaptive networks. Distributed estimation. Nonlinear adaptive filtering. Graph filtering. Sampling. Energy efficiency.

RESUMO

TIGLEA, D. G. **Redes de difusão adaptativas podem se sair melhor com menos dados?** 2024. Tese (Doutorado em Engenharia Elétrica) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2024.

As redes de difusão adaptativa consistem em um conjunto de agentes capazes de medir e processar localmente dados em tempo real e de cooperar entre si para melhorar o desempenho geral. Desde o seu surgimento, essas redes se consolidaram como ferramentas interessantes para estimação e aprendizagem distribuída e deram origem a diversos tipos de soluções para esses problemas. A fim de reduzir a quantidade de dados medidos, processados e transmitidos nessas redes, diversas técnicas foram propostas na literatura. Frequentemente, elas afetam o desempenho das soluções originais, mas são necessárias para prolongar a vida útil da rede. Neste trabalho, além de uma extensa revisão bibliográfica, são apresentadas técnicas de amostragem que eliminam a necessidade de medir e processar os dados em todos os nós a cada instante de tempo. Ao controlar a amostragem dos nós com base no seu erro de estimação, as técnicas propostas são capazes de manter a taxa de convergência das soluções originais, ao mesmo tempo em que alcançam menor custo computacional e melhor desempenho no regime permanente. Isso ocorre às custas apenas de um ligeiro aumento no custo computacional durante o transitório em comparação com o das soluções originais. Além disso, com pequenas modificações, as técnicas apresentadas também podem ser utilizadas para restringir o número de transmissões entre os nós da rede. Por último, é apresentada uma análise teórica para compreender o desempenho das soluções propostas, que concorda com os resultados de simulações.

Palavras-chave: Redes adaptativas. Estimação distribuída. Filtragem adaptativa não linear. Filtragem em grafos. Amostragem. Eficiência energética.

LIST OF ACRONYMS

ACW	*adaptive combination weights*
APA	*affine projection algorithm*
AS	*adaptive-sampling*
ASC	*adaptive-sampling-and-censoring*
ATC	*adapt-then-combine*
cdf	*cumulative density function*
CTA	*combine-then-adapt*
dAPA	*diffusion APA*
dGLMS	*diffusion Graph LMS*
dKLMS	*diffusion kernel LMS*
dLMS	*diffusion LMS*
dNLMS	*diffusion NLMS*
dRLS	*diffusion RLS*
DTAS	*Dynamic-Tuning AS*
DTRAS	*Dynamic-Tuning-and-Resetting AS*
FIR	*finite impulse response*
GSP	*graph signal processing*
i.i.d.	*independent and identically distributed*
IoT	*internet of things*
LMS	*least mean squares*
MSD	*mean square deviation*
MSE	*mean square error*
NEMSE	*network excess MSE*

NLMS	*normalized least mean squares*
NMSD	*network MSD*
NMSE	*network MSE*
pdf	*probability density function*
rhs	*right-hand side*
RFF	*random Fourier features*
RLS	*recursive least squares*
SNR	*signal-to-noise ratio*
VSS	*variable step size*
WSN	*wireless sensor network*

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Notation. We use lowercase normal font letters to denote scalars, boldface lowercase letters for vectors, boldface uppercase letters for matrices, and calligraphic fonts for sets. To simplify the arguments, we assume real data throughout this work. Next, we provide a list of the symbols used in this work.

General Symbols

$(\cdot)^{\mathrm{T}}$	transposition operator
$(\cdot)^{*}$	complex conjugate
$\mathrm{E}\{\cdot\}$	mathematical expectation
$\sum$	sum
$\|\cdot\|$	Euclidean norm
$\lvert\cdot\rvert$	cardinality if the argument is a set, or absolute value if the argument is a scalar
$\mathbf{I}_N$	$N \times N$ identity matrix
$\mathbf{0}_N$	$N \times 1$ vector whose entries are all equal to 0
$\mathbf{1}_N$	$N \times 1$ vector whose entries are all equal to 1
$\mathbf{0}_{N\times M}$	$N \times M$ matrix whose entries are all equal to 0
$\mathbf{1}_{N\times M}$	$N \times M$ matrix whose entries are all equal to 1
$[\mathbf{X}]_{ij}$	(i,j)-th entry of the matrix $\mathbf{X}$
$\otimes$	Kronecker product, or number of multiplications per iteration
$\oplus$	number of sums per iteration
$\odot$	Hadamard product
$\mathcal{X} \cap \mathcal{Y}$	intersection between the sets $\mathcal{X}$ and $\mathcal{Y}$
$\mathcal{X} \backslash \mathcal{Y}$	difference between the sets $\mathcal{X}$ and $\mathcal{Y}$
min	minimum of a finite set or compact function

$\max$	maximum of a finite set or compact function
Δ	difference or variation
$\arg\min$	arguments of the minima
$\mathrm{sgm}(\cdot)$	sigmoid function
$\exp(\cdot)$	exponential function
$\mathrm{erf}(\cdot)$	error function
$\mathrm{erf}^{-1}(\cdot)$	inverse error function
$\mathrm{erfc}(\cdot)$	complementary error function
$\mathrm{col}\{\cdot\}$	column vector formed by the stacking of the arguments in their respective order
$\mathrm{diag}\{\cdot\}$	diagonal matrix formed by the aggregation of the arguments in their respective order
$\mathrm{vec}\{\cdot\}$	column vector formed by the stacking of the columns of a matrix argument on top of each other in their respective order
∇	gradient
$\lambda_{\max}(\cdot)$	maximum absolute value of the eigenvalues of a matrix
$\lambda_i(\cdot)$	i-th greatest eigenvalue of a matrix
$\rho(\cdot)$	spectral radius of a matrix
$\widehat{x}$	estimate of the true value of the variable x
$\Pr(\cdot)$	probability of an event
p	probability of success of a Bernoulli random variable
$f_X(x)$	pdf of a random variable
$F_X(x)$	cdf of a random variable
$g_X(x)$	Gaussian pdf
$G_X(x)$	Gaussian cdf
$\lceil\cdot\rceil$	ceiling function

$\lfloor \cdot \rfloor$	floor function

Distributed signal processing

n	time instant
V	number of nodes in an adaptive diffusion network
e	estimation error
d	desired signal
v	additive noise
u	input signal
$\mathbf{u}$	input regressor vector
μ	adaptation step
M	number of taps of the algorithm
$\mathbf{w}^{\mathrm{o}}$	coefficient vector of the unknown system in single-task scenarios
J	cost function
$\boldsymbol{\psi}$	local estimate of $\mathbf{w}^{\mathrm{o}}$ produced by the diffusion algorithms
$\mathbf{w}$	combined estimate of $\mathbf{w}^{\mathrm{o}}$ produced by the diffusion algorithms
$\mathcal{N}_k$	neighborhood of a given node k
c_{ik}	combination weight assigned by node k to the estimate received from neighbor i
b_{ik}	combination weight assigned by node k to the gradient received from neighbor i
$\mathbf{R}_u$	autocorrelation matrix of the input signal u
$\widetilde{\mathbf{w}}$	difference between the combined estimate $\mathbf{w}$ and the true coefficient vector to be estimated
$\widetilde{\boldsymbol{\psi}}$	difference between the combined estimate $\boldsymbol{\psi}$ and the true coefficient vector to be estimated
ν	forgetting factor
σ_v^2	variance of the additive noise

$\hat{\sigma}^2_{ik}$	estimate at node k of variance of the additive noise at node i
ζ	binary variable that determines the sampling and/or the censoring of each node
$\mathbf{w}^{\mathrm{o}}_k$	coefficient vector of the unknown system at node k in multitask scenarios
$C(k)$	cluster to which the node k belongs
C_i	i-th cluster in the network
$\mathbf{w}^{\mathrm{o}}_{C_i}$	coefficient vector of the unknown system at the nodes of cluster C_i in clustered multitask scenarios
η	regularization parameter of clustered multitask adaptive diffusion networks
$\bar{\rho}_{ki}$	combination weights for adjusting the regularization strength between two nodes i and k in a clustered multitask scenario
e_{ik}	estimation error obtained with the desired signal and input regressor vector of node i, using the combined estimate of node k
$\varphi^{\mathrm{o}}(\cdot)$	nonlinear transformation of the input vector $\mathbf{u}$
$\kappa(\cdot,\cdot)$	Mercer kernel
h	Gaussian kernel bandwidth
$\mathscr{D}$	dictionary of a kernel-based method
D	number of elements in a dictionary
$\boldsymbol{\kappa}$	vector comprised of the kernel values computed between the input vector $\mathbf{u}$ and the elements of the dictionary $\mathscr{D}$
$\boldsymbol{\omega}$	vector of random frequencies in RFF methods
θ	random phase in RFF methods
$\mathbf{z}$	mapped input vector of RFF methods
$\mathbf{A}$	adjacency matrix of a graph
$\mathbf{H}$	linear shift-invariant graph filter
$\mathbf{h}$	output of a linear shift-invariant graph filter

$\mathbf{x}$	input vector of adaptive diffusion networks for GSP
δ_r	regularization factor to avoid division by zero when normalizing a step size
$\widetilde{\mu}$	normalized step size
δ_c	regularization factor to avoid division by zero in the ACW algorithm
β	parameter used to control the penalization of the sampling of the nodes in the AS-dNLMS algorithm
$\bar{\zeta}$	auxiliary variable used in the derivation of the AS-dNLMS algorithm
α	auxiliary variable used in the adaptation of $\zeta(n)$
α^+	greatest value that α can assume
$\phi(\cdot)$	sigmoid function that establishes the relationship between the variables $\bar{\zeta}$ and α
$\phi'(\cdot)$	derivative of ϕ with respect to α
ε	last measurement of e that we have access to
μ_ζ	step size used in the adaptation of α
μ_ζ	step size used in the adaptation of α
$\sigma^2_{\max}$	maximum noise variance in the network
$\sigma^2_{\min}$	minimum noise variance in the network
β_r	ratio between β and $\sigma^2_{\max}$
$\breve{\eta}$	number of iterations during each duty cycle of the sampling mechanism in which the node is sampled
$\bar{\eta}$	number of iterations during each duty cycle of the sampling mechanism in which the node is not sampled
V_s	number of nodes sampled in the network
V_t	number of transmitting nodes in the network
ϕ'_0	ϕ' evaluated at $\alpha = 0$
ϕ'_{α^+}	ϕ' evaluated at $\alpha = \alpha^+$

Δn	parameter of the sampling algorithms, used to control when the nodes cease to be sampled
$\mathbf{q}$	random column vector used to model the changes in the optimal system in a random-walk scenario
$\mathbf{Q}$	autocovariance matrix of $\mathbf{q}$
γ	parameter used to control the number of sampled nodes in the DTAS-dNLMS, DTRAS-dNLMS, and DTASC-RFF-dKNLMS algorithms
$\widehat{\sigma}_{\mathcal{N}}^2$	estimate of the average noise power in the neighborhood of a node
$\widehat{\sigma}_f^2$	fast estimate of the noise power calculated by the algorithm of [265]
$\widehat{\sigma}_m^2$	medium-speed estimate of the noise power calculated by the algorithm of [265]
$\widehat{\sigma}_s^2$	slow estimate of the noise power calculated by the algorithm of [265]
$\widehat{\sigma}_\iota^2$	intermediate estimate of the noise power calculated by the algorithm of [265]
$\widehat{\sigma}_v^2$	final estimate of the noise power calculated by the algorithm of [265]
χ	sensitivity threshold of the reset mechanism of DTRAS-dNLMS
$\mathbf{U}$	input matrix of the diffusion Affine Projection Algorithm
$\mathbf{d}$	desired vector of the diffusion Affine Projection Algorithm
$\mathbf{e}$	estimation error vector of the diffusion Affine Projection Algorithm
$\breve{e}$	estimation error of the RFF-dKNLMS algorithm
$\breve{\varepsilon}$	last measurement of $\breve{e}$ that we have access to
$H(z)$	transfer function of a discrete-time linear time-invariant system
z^{-1}	unit delay operator
σ_u^2	variance of the input signal
$\mathbf{C}$	matrix formed from the aggregation of the combination weights c_{ik}
$\mathbf{R}_v$	diagonal matrix whose i-th element is equal to the noise variance at node i
χ_{LMS}	steady-state MSD of a single LMS filter

χ_{nc}	steady-state NMSD of a noncooperative dLMS network
K_V	network topology in which there is a link between every pair of nodes
χ_{K_V}	steady-state NMSD of a dLMS network arranged according to a K_V topology
$\bar{\sigma}_v^2$	average noise power in the network
$n_{\text{s.s.}}$	predicted iteration at which the DTRAS-dNLMS algorithm achieves the steady state in terms of the NMSE
n_{switch}	predicted iteration at which the sampling probability of DTRAS-dNLMS switches from one to its steady-state value

TABLE OF CONTENTS

1 INTRODUCTION

In this chapter, we seek to motivate and contextualize the present work, to present its main contributions, and to explain how it is organized. In Sec. 1.1, we provide the motivation for this dissertation. In Secs. 1.2 and 1.3, respectively, we state the goals and the justification for the current work. Lastly, in Secs. 1.4 and 1.5 we present the main contributions of our work and the structure of this dissertation.

1.1 Motivation

Over the past two decades, adaptive diffusion networks have consolidated themselves as an interesting tool for distributed signal processing [1–6]. These networks consist of a set of connected *agents* or *nodes*, capable of collecting data, performing calculations locally and communicating with other nearby agents, called *neighbors*. The objective of the network as a whole is to estimate a vector of parameters of interest. For this, each node calculates its own *local* estimate in the so-called *adaptation step*. Then, in the *combination step*, neighboring nodes cooperate to reach a *combined* estimate of the vector of interest. The order in which these steps are performed leads to two possible schemes: "adapt-then-combine" (ATC) and "combine-then-adapt" (CTA) strategies. With these two steps, the idea is to estimate the parameters of interest without having a central processing unit [1–6].

Compared to centralized approaches, which require a central unit to receive and process data from the entire network, this type of solution offers better scalability, autonomy and flexibility [1–6]. Moreover, they also present an improved robustness in comparison with other distributed approaches, such as the incremental [7–10] and consensus [11–17] strategies. Consequently, adaptive diffusion networks are considered effective solutions in various applications, such as target localization and tracking [1], spectral sensing in mobile networks [1, 18], medical applications [19], among others. Moreover, due to their popularity, they have branched out into many research topics, such as multitask networks [20–31], nonlinear adaptive networks [32–40], among others. Moreover, the field of graph signal processing (GSP) has often been inspired by these techniques, since it deals with applications that are usually distributed in nature [41–43, 47, 48]. As a result, many graph adaptive filtering algorithms can be seen as an extension of adaptive diffusion networks to domains where space, as well as time, plays a role in the development of the signals of interest. It is worth noting that, in this work, we are primarily concerned with adaptive diffusion networks designed for distributed linear and nonlinear adaptive filtering, both in the traditional sense and for GSP. In recent years, the term "adaptive networks" has sometimes been employed in a wider sense to refer to networked strategies for, e.g., optimization [49, 50] and social learning [51–53]. However, these topics are out of the scope of the current work.

In the implementation of solutions for distributed signal processing, it is oftentimes desirable to reduce the amount of data measured, processed, and transmitted throughout the network. For example, when dealing with wireless sensor networks (WSNs), energy consumption is often the most critical constraint, especially if devices that run on batteries are employed. Since the communication between the nodes can be particularly draining in this regard, several techniques were proposed over the years to reduce the energy consumption associated with the exchange of information [54–82]. Among these, there is a group of solutions known as *censoring* techniques. They seek to cut the transmission from certain nodes to any of their neighbors [72–82], hence allowing censored nodes to turn their transmitters off. The search for efficient mechanisms that reduce the energy consumption associated with the communication between nodes while preserving the performance of adaptive diffusion networks is a topic that continues to inspire the distributed signal processing community to this day [60–62, 78, 79, 81–84].

Furthermore, in certain situations, the cost of measuring and processing the data available at each node at each iteration can be prohibitively high. In these cases, it is necessary to employ *sampling* techniques [42–46], which can significantly reduce the computational and memory cost associated with learning.

For the reasons explained above, sampling and censoring techniques are often necessary for the feasibility of adaptive diffusion networks. However, as one might expect, they can also have a negative effect on the network performance [42, 43, 72–82]. Thus, in this work, our aim is to derive sampling and censoring techniques that negatively impact the performance as little as possible. Moreover, we seek to analyze the impact of sampling, so as to understand in detail its effects on the network performance.

1.2 Objectives

The main objectives of this work are as follows.

1. To carry out a literature review on adaptive diffusion networks, in order to understand the different solutions that have been proposed in the area;
2. to propose adaptive sampling and censoring mechanisms, in order to obtain efficient algorithms for adaptive diffusion networks in terms of computational cost and energy consumption;
3. to test the proposed algorithms, considering different types of scenarios and practical applications, when feasible;
4. to perform a theoretical analysis of the performance of the proposed algorithms and to understand the impact of the sampling of the nodes on the performance.

1.3 Justification

With the deployment of 5G communication networks [85], the implementation of Internet of Things (IoT) has been facilitated in the past few years, a trend that is expected to continue in the near future [86, 87]. Furthermore, investigations on 6G communication networks have begun [88–92], and there is an expectation that they may come to fruition in the second half of this decade or in the early 2030's [89]. In addition to these development, there has been an increase in the usage of WSNs in practical applications. For instance, wireless body area networks (WBANs) [93, 94] have become appealing solutions in, e.g., healthcare and security applications [95, 96]. Moreover, WSNs have also been employed in agriculture [97], speech, video and image enhancement, spectrum sensing, power system state estimation, among others [98].

As a results of these advances, we may expect to see an increased applicability of adaptive diffusion networks in the near future [99, 100]. In this context, computational and energy efficiency will be crucial for the success of these solutions. However, many of the existing censoring and sampling techniques, although praiseworthy and interesting in their own right, usually affect the network performance in a noticeable manner [42, 43, 72–82].

Hence, obtaining a technique that leads to energy and computational cost savings while preserving performance is a relevant issue and a matter of practical interest. This would make it possible to implement adaptive diffusion networks without sacrificing the characteristics that make them so attractive.

1.4 Contributions

Based on the results presented in this dissertation, we have published four journal papers [JP-1, JP-2, JP-3, JP-4], two international conference papers [IC-1 and IC-2], and five Brazilian national conference papers [NC-1, NC-2, NC-3, NC-4 and NC-5]. It is worth mentioning that the paper [IC-1] was ranked in the top 3% of the papers presented at the 2023 edition of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), and received a certificate for it. Moreover, we have also submitted another paper [SP] to the *IEEE Transactions on Signal Processing*, which is currently under review.

[JP-1] D. G. Tiglea, R. Candido, and M. T. M. Silva, "A low-cost algorithm for adaptive sampling and censoring in diffusion networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 58–72, Jan. 2021. Available: <https://ieeexplore.ieee.org/abstract/document/9257199?casa_token=E9aAA4dv8wAAAAAA:VJpn52pGlAKte8seaLZbU1FDqBlwNqYuZSLPCPJNUY_OznSQBPzlClpzCD7-Mx4DrtTYA_BaHg>

[JP-2] D. G. Tiglea, R. Candido, and M. T. M. Silva, "An adaptive algorithm for sampling over diffusion networks with dynamic parameter tuning and change detection mechanisms," *Digital Signal Processing*, vol. 127, pp. 103587, Jul. 2022. Available: <https://www.sciencedirect.com/science/article/pii/S1051200422002044?casa_token=SkoETIl9rx8AAAAA:WIN9u9HxnjKaqjFq37XxidLTUasQG8fxkXP70N_HfDav9cALaHWjpFb2bZ0Hu7-iyzdsMmIn2g>

[JP-3] D. G. Tiglea, R. Candido, and M. T. M. Silva, "A variable step size adaptive algorithm with simple parameter selection," *IEEE Signal Processing Letters*, vol. 29, pp. 1774–1778, Aug. 2022. Available: <https://ieeexplore.ieee.org/abstract/document/9847065?casa_token=w1LHgphtsZ0AAAAA:RcVf0FvnI6GSssXDWpcdPE-kjuYl5R8CJttIPHy_-G6JD7jv0vYnV-XdKUL_M8HHTvezda2PiA>

[JP-4] D. G. Tiglea, R. Candido, and M. T. M. Silva, "Adaptive Diffusion Networks: An

Overview,"*Signal Processing*, vol. 223, p. 109570, Oct. 2024. Available online since June 2024 on: <https://www.sciencedirect.com/science/article/pii/S0165168424001890?casa_token=u4B3BmSChwQAAAAA:R91SOTlDokLZJ_4Pr9rvQBlqtYdwl63IW8NwZkFtXLnhxZ9AdokLqe7pPIJ2h1uQGjs1jeqRlw>

[IC-1] D. G. Tiglea, R. Candido, L. A. Azpicueta-Ruiz, and M. T. M. Silva, "Reducing the communication and computational cost of random Fourier features kernel LMS in diffusion networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5. Available: <https://ieeexplore.ieee.org/abstract/document/10095416?casa_token=2CjvtrQKqawAAAAA:6YYgCmHOF7nXUkFUcsAhXRLH3qZonSgYvxQWPgYnq5JUxb3WwSi1DkSRw5Zc2bnoYI0jWd28dQ>

[IC-2] D. G. Tiglea, R. Candido, and M. T. M. Silva, "Can Adaptive Diffusion Networks Do Better with Less Data?" Accepted for publication in *Proc. 19th International Symposium on Wireless Communication Systems*. To appear.

[NC-1] D. G. Tiglea, R. Candido, and M. T. M. Silva, "Adaptive sampling for diffusion networks with online parameter adjustment," (in Portuguese), in *Anais do XXXIX Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, 2021, pp. 1–5. Available: <https://biblioteca.sbrt.org.br/articles/2839>

[NC-2] A. A. Bueno, D. G. Tiglea, R. Candido, and M. T. M. Silva, "A kernel algorithm based on Gram-Schmidt orthogonalization for adaptive diffusion networks," (in Portuguese) in *Anais do XXXIX Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, 2021, p. 1–5. Available: <https://biblioteca.sbrt.org.br/articles/2897>

[NC-3] L. Gamballi, D. G. Tiglea, R. Candido, and M. T. M. Silva, "Distributed training of neural networks for geometric figures classification," (in Portuguese) in *Anais do XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, 2022, pp.1–3. Available: <https://biblioteca.sbrt.org.br/articles/3599>

[NC-4] L. Gamballi, D. G. Tiglea, R. Candido, and M. T. M. Silva, "Distributed MLP networks for cardiac arrhythmia classification," (in Portuguese) in *Anais do XL Brasileiro de Telecomunicações e Processamento de Sinais*, 2022, pp.1–5. Available: <https://biblioteca.sbrt.org.br/articles/3600>

[NC-5] D. G. Tiglea, R. Candido, and M. T. M. Silva, "A sampling algorithm for multitask adaptive diffusion networks," (in Portuguese), in *Anais do XLI Brasileiro de Telecomunicações e Processamento de Sinais*, 2023, pp.1–5. Available: <https://biblioteca.sbrt.org.br/articles/4466>

[SP] D. G. Tiglea, R. Candido, and M. T. M. Silva, "On the Impact of Random Node Sampling on Adaptive Diffusion Networks." Submitted to the *IEEE Transactions on Signal Processing* on 12th March 2024.

The main contributions of this work are listed next.

1. **Extensive literature review**.

   In this work, we carry out a literature review that analyzes the evolution of adaptive diffusion networks since their inception in the mid 2000's, and covers many of the solutions that have branched out from these techniques, such as kernel-based and multitask adaptive diffusion networks, as well as adaptive diffusion networks for GSP. We also examine the technological developments that led to the emergence of adaptive diffusion networks and to the prolonged interest in them. This review is presented in Chapter 2.

2. **Adaptive Sampling and Censoring Algorithms**.

   We present adaptive algorithms for the sampling and censoring of diffusion networks, which aim to reduce the computational cost and energy consumption of these solutions without negatively affecting their performance. These algorithms are presented in Chapter 3, where we also analyze the number of nodes sampled per iteration based on the selection of the parameters of the algorithms.

3. **Theoretical Analysis of the Effects of Sampling on Network Performance**.

   We also analyze the effects of sampling on the network performance, which is done in Chapter 4. In comparison with existing works on this topic, we adopt a different approach, which facilitates the qualitative interpretation of some of the main results obtained. As a result, we draw some insights that, to the best of our knowledge, are novel in the literature. We also observe a good match between the theoretical analysis presented and simulation results.

4. **A Variable Step Size Adaptive Filtering Algorithm**.

   Lastly, a contribution of this work not directly related to diffusion networks was the derivation of a variable step size (VSS) adaptive filtering algorithm, which was proposed in the [JP-3] paper mentioned previously. This algorithm was heavily based on the adaptive sampling and censoring mechanisms of the [JP-1] and [JP-2] papers, though with a different goal. Its aim is to control the step size of an adaptive filter along the iterations such that the resulting algorithm presents an improved steady-state performance in comparison with the solution with a fixed step size. Simulation results showed that it can outperform other VSS algorithms, or attain a similar performance but with less parameters, whose selection is comparatively simple.

1.5 Dissertation Structure

This work is structured into five chapters and seven appendices. In Chapter 2, we provide an extensive literature review on adaptive diffusion networks, from their inception to the latest advancements and open research problems.

In Chapter 3, we present the various algorithms proposed for the reduction of the computational cost and number of transmissions in steady state. Moreover, we analyze how the parameters of these algorithms influence the expected number of sampled (or censored) nodes, and show simulation results obtained with them.

In Chapter 4, we analyze the impact of node sampling on the transient and steady-state performance of the algorithms. We begin by investigating the case in which the nodes are sampled randomly, and then extend the analysis to the algorithms presented in Chapter 3.

Finally, in Chapter 5, we present the main conclusions of our work, as well as suggestions for future research.

2 LITERATURE REVIEW

In this chapter, we provide a literature review on adaptive diffusion networks and introduce many of the algorithms and core concepts that are relevant to Chapters 3–5.

This chapter is organized as follows. In Sec. 2.1, we examine the historical development of adaptive diffusion networks in the literature. In Sec. 2.2, we present mathematical and pseudo-code descriptions of several solutions that have been proposed over the years, and discuss them in detail. Furthermore, we also show some theoretical results found in the literature as well as simulations to illustrate the behavior of these techniques with synthetic and real-world data. Lastly, in Sec. 2.3 we present a summary of the main conclusions of this chapter.

2.1 Brief History of Adaptive Diffusion Networks

In this section, we present an overview of the timeline of adaptive diffusion networks, as well as correlated research areas and technological advances. For ease of reading, we have divided it into the following subsections. In Sec. 2.1.1, we explore the context that led to the inception of adaptive diffusion networks, which are then addressed in Sec. 2.1.2, along with other strategies for distributed signal processing. In Sec. 2.1.3, we examine the consolidation of these tools in the literature and some of the research topics that were spanned by it in the following years after its emergence. Finally, in Sec. 2.1.4, we address other innovations that happened in parallel with the consolidation of adaptive diffusion networks, and the birth of other research fields that drew inspiration from them at some point.

2.1.1 Technological Background: The Emergence of Wireless Sensor Networks

The late 1990's and early 2000's were marked by rapid progress in the fields of electronics and wireless communications. Advances in commercial integrated circuit fabrication and in very large-scale integration (VLSI) technologies enabled the combination of wireless transceivers, processors, and sensors on a single chip and amplified the usage of computing and communication devices in commercial applications [101, 102]. There were also remarkable breakthroughs in wireless communications technologies. Just to put things in perspective, in 1998, the Bluetooth technology was launched as an open standard for wireless communications [102]. Another project, initiated in 1990 by the IEEE, would lead to the release of a wireless networking standard in 1997: the IEEE 802.11 [103]. This, in its turn, would become the basis for wireless local access areas (WLANs), and give rise to a family of wireless network protocols that received the brand name of Wi-Fi™ in 1999 [104]. In 2001, the first commercial 3G mobile service was launched in Japan [105].

Such advances would lead to the development of low-cost, low-power micro-sensors with embedded processors and radios [106–109]. Thus, these devices were capable of sensing and processing data, as well as communicating untethered over short distances [19, 110]. This made them promising mainly for three reasons:

1. Their low cost facilitated the deployment of numerous sensors over an area in order to monitor a signal of interest, which is particularly beneficial when the exact location of the source of such signal is not known beforehand. In this case, the placement of many sensors can lead to improved signal-to-noise ratios (SNR) [110].

2. Their capability of communicating wirelessly enabled their use in areas where wired networking was impractical.

3. Their low power consumption allowed them to run on small energy sources, dismissing the need for a continuous connection to the energy infrastructure (e.g., the power grid).

These features enabled the use of such micro-sensors in remote regions and made them suitable for environmental monitoring, precision agriculture, smart homes, military applications, among many other applications [19, 110].

Thus, although energy consumption and production costs were still too high to make them feasible for large-scale applications in the short term [111], expectations arose that WSNs would emerge in the following years or decades and revolutionize our relations with complex systems [110]. Indeed, over the 2000's, some of the initial shortcomings were addressed. For instance, although Bluetooth and WLAN are not specifically suitable for low-power networks, other standardization attempts would arise in order to meet the needs of WSNs, such as ZigBee [112], WirelessHART [113], and SP100.11a [114]. Moreover, the 6LoWPAN standard would be proposed in order to make WSNs compatible with the internet [115].

Among the technical challenges posed by WSNs, energy consumption was the most demanding. This was mainly due to their reliance on small power sources and on wireless communications. Besides presenting short range, the minimum output power required to transmit a signal over a distance d is proportional to d^n, where $2 \leqslant n < 4$ [19]. For low-lying antennae and near-ground channels, as is the case in most sensor networks, the exponent n is close to 4 due to ground reflections [19, 109, 110].

As a result, distributed signal processing techniques were soon perceived to be more appropriate than centralized approaches for WSNs [7, 9, 116, 117]. In a distributed setting, each sensor node only communicates with its immediate neighbors. On the other hand, in the centralized framework, every node must send its data to a central processing unit, sometimes called a *fusion center*. However, due to the energy constraints, it is desirable to process the data as much as possible within the network, in order to limit the amount of bits transmitted over long distances [110]. Another disadvantage associated with this approach resides in the fact that it inserts a critical point of failure in the setting, namely, the fusion center. Furthermore, the central unit must be capable of dealing with large amounts of data, meaning that it requires more sophisticated processors and communication modules. Lastly, the short range of the radio components in the sensors limits the scalability of the network in the centralized setting.

Furthermore, there was a concern that the mass production of micro-sensors and their large-scale usage in future applications would create an impracticable demand on cable installation and network bandwidth [109]. Thus, by processing as much data as possible locally, the financial, computational, and management burden on communication systems and networks could be significantly alleviated.

The quickly-developing nature of this topic sparked the interest of the academic community. In 2000, the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) had a session entitled “Signal Processing and Protocols for Wireless Sensors”. It was the first edition of the conference to dedicate an entire session to this topic. In the following year, there were three sessions dedicated to wireless networks, communications, and systems in ICASSP. That same year, a workshop entitled *Collaborative Signal Processing* was held in Palo Alto, California. This would later be renamed as the *IEEE/ACM International Conference on Information Processing in Sensor Networks*, a symposium that has been held every year since then. In 2002, the Association for Computing Machinery held its *1st ACM International Workshop on Wireless Sensor Networks and Applications*. In 2004, the European Association for Signal Processing (EURASIP) launched an open-access journal entitled *Eurasip Journal on Wireless Communications and Networking*, which places an emphasis on signal processing techniques for these technologies [118]. It is interesting to note that, around the same time, the interest of the scientific community in complex networks began to rise for a number of factors. Firstly, the study of complex systems in general was aided by the increasing availability of powerful computers. In particular, this facilitated computations involving networks with millions of nodes. Moreover, the popularization of the internet also played a role, by enabling the exchange of databases among researchers. In fact, the internet itself was an example of a complex network, and was the subject of a handful of studies [119–121].

The prominence that the aforementioned conferences gave to the topic of wireless sensor networks is an indication of the interest that it sparked among the engineering and computer science communities. This enthusiasm would not fade in the following years – rather, it would skyrocket. As depicted in Fig. 1, between 2002 and 2005 the number of academic publications with the words “wireless sensor networks” in their title escalated from a few tens to more than a thousand, including conference and journal papers, editorials, books and book chapters, among others [122]. Furthermore, attention on the topic continued to grow steadily throughout the 2000’s, and it maintained the widespread interest of the scientific community during the 2010’s.

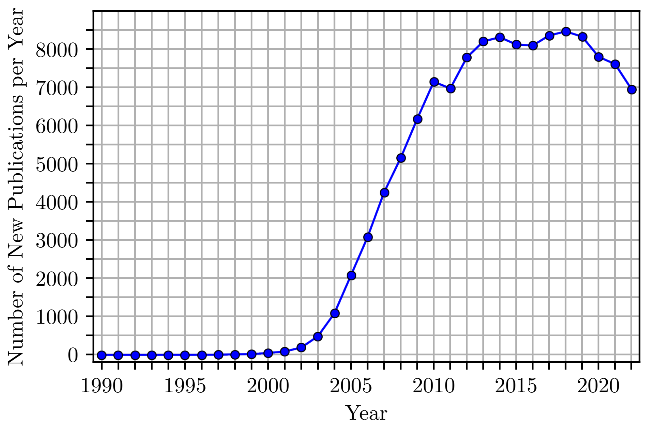


Figure 1: Number of publications with the words “wireless sensor networks” in their title from 1991 through 2022 [122].

2.1.2 How to Distribute the Processing?

Up until the mid 2000’s, wireless sensor networks were oftentimes considered as a tool for directly estimating a certain signal of interest. Thus, the main tasks of WSNs were typically to filter the measurement noise out using, e.g., linear filtering, and to estimate a parameter of interest utilizing the maximum likelihood method, for instance. The results corresponding to different spatial locations were then averaged [12, 13, 16, 123, 124]. In the second half of the decade, *adaptive sensor networks with distributed processing* were proposed [1–6, 9, 10]. These solutions consisted in the extension of well-established adaptive filtering algorithms, such as least-mean-squares (LMS) and recursive least-squares (RLS) algorithms [125, 126], to distributed problems over sensor networks. With this fusion of ideas, WSNs could be endowed with the ability to solve optimization problems in an adaptive manner, which enabled them to follow changes in the environment, improved their overall flexibility, and broadened their scope of applications, including, e.g., the study of biological and social networks, system identification problems, distributed spectrum sensing in cognitive radio networks, environmental monitoring, among others [1, 22, 23, 33, 127].

In parallel, another question was the subject of much scrutiny throughout the 2000's: *how* exactly to carry out the processing in a distributed manner. More specifically, researchers sought to determine how to organize the communication between nodes and how to incorporate their cooperation in the processing of the data in an efficient manner, ensuring satisfactory performance as well as low power consumption and computational cost. Some of the earliest approaches to data fusion in sensor networks were the so-called *flooding* techniques, such as the ones used in conventional *ad hoc* networks. In its purest form, flooding works as follows: each node keeps a table where it stores all its known data. At every time instant, each node then broadcasts this table to its neighbors. In a stationary environment, this means that eventually every node will be able to act as a fusion center and estimate the parameters of interest [12]. Naturally, this procedure demands high storage and communication capabilities and generates a large overhead [16]. More efficient versions of flooding were proposed over the years [128–131], but the potential for increased node density in WSNs and the technical limitations of each individual sensor made this technique unappealing in the field.

In the mid 2000's, two approaches for communication and cooperation between nodes began to attract attention in the context of wireless sensor networks: the *consensus* [11–17] and *incremental* [8–10] strategies. Both approaches predate the emergence of WSNs, hailing from the fields of statistics [132, 133] and optimization theory [8], respectively. However, as will be seen in Sec. 2.2, they present some shortcomings that hindered their potential for application in the specific context of WSNs. The incremental strategy is not robust to link and node failures, since the breakdown of a single node or link halts the entire learning process due to the disruption of the information chain. Moreover, it requires that the nodes be arranged in a Hamiltonian cycle, which is, in general, an NP-hard problem [134]. In contrast, in consensus techniques, the nodes are allowed to communicate with each other according to a predefined topology. As a result, they may cooperate with multiple peers, which eliminates some of the issues associated with the incremental strategy. However, they also present some drawbacks. Initially, they required the existence of two time-scales: one for the processing of the data, and the other for reaching a consensus between the nodes at each iteration of the adaptation problem [16, 135]. Furthermore, the step sizes in traditional consensus techniques diminish over time. It was noted that both of these traits could be problematic when dealing with online applications with streaming data. As a result, adjustments were proposed to the consensus schemes in order to alleviate some of these challenges, but some discrepancies remained [136, 137]. A more direct technique for adaptation over networks was proposed earlier leading to the *diffusion strategies* [4–7, 138]. Later, it was shown that the diffusion strategies provided better stability than their consensus counterparts for adaptation with in-network processing [139]. As a result, diffusion strategies remained the predominant technique for the use in adaptive networks, although incremental and consensus strategies continued to be considered over the following years [140–145]. For this reason, in this work, we focus mainly on diffusion techniques. An excellent, in-depth comparison of incremental, consensus, and diffusion strategies can be seen in [3].

Thus, by endowing the networks with the ability to adapt to the signals of interest at hand, and by adopting diffusion strategies to disseminate information throughout the network, the backbones of adaptive diffusion networks were laid out [4–7,138], and they began to popularize.

2.1.3 Diffusion strategies consolidate and are extended

Diffusion strategies soon became the most widely used protocol in adaptive sensor networks [1–6]. From this point on, much of the research done on adaptive sensor networks focused on them – either expanding their methods to other fields of the distributed signal processing area, or attempting to address some of their limitations. For example, despite their advantages over centralized approaches and other distributed schemes, diffusion networks may require a high number of communications. For this reason, over the 2010s decade, several techniques were proposed to reduce the energy consumption associated with the communication processes. Some aimed to reduce the amount of information sent in each transmission [54–62], whereas others shut links off according to selective communication policies [63–65, 67–71]. Last but not least, there is a group of solutions known as *censoring* techniques. They seek to cut the transmission from certain nodes to any of their neighbors [72–82], hence allowing censored nodes to turn their transmitters off. This saves energy and reduces the amount of information used in the processing [74, 75].

Meanwhile, other works sought to investigate the impact of some forms of uncertainty on the behavior of adaptive diffusion networks, which may be inevitable in practical implementations. These uncertainties included changes in the network topology during the operation of the adaptive algorithms [63, 65], random link failures, random data arrival times, and agents turning on and off randomly [146–148]. Since these effects are asynchronous in nature due to their randomness, networks where these occurrences can be observed were named as *asynchronous networks* in the literature [1, 3, 146–148].

Furthermore, from the mid 2010's onward, extensions of adaptive diffusion networks began to be proposed. For instance, the topic of *multitask networks* received considerable attention from 2014 onward [20–31]. These networks can be seen as a generalization of the solutions that had been previously studied, which, in the multitask literature, are named as *single-task networks* for distinction. In contrast with single-task solutions, the multitask approach considers a network of nodes that do not share the same exact objective, but rather have overlapping (albeit distinct) estimation interests [23]. Hence, there are multiple parameter vectors to be inferred simultaneously and in a collaborative manner [22]. This situation is frequently observed in distributed temperature estimation problems where the parameters that determine the evolution of temperature over time vary in space [22]. Other examples of applications include target tracking problems in which there are multiple targets to be tracked simultaneously, and spectrum sensing over cognitive radio networks, for instance [22, 23].

Around the same period, diffusion networks with nonlinear processing began to be proposed. In particular, solutions based on *methods* became popular in the diffusion networks literature [32–36]. Some of the first solutions of this kind can be viewed as an extension of kernel adaptive filters [149–151] to the context of diffusion networks [32–34]. Kernel adaptive filters map the input signal to a vector space of higher dimension (typically through a nonlinear mapping function), where a linear adaptive filter is employed. Thus, nonlinear filtering can be achieved by the use of linear techniques on the mapped signals. Moreover, by making use of the kernel trick [149, 151], kernel adaptive filters typically do not calculate the mapping of the inputs explicitly, which saves computational power. Evidently, the utility of these solutions stems from the fact that nonlinear functions are often better models for physical phenomena than linear ones. This also holds in many applications involving sensor networks. One possible example is that of environmental monitoring, where a network of sensors is deployed to observe an oftentimes nonlinear diffusion field [152]. Thus, kernel-based adaptive networks seek to address the needs of applications where the signal processing must be simultaneously nonlinear and carried out in a distributed manner [32–36].

The aforementioned research topics are not isolated from each other. For example, there have been proposals of multitask kernel networks [153] and multitask asynchronous networks [154] in the literature. Finally, one cannot review the mid 2010's in the adaptive diffusion networks literature and not mention one of the main milestones achieved by the area around this time. In 2015, the first large-scale journal dedicated specifically to the subject was launched: the *IEEE Transactions on Signal and Information Processing over Networks* [155], which covers topics such as distributed algorithms for filtering, detection, estimation, adaptation and learning, model selection, data fusion, diffusion or evolution of information over networks, applications of distributed signal processing, among others.

2.1.4 Other Advances Span Correlated Tools

Evidently, while the theory and many solutions of adaptive diffusion networks were being developed, technological and scientific advances in other areas continued to unfold. During this period, smartphones stormed the mobile phone market and the number of mobile users across the globe skyrocketed [156], social media were developed and rapidly grew [157], big data and data mining applications became commonplace [158, 159], and the technologies that enable 5G communication networks were continuously developed [85], facilitating the implementation of the internet of things (IoT) in many applications [86, 87]. Moreover, in this period, the usage of WSNs in practical applications was heavily explored. Examples include, e.g., wireless body area networks (WBANs) [93, 94], which rose as appealing solutions in, e.g., healthcare and security applications [95]. Among these, one could mention for instance the wireless electroencephalography (EEG) sensor networks (WESNs), which were proposed for neuromonitoring applications [96]. Furthermore, WSNs were also employed in agriculture [97], speech, video and image enhancement, spectrum sensing, power system state estimation, among many other applications (see, e.g., [98] and the references therein).

Due to the inherent connectivity between the elements that comprise these technologies, they share an interesting trait: they can be well represented by graphs. For example, in the context of social networks, each user can be represented by a node, and the existence of an edge connecting two nodes can be used to indicate that they are friends on that platform [160]. Analogously, each end user in a communication network can be represented by a node, and an edge linking two users could indicate that they are geographically close to one another, and, therefore, the power of the signal received by them must be similar [161].

As a result, a field of research emerged and quickly attracted widespread attention in the signal processing community: the field of graph signal processing [41, 42, 47, 48, 160–167]. Broadly speaking, GSP techniques seek to explore the relationships between elements of a networked system and from (potentially partial) observations collected by them to extract useful information. In some cases, the goal is to reconstruct a certain signal defined over the graph (also known as graph signal in the literature) [41,42,161], in others, it is to infer the underlying graph topology from the resulting graph signal [163–165], and in others, it is even to identify the system that dictates the dynamics of a signal defined over a spatially distributed set of sensors [47,48]. The field of GSP has grown to become a vast research area, with many variations and nuances. Since in this work our focus lies on adaptive diffusion networks, we restrict our review to the topics of GSP that have a clear interface with those networks.

For example, the solutions aimed at system identification for graph signals usually assume that there is a graph shift operator [160] that influences the behavior of a signal of reference over time. Hence, both the topology of the network and the temporal factor play a major role in how the reference signal unfolds at each node. For this reason, this approach has enjoyed success in meteorology applications [47]. Furthermore, these solutions also took heavy inspiration from adaptive diffusion networks. In fact, they explicitly sought to develop diffusion versions of classical adaptive filtering algorithms for use in GSP [47, 48]. Consequently, these solutions bear striking similarities to many previous diffusion adaptive algorithms, despite the obvious differences that arise from the GSP context. The use of adaptive diffusion strategies in the GSP field represents yet another reason why these strategies are as relevant today as they have ever been. In fact, they have become commonplace in papers of the area [39,41,42,47,48]. This is only one of the factors that makes them such an interesting subject.

More recently, some strategies have been proposed for the distributed training of neural networks [168–171]. The reasoning behind this concept is that, even when the training of neural networks is done offline, it may be advantageous to carry it out in a distributed manner if the scale of the problem is too large or if there are privacy concerns involved. In the former case, hardware and software limitations can make it infeasible for one single device to handle massive amounts of data. At the same time, the growing availability of devices offers an opportunity for distributing the learning task among many agents. In the latter case, sending a great deal of raw sensitive information to a single processing unit may be undesirable due to security concerns. Examples include, e.g., data related to personal behaviors or medical conditions [169, 172]. Yet, we would still like to train the network using all the data available in these cases. Furthermore, it has been shown that diffusion strategies can escape from saddle points in non-convex optimization problems [173, 174]. Since neural networks often struggle with the existence of local minima in their respective loss functions, this is another reason why the topic has a great potential to be explored in future research.

These are some of the primary concerns of the field of *Federated Learning* [172, 175–179], a promising and relatively young research area – its name was coined in a 2016 paper [176] –, which emerged in the wake of the same technological developments previously mentioned, i.e., the increasing ubiquity of smart devices and the auspicious promise of IoT. In the past few years, it has attracted a growing interest in the machine learning and signal processing communities. In Fig. 2, we depict a timeline with some of the main developments in wireless communication technology, network applications, and some milestones for the adaptive diffusion networks and related fields.

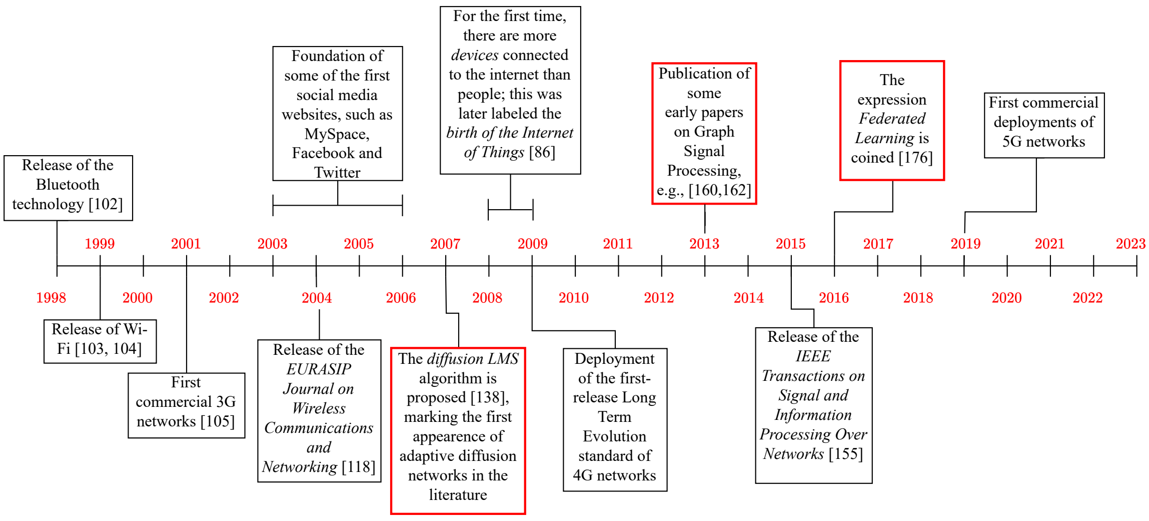

Release of the Bluetooth technology [102]
Foundation of some of the first social media websites, such as MySpace, Facebook and Twitter
For the first time, there are more *devices* connected to the internet than people; this was later labeled the *birth of the Internet of Things* [86]
Publication of some early papers on Graph Signal Processing, e.g., [160,162]
The expression *Federated Learning* is coined [176]
First commercial deployments of 5G networks
1999
2001
2003
2005
2007
2009
2011
2013
2015
2017
2019
2021
2023
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
Release of Wi-Fi [103, 104]
First commercial 3G networks [105]
Release of the *EURASIP Journal on Wireless Communications and Networking* [118]
The *diffusion LMS* algorithm is proposed [138], marking the first appearence of adaptive diffusion networks in the literature
Deployment of the first-release Long Term Evolution standard of 4G networks
Release of the *IEEE Transactions on Signal and Information Processing Over Networks* [155]

Figure 2: A timeline with several of the main events related to wireless communication technology and network applications, as well as some milestones of the adaptive diffusion network and graph signal processing fields.

Overall, many research topics remain open in the area of distributed learning. Many possible applications are becoming more viable, and the growing potential of the correlated fields is deemed promising. Efficient techniques for reducing the communication costs in adaptive diffusion networks and for sampling over diffusion GSP solutions continue to inspire the search for novel solutions, as well as the extension of these networks to more complex and challenging scenarios. For this reason, it is only fair we devote more time and space to their understanding.

2.2 Adaptive Diffusion Networks

This section is organized as follows. We begin by reviewing the single-task adaptive diffusion networks for linear adaptive filtering. In Sec. 2.2.1, we present several rules for the selection of the combination weights – an important set of parameters of adaptive diffusion networks that will be explained in more detail next. Furthermore, important theoretical results found in the literature are also presented to provide some insights into the performance of adaptive diffusion networks. In Sec. 2.2.2, we show some simulation results with synthetic data to illustrate the behavior of adaptive diffusion networks based on the discussion presented thus far. In Sec. 2.2.3, we provide an overview of techniques for restricting the number of communication processes among the nodes of a network, which is important for their feasibility in practical applications. In Sec. 2.2.4, we review multitask adaptive diffusion networks, whereas in Sec. 2.2.5 we study kernel-based adaptive diffusion networks. In Sec. 2.2.6, we discuss adaptive diffusion networks that incorporate aspects from the GSP framework. Finally, in Sec. 2.2.7 we present simulation results with real-world data to illustrate the potential as well as the challenges of adaptive diffusion networks in a more practical scenario.

Let us consider a collection of V labeled *nodes* or *agents*. As illustrated in Fig. 3, in most adaptive diffusion network applications we consider that each node k, $k = 1, \cdots, V$, has access at each time instant n to an input signal $u_k(n)$ and to a desired signal $d_k(n)$, modeled as [1–5]

$$d_k(n) = \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}^{\mathrm{o}} + v_k(n), \tag{2.1}$$

where $\mathbf{w}^{\mathrm{o}}$ and $\mathbf{u}_k(n)$ are M-length column vectors that represent respectively an unknown system and the input vector at node k. In particular, $\mathbf{u}_k(n)$ often represents a regressor vector formed by the last M samples of the input signal $u_k(n)$, i.e., $\mathbf{u}_k(n) = [u_k(n)\ u_k(n-1)\ \cdots\ u_k(n-M+1)]^{\mathrm{T}}$, although this is not necessarily the case in every application [1–3]. Furthermore, $v_k(n)$ is the measurement noise at node k, which is assumed to be white with variance $\sigma^2_{v_k}$, and independent of the other variables. Moreover, given a certain node k, we assume that the values of $v_k(n)$ along the iterations are independent and identically distributed (i.i.d.).

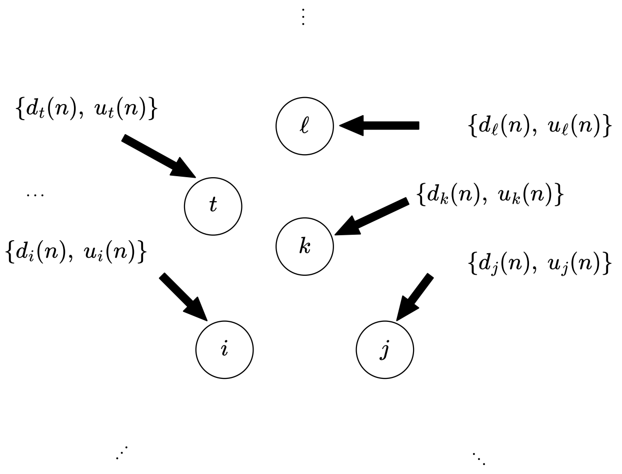


Figure 3: A collection of nodes with their respective desired signals d and input signals u.

The goal of the agents is to obtain an estimate $\mathbf{w} = [w_0 \ \cdots \ w_{M-1}]^{\mathrm{T}}$ of $\mathbf{w}^{\mathrm{o}}$. To this end, it is customary to introduce a cost function J such that [2]

$$\mathbf{w}^{\mathrm{o}} = \arg\min_{\mathbf{w}} J(\mathbf{w}). \tag{2.2}$$

For this reason, the coefficient vector $\mathbf{w}^{\mathrm{o}}$ is oftentimes referred to as the "optimal system" in the literature. One of the most widely adopted cost functions is the mean-squared error (MSE). In the case of single-agent learning, this cost function is given by $J(\mathbf{w}) \triangleq \mathrm{E}\{[d(n) - \mathbf{u}^{\mathrm{T}}(n)\mathbf{w}]^2\}$, where $\mathrm{E}\{\cdot\}$ denotes mathematical expectation and we have dropped the index k of the variables due to the existence of only one agent. The MSE is used as a cost function in the derivation of many classical adaptive filtering algorithms, such as those of the LMS and RLS types, for example [125, 126]. Since we are interested in working with the multiple agents that form the network, we introduce a local cost function $J_k(\mathbf{w})$ for each node k, which is then given by [2, 125]

$$J_k(\mathbf{w}) \triangleq \mathrm{E}\{[d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}]^2\}. \tag{2.3}$$

The first idea that may come to mind in face of (2.3) is to minimize $J_k(\mathbf{w})$ at each node k by implementing a stochastic gradient descent algorithm in each agent using only the locally available information. Following this approach, if we denote the gradient of J_k with respect to

$\mathbf{w}$ by [126]

$$\nabla_{\mathbf{w}} J_k(\mathbf{w}) \triangleq \frac{\partial J_k(\mathbf{w})}{\partial \mathbf{w}} = \left[\frac{\partial J_k(\mathbf{w})}{\partial w_0} \cdots \frac{\partial J_k(\mathbf{w})}{\partial w_{M-1}} \right]^{\mathrm{T}}, \tag{2.4}$$

and its approximate value evaluated at node k by $\widehat{\nabla}_{\mathbf{w}} J_k[\mathbf{w}_k(n-1)]$, we get, for $k = 1, \cdots, V$, [2]

$$\mathbf{w}_k(n) = \mathbf{w}_k(n-1) - \mu_k \widehat{\nabla}_{\mathbf{w}} J_k[\mathbf{w}_k(n-1)], \tag{2.5}$$

where $\mu_k > 0$ is a step size and $\mathbf{w}_k(n)$ the coefficient vector at node k. The expression for $\widehat{\nabla}_{\mathbf{w}} J_k[\mathbf{w}_k(n-1)]$ depends on the approximations made in the stochastic descent algorithm. For example, following an LMS approach, (2.5) could be recast as [2, 125]

$$\mathbf{w}_k(n) = \mathbf{w}_k(n-1) + \mu_k e_k(n) \mathbf{u}_k(n), \tag{2.6}$$

where

$$e_k(n) = d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n) \mathbf{w}_k(n-1) \tag{2.7}$$

is the estimation error.

We should notice that in (2.5) we are not assuming any form of communication between the different nodes. In fact, each node is acting as an individual adaptive filter, and they are not working together as a network. For this reason, this approach is referred to as the *noncooperative* setup in the literature [1–5]. Although this solution can be effective in many situations, if the nodes have the ability to communicate with one another, not doing so can be seen as a waste of potential. After all, it is expected that the sharing of data between the nodes should improve their performances, since this means that more information is being used in the update process.

Hence, we now seek to analyze configurations in which the nodes cooperate. To do so, we describe the objective of the network as a whole by the optimization problem [1–5]

$$\min_{\mathbf{w}} J_{\text{global}}(\mathbf{w}) = \min_{\mathbf{w}} \sum_{k=1}^{V} J_k(\mathbf{w}). \tag{2.8}$$

Let us now assume that the nodes can communicate with one another through a given topology. For any node k, the subset of nodes it can communicate with (including node k itself) is called its *neighborhood*, which is denoted by $\mathcal{N}_k$. Furthermore, the nodes that it can communicate with are called its *neighbors*. An example is depicted in Fig. 4. It is important to note that a common assumption in the literature is that the network is *strongly connected*, i.e., given any pair of nodes k and i, there is a path from node k to node i and vice-versa [1–3].

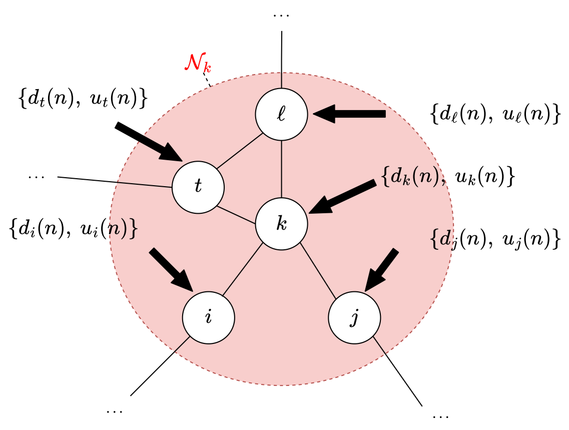


Figure 4: A network of nodes with their respective desired signals d and input signals u and a predefined topology. In particular, the neighborhood of node k is highlighted.

In this scenario, if we employ a stochastic gradient approach to solve (2.8), the cooperation between the nodes can be enforced by adopting [1–3]

$$\begin{cases} \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) - \mu_k \widehat{\nabla}_{\mathbf{w}} J_k[\mathbf{w}_k(n-1)] & (2.9a) \\ \mathbf{w}_k(n) = \sum_{i \in \mathcal{N}_k} c_{ik} \boldsymbol{\psi}_i(n), & (2.9b) \end{cases}$$

where $\{c_{ik}\}$ are convex combination weights satisfying the following set of conditions:

$$\begin{cases} c_{ik} = 0 \text{ if } i \notin \mathcal{N}_k & (2.10a) \\ \sum_{i \in \mathcal{N}_k} c_{ik} = 1 & (2.10b) \\ c_{ik} \geqslant 0,\ \forall i,k. & (2.10c) \end{cases}$$

Evidently, (2.10a) incorporates the constraints related to the network topology. On the other hand, (2.10b) is used to ensure that the combined estimates $\mathbf{w}_k(n)$ are unbiased estimators of $\mathbf{w}^{\mathrm{o}}$.

Eqs. (2.9a) and (2.9b) are known as the *adaptation* and *combination* steps, respectively of adaptive diffusion networks. The order of (2.9a) and (2.9b) characterize a configuration that is known as *adapt-then-combine* (ATC) in the literature [1–6]. One could also switch their order: in this case, the adaptation step precedes the combination one, leading to the *combine-then-adapt* (CTA) configuration, given by

$$\begin{cases} \mathbf{w}_k(n-1) = \sum_{i \in \mathcal{N}_k} c_{ik} \boldsymbol{\psi}_i(n-1) & (2.11a) \\ \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) - \mu_k \widehat{\nabla}_{\mathbf{w}} J_k[\mathbf{w}_k(n-1)]. & (2.11b) \end{cases}$$

It is worth mentioning that the ATC protocol is the one most commonly adopted in the literature [54, 56, 60, 63, 65, 67–69, 72–75]. For example, the ATC diffusion LMS (dLMS) algorithm is given by [1–5]

$$\begin{cases} \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) + \mu_k e_k(n)\mathbf{u}_k(n) & \text{(2.12a)} \\ \mathbf{w}_k(n) = \sum_{i\in\mathcal{N}_k} c_{ik}\boldsymbol{\psi}_i(n). & \text{(2.12b)} \end{cases}$$

For convenience, we also provide a pseudo-code representation of the ATC dLMS as Algorithm 1 next.

Algorithm 1 The ATC dLMS Algorithm of Eq. (2.12).

```
 1: % Initialization - for each node $k = 1, \cdots, V$, select a step size $\mu_k$ and combination weights
    $c_{ik}$ satisfying (2.10) for $i = 1, \cdots, V$, and set $\boldsymbol{\psi}_k(0) \leftarrow \mathbf{0}_M$ and $\mathbf{w}_k(0) \leftarrow \mathbf{0}_M$
 2: for $n = 1, 2, \cdots$ do
 3:     % Adaptation Step
 4:     for $k = 1, \cdots, V$ do
 5:         Update $\mathbf{u}_k(n)$
 6:         % Calculating the estimation error:
 7:         $e_k(n) \leftarrow d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n-1)$
 8:         % Adapting the local estimate $\boldsymbol{\psi}_k(n)$:
 9:         $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1) + \mu_k e_k(n)\mathbf{u}_k(n)$
10:     end for
11:     % The nodes transmit their local estimates $\boldsymbol{\psi}$ to their neighbors
12:     % Combination Step
13:     for $k = 1, \cdots, V$ do
14:         % Forming the combined estimate $\mathbf{w}_k(n)$:
15:         $\mathbf{w}_k(n) \leftarrow \mathbf{0}_M$
16:         for $i \in \mathcal{N}_k$ do
17:             $\mathbf{w}_k(n) \leftarrow \mathbf{w}_k(n) + c_{ik}\boldsymbol{\psi}_i(n)$
18:         end for
19:     end for
20: end for
```

Besides the dLMS algorithm, diffusion versions of other adaptive algorithms were proposed in the literature, such as diffusion RLS [6, 180–182], diffusion Normalized LMS (dNLMS) [63, 183, 184], the diffusion Affine Projection Algorithm (dAPA) [185, 186], among others [64, 142].

Eqs. (2.11) and (2.9) are the basis for diffusion adaptive networks, and form the groundwork for many diffusion solutions that were proposed over the following years. They allow us to attain the good performance of centralized solutions in a fully distributed way, without many of their limitations. The difference between the diffusion strategies and the noncooperative approach described by (2.5) lies in the existence of the combination step, which allows the knowledge gained by a node to disseminate throughout the whole network, enabling a better performance. It is worth noting, nonetheless, that the potentially high number of communication processes between the nodes can pose a challenge in terms of bandwidth requirements and especially energy consumption – which will be addressed in Sec. 2.2.3.

There is a generalized form of the diffusion strategies of (2.9) and (2.11) that is worth mentioning. In addition to the exchange of local estimates $\boldsymbol{\psi}_i$ prior to the combination step, we could also allow each node to share its approximate gradient $\widehat{\nabla}_{\mathbf{w}} J_i[\mathbf{w}_i(n-1)]$ before the adaptation step. Then, each node k can perform a combination of these approximate gradients in the update of $\boldsymbol{\psi}_k(n)$. Following the ATC configuration, this leads to [1]

$$\begin{cases} \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) - \mu_k \sum\limits_{i \in \mathcal{N}_k} b_{ik} \widehat{\nabla}_{\mathbf{w}} J_i[\mathbf{w}_k(n-1)] & (2.13a) \\ \mathbf{w}_k(n) = \sum\limits_{i \in \mathcal{N}_k} c_{ik} \boldsymbol{\psi}_i(n), & (2.13b) \end{cases}$$

where b_{ik} are combination weights satisfying (2.10). Evidently, an analogous version of (2.13) can be obtained for CTA. These schemes are sometimes referred to as *diffusion strategies with enlarged cooperation* [1, 3]. It should be noted that (2.13) implies that the nodes communicate twice per iteration: first, before (2.13a) is executed, each node i must share its local data $\{d_i(n), \mathbf{u}_i(n)\}$ with its neighbors to enable the calculation of $\widehat{\nabla}_{\mathbf{w}} J_i[\mathbf{w}_k(n-1)]$. After (2.13a) is carried out, they must then share their local estimates $\boldsymbol{\psi}$ before (2.13b) is run. Since this can be very costly from an energetic point of view, the diffusion strategies of the forms (2.9) and (2.11) are more common in the literature; see e.g. [54, 56, 60, 63, 65, 67–69, 72–75].

It can be shown that the stability of the standard diffusion LMS algorithm described by (2.12) is ensured in the mean sense if each step size μ_k satisfies [2]

$$0 < \mu_k < \frac{2}{\lambda_{\max}\left(\mathbf{R}_{u_k}\right)}, \tag{2.14}$$

where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue, and $\mathbf{R}_{u_k} \triangleq \mathrm{E}\{\mathbf{u}_k(n)\mathbf{u}_k^{\mathrm{T}}(n)\}$ is the autocorrelation matrix of the input signal at node k. There is a clear connection between (2.14) and the stability condition for a single LMS adaptive filter (see, e.g., [125, 126]). In fact, adaptive diffusion networks inherit many of the traits of the adaptive filters that inspired them. Typically, the greater the step sizes μ_k, the faster the convergence rate of the network in the transient phase. After a while, the performance of the network stabilizes and a steady state is achieved. In general, the greater the step sizes, the worse the steady-state performance [1–3]. This will become clear in Sec. 2.2.3.2, when we introduce some performance indicators and present theoretical predictions for them based on the step sizes, filter length M, and combination weights. Moreover, due to their adaptive nature, the algorithms of Eqs. (2.9) and (2.11) are able to track changes in the environment. In this regard, it is interesting to notice that this is only possible if the step sizes do not diminish indefinitely over time – which was a trait of the first consensus strategies that proved detrimental to their usage in adaptive networks.

As is known since the inception of the first adaptive filtering algorithms [125, 126], adaptation is a powerful feature for online learning. The ability to track changes in the environment, along with the fact that we do not need to have prior knowledge on the statistics of the signals involved, is one of the many features that made the extension of adaptive signal processing techniques to WSNs so appealing.

Besides the standard diffusion strategies shown so far, modified versions of these schemes continued to be proposed over the years with different purposes. Next, we mention a few of these modified diffusion protocols. For instance, a group of adaptive algorithms for diffusion networks was proposed in [187]. These algorithms sought to improve the robustness of the networks in a scenario where nodes can fail and the data collected are very noisy. To this end, each node projects its combined estimate onto a hyperslab or a halfspace, and then uses the latest q projections in the adaptation. By adopting cost functions common in the robust statistics field, such as a modified version of the Huber cost [188], the combined estimates can be projected on halfspaces that simultaneously reduce the MSE and mitigate the effect of outliers in the data, which may be present due to malfunctioning nodes or noisy measurements. Other robust diffusion algorithms based on the Huber cost function were also proposed in [189, 190]. Furthermore, several robust diffusion techniques have also been proposed to deal with errors-in-variable (EIV) models based on the total least-squares approach [191]. EIV models can be used to represent, e.g., situations in which the input signal is subject to noise, as well as the desired signal. Representative examples include, e.g., the solutions proposed in [192–196]. The interested reader can find a review focused primarily on robust diffusion solutions in [197].

In addition to the distributed estimation algorithms that have been discussed so far, diffusion-based detection solutions have also been proposed [198–202]. In these cases, the goal of each agent in the network is to provide a decision about the state of a system, which can vary over time. It was shown in [199] that, by adopting a similar strategy to the diffusion schemes for estimation, one can achieve the same performance of a centralized stochastic-gradient approach in terms of detection error exponents. Thus, despite their peculiarities, these solutions consist in i) updating the estimates about the current state of the system locally at each node, and ii) enabling the agents to exchange and combine their estimates [199–202].

Furthermore, one could mention the *sparse diffusion solutions* [182,203–206], which aim to take advantage of the inherent sparsity of many signals and system models, i.e., the presence of only a small number of nonzero entries. When the optimal system $\mathbf{w}^{\mathrm{o}}$ is sparse, these solutions present a faster convergence rate and can outperform the standard diffusion solutions of (2.11) and (2.9).

Lastly, it is worth noting that, in the solutions examined thus far, we consider that each node k has access to the unprocessed signals $d_k(n)$ and $u_k(n)$. However, other solutions have been recently proposed in which certain features are first extracted from the available data, and each node has access directly to the information from these features, rather than to the unprocessed data. Thus, if we aggregated the information from the features into a feature vector, each node would only have access to a certain block of entries in that vector. This approach can be adopted in scenarios in which there are privacy concerns, for example, or in which the data are already collected in a distributed manner, as is the case in spatial filters and sensor array processing. Some of the original ideas regarding this approach were proposed in [207], although not in a distributed setting. Examples of solutions that adopt this feature-based framework considering a distributed approach include, e.g., [208–210]. In particular, it is worth noting that in [208] a diffusion-based approach is adopted to address this scenario in which the features are distributed among the nodes, in a similar manner to Eq. (2.9).

2.2.1 The Selection of the Combination Weights and the Steady-State Performance

There are several possible rules for the selection of $\{c_{ik}\}$, which can play a significant role in the behavior of diffusion adaptive networks. Among the most widely adopted strategies found in the literature are the Uniform or Averaging [211], Laplacian [12], Metropolis [12,212, 213], Maximum-Degree [12] and Relative Degree [6] rules. For ease of reference, they are summarized as rules 1–5 in Table 1.

Although it is not the goal of this work to provide an in-depth theoretical analysis of each solution that we will review, it may be interesting to compare the performances of the noncoop-

Table 1: Summary of the some rules for the selection of the combination weights most widely adopted in the literature.

Name	Equations
1) Uniform or Average [211]	$c_{ik} = \begin{cases} \dfrac{1}{\lvert\mathcal{N}_k\rvert}, \text{ if } i \in \mathcal{N}_k \\ 0, \quad \text{otherwise} \end{cases}$
2) Metropolis [12,212,213]	$c_{ik} = \begin{cases} \dfrac{1}{\max\{\lvert\mathcal{N}_k\rvert,\lvert\mathcal{N}_i\rvert\}}, \text{ if } i \in \mathcal{N}_k\backslash\{k\} \\ 1 - \sum_{i\in\mathcal{N}_k} c_{ik}, \text{ if } i = k \\ 0, \text{ otherwise} \end{cases}$
3) Relative-Degree [6]	$c_{ik} = \begin{cases} \dfrac{\lvert\mathcal{N}_l\rvert}{\sum_{i\in\mathcal{N}_k}\lvert\mathcal{N}_i\rvert}, \text{ if } i \in \mathcal{N}_k \\ 0, \text{ otherwise} \end{cases}$
4) Laplacian [12]	$c_{ik} = \begin{cases} \dfrac{1}{\max_{i=1,\cdots,V}\lvert\mathcal{N}_i\rvert}, \text{ if } i \in \mathcal{N}_k\backslash\{k\} \\ 1 - \dfrac{\lvert\mathcal{N}_k\rvert - 1}{\max_{i=1,\cdots,V}\lvert\mathcal{N}_i\rvert}, \text{ if } i = k \\ 0, \text{ otherwise} \end{cases}$
5) Maximum-Degree [14]	$c_{ik} = \begin{cases} \dfrac{1}{V}, \text{ if } i \in \mathcal{N}_k\backslash\{k\} \\ 1 - \dfrac{\lvert\mathcal{N}_k\rvert - 1}{V}, \text{ if } i = k \\ 0, \text{ otherwise} \end{cases}$
6) Hastings [214]	$c_{ik} = \begin{cases} \dfrac{\sigma^2_{v_k}}{\max\{\lvert\mathcal{N}_k\rvert\sigma^2_{v_k},\lvert\mathcal{N}_i\rvert\sigma^2_{v_i}\}}, \text{ if } i \in \mathcal{N}_k\backslash\{k\} \\ 0, \text{ if } i \notin \mathcal{N}_k \\ 1 - \sum_{i\in\mathcal{N}_k} c_{ik}, \text{ if } i = k \end{cases}$
7) Relative Variance [215–217]	$c_{ik} = \begin{cases} \dfrac{(\sigma^2_{v_i})^{-1}}{\sum_{i\in\mathcal{N}_k}(\sigma^2_{v_i})^{-1}}, \text{ if } i \in \mathcal{N}_k \\ 0, \text{ otherwise} \end{cases}$
8) Adaptive Combination Weights (ACW) [215]	$\hat{\sigma}^2_{ik}(n) = (1-\nu_{k_{\text{ACW}}})\hat{\sigma}^2_{ik}(n-1) + \nu_{k_{\text{ACW}}}\lVert\boldsymbol{\psi}_i(n) - \mathbf{w}_k(n-1)\rVert^2$ $c_{ik}(n) = \begin{cases} \dfrac{[\hat{\sigma}^2_{ik}(n)]^{-1}}{\sum_{i\in\mathcal{N}_k}[\hat{\sigma}^2_{ik}(n)]^{-1}}, \text{ if } i \in \mathcal{N}_k \\ 0, \text{ otherwise} \end{cases}$

erative and diffusion approaches. Thus, we will present some of the main results of the analysis on these solutions without extensively deriving them for now. In Chapter 4, we shall analyze the performance of the ATC dLMS algorithm in more detail. For simplicity, we will focus on implementations of the LMS type in the form of (2.6). Furthermore, we limit our attention to a performance indicator commonly adopted in the adaptive network literature, the network mean-square deviation (NMSD), given by [1]

$$\text{NMSD}(n) = \frac{1}{V}\sum_{k=1}^{V} \mathrm{E}\left\{\left\|\widetilde{\mathbf{w}}_k(n)\right\|^2\right\}, \tag{2.15}$$

where we have introduced the weight-error vector

$$\widetilde{\mathbf{w}}_k(n) \triangleq \mathbf{w}^{\mathrm{o}} - \mathbf{w}_k(n). \tag{2.16}$$

Since the noncooperative case (2.5) corresponds to a situation in which there are V adaptive filters running separately, the NMSD defined by (2.15) will be equal to the average of the mean-square deviations (MSDs) of each individual filter. This is reasonable, since they act as isolated agents. For sufficiently small step sizes μ, it can be shown that the MSD of a single LMS adaptive filter is given by [125, 126]

$$\text{MSD}_k(\infty) = \frac{\mu_k M}{2}\sigma_{v_k}^2, \tag{2.17}$$

where $\text{MSD}_k(n) \triangleq \mathrm{E}\{\|\widetilde{\mathbf{w}}_k(n)\|^2\}$. For simplicity, let us now assume that the step sizes μ_k are the same for every node k, i.e., $\mu_1 = \cdots = \mu_V = \mu$, since this will make the comparison with the other strategies easier to understand. In this case, summing the right-hand side of (2.17) for $k = 1, \cdots, V$ leads to [1–3]

$$\text{NMSD}_{\text{ncoop}}(\infty) = \frac{\mu M}{2}\left(\frac{1}{V}\sum_{k=1}^{V}\sigma_{v_k}^2\right). \tag{2.18}$$

In contrast, assuming that the input signals have the same positive definite autocorrelation matrix $\mathbf{R}_{u_k}$ at every node k, i.e., $\mathbf{R}_{u_k} = \mathbf{R}_u > 0$ for $k = 1, \cdots, V$, it can be shown that, when the diffusion LMS algorithm is implemented with Uniform or Metropolis combination weights, its steady-state NMSD can be well approximated by [1, 3]

$$\text{NMSD}_{\text{diff.}}^{\text{unif.}}(\infty) \approx \text{NMSD}_{\text{diff.}}^{\text{metr.}}(\infty) \approx \frac{\mu M}{2V}\left(\frac{1}{V}\sum_{k=1}^{V}\sigma_{v_k}^2\right). \tag{2.19}$$

We remark that the difference between Eqs. (2.18) and (2.19) lies in the presence of a factor of V in the denominator of the fraction outside the parentheses in (2.19), which is absent in (2.18). Hence, in comparison with the noncooperative scheme, the NMSD is reduced by a factor of $\frac{1}{V}$. Furthermore, if $c_{ik} = c_{ki}$, for any k and i, it can be shown that the steady-state NMSD of the diffusion strategies is similar to that of the centralized and incremental approaches [1,3,214].

As can be seen from Table 1, the aforementioned rules use information from the network topology to determine the combination weights. As evidenced by (2.19), this can already lead to a significant improvement in performance in comparison with the noncooperative scheme. However, the network topology is evidently not the only factor that influences the performance of the diffusion algorithms. If we incorporate more information into the $\{c_{ik}\}$, we should be able to assign greater weights to the nodes that are somehow expected to perform better. For example, it is intuitive that we should privilege nodes that are subject to lower noise powers in the combination step, since they are expected to present the more refined local estimates $\boldsymbol{\psi}$. For this reason, some policies that incorporate information from the noise profile across the network were proposed. One possible example is the Hastings rule [1,3,214]. Assuming that the noise variances $\sigma^2_{v_1}, \cdots, \sigma^2_{v_V}$ are not all equal, i.e., at least one $\sigma^2_{v_k}$ is different from the others, the Hastings weights can be obtained by seeking to minimize the Network Excess MSE (NEMSE), which is given by [1,3,214]

$$\text{NEMSE} = \lim_{n\to\infty} \frac{1}{V} \cdot \sum_{k=1}^{V} \mathrm{E}\{[e_k(n) - v_k(n)]^2\}. \tag{2.20}$$

Incorporating the constraints of (2.10), this leads to an optimization problem that can be solved using a procedure proposed by Hastings [213,218], thus yielding Rule 6) of Table 1 [214]. It is easy to see from the table that the Hastings rule assigns greater weights to the least noisy neighbors, which is in accordance with our expectations. If all the nodes are subject to the same noise power, i.e., $\sigma^2_{v_k} = \sigma^2_v$ for $k = 1, \cdots, V$, the Hastings weights coincide with the Metropolis ones.

It can be shown that, if the Hastings weights are used, the diffusion LMS generally outperforms the same strategy with Uniform or Metropolis weights, i.e., [1,3,214]

$$\text{NMSD}^{\text{Hastings}}_{\text{diff.}}(\infty) < \text{NMSD}^{\text{unif.}}_{\text{diff.}}(\infty) < \text{NMSD}_{\text{ncoop}}(\infty). \tag{2.21}$$

One important aspect of the Hastings rule is that it requires the *a priori* knowledge of the noise variances across all nodes, i.e., $\sigma^2_{v_1}, \cdots, \sigma^2_{v_V}$. However, this information may not always be known beforehand. For this reason, several *adaptive combination weights* (ACW) algorithms were proposed for the selection of the combination weights in an adaptive manner, while seeking to incorporate information from the noise profile in the network. Some early examples of this strategy were proposed in [219] and [215].

For instance, the algorithm of [215], which was later analyzed in more detail in [216], seeks to minimize $\mathrm{Tr}(\mathbf{C}^{\mathrm{T}}\mathbf{R}_v\mathbf{C})$ subject to (2.10), where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix, $\mathbf{C}$ is a $V \times V$ matrix aggregating the combination weights, such that $[\mathbf{C}]_{ik} = c_{ik}$, and $\mathbf{R}_v$ is a diagonal matrix whose entries are equal to $\sigma^2_{v_1}, \cdots, \sigma^2_{v_V}$. Hence, it incorporates information from the noise power profile directly into the combination weights, similarly to the Hastings rule. It can be shown that the solution to this optimization problem leads to Rule 7) of Table 1 [215, 216], which is named as *relative variance rule* in the literature [215–217]. In order to eliminate the need for *a priori* knowledge of the noise powers, in the algorithm of [215], each node k estimates the noise power at the neighboring nodes $i \in \mathcal{N}_k$ as

$$\widehat{\sigma}^2_{ik}(n) = (1 - \nu_{k_{\mathrm{ACW}}})\widehat{\sigma}^2_{ik}(n-1) + \nu_{k_{\mathrm{ACW}}}\|\boldsymbol{\psi}_i(n) - \mathbf{w}_k(n-1)\|^2, \tag{2.22}$$

where $0 < \nu_{k_{\mathrm{ACW}}} < 1$ for $k = 1, \cdots, V$ [215] is a parameter to aid in the estimation of the noise variance. Oftentimes, $\nu_{k_{\mathrm{ACW}}} = 0.2$ is adopted in the literature [72, 215, 217]. Then, the combination weight $c_{ik}(n)$ is given by

$$c_{ik}(n) = \begin{cases} \dfrac{[\widehat{\sigma}^2_{ik}(n)]^{-1}}{\sum_{i \in \mathcal{N}_k}[\widehat{\sigma}^2_{ik}(n)]^{-1}}, \text{ if } i \in \mathcal{N}_k \\ 0, \text{ otherwise} \end{cases}. \tag{2.23}$$

These equations are summarized as Rule 8) in Tab. 1.

The improvement in NMSD brought by the adoption of this adaptive algorithm for the selection of the combination weights comes at the expense of a slower convergence rate in comparison with static combination rules [217, 220]. Furthermore, it has been noted that if the nodes use different step sizes μ_k, the algorithm of [215] can privilege the slower nodes in the combination step [221], and that the performance of the algorithm may be affected by the initial values assigned to each $\widehat{\sigma}^2_{ik}$ [222]. Nonetheless, due to its relative simplicity and the improvement in steady-state performance that it produces, the algorithm of [215] is widely used in the literature [67, 72, 74, 214, 216, 220, 222, 223].

Many other techniques were proposed over the years for the adaptive selection of the combination weights. As previously mentioned, in [223], the $\{c_{ik}\}$ are updated at every iteration following the APA algorithm or a LS method in order to minimize instantaneous approximations of the MSE at node k and time instant n. In [217] and [220], adaptive algorithms were proposed to take advantage of the improvement in steady-state NMSD due to the adaptive combination weights while seeking to mitigate the deterioration in the convergence rate that arises from its adoption. For this purpose, they implemented switching mechanisms that adapt the combination weights in steady state, but lead to the adoption of static rules in the transient. In [222], an algorithm for the selection of the combination weights was proposed to minimize the steady-state NMSD based on the consensus propagation technique, which is typically used for averaging results across a network of agents [224], and was shown capable of outperforming the algorithm of [215]. In [225], an algorithm for the selection of the $\{c_{ik}\}$ that takes the communication channel distortion into consideration was proposed. These are only a few of the solutions that can be found in the literature, and although it is out of the scope of this work to provide an exhaustive list, the reader is encouraged to consult, e.g., [221, 226–228] and the references therein.

2.2.2 Simulations – Exploring the Theoretical Results

In this subsection, we provide some simulation results to illustrate the main arguments made so far. We consider the network of Fig. 5, which has $V = 20$ nodes, and was generated randomly according to the Erdös-Renyi model [47]. The average neighborhood size throughout the network is $\frac{1}{V}\sum_{k=1}^{V}|\mathcal{N}_k| = 4.7$. Furthermore, we consider distributed implementations of the LMS algorithm.

The simulation results were obtained over an average of 100 independent realizations in a system identification setup. We set the length of both the filter and the optimal system $\mathbf{w}^{\mathrm{o}}$ to $M = 64$. Moreover, the coefficients of $\mathbf{w}^{\mathrm{o}}$ are generated randomly following a Uniform distribution in the range $[-1,1]$, and later normalized so that $\mathbf{w}^{\mathrm{o}}$ has unit norm. We consider a white Gaussian distribution for $u_k(n)$ with zero mean and unit variance. Lastly, we consider a white Gaussian distribution for $v_k(n)$ with zero mean and a different noise variance $\sigma^2_{v_k}$ for $k = 1, \cdots, V$, drawn from a Uniform distribution in the range $[10^{-3}, 10^{-2}]$. As a performance indicator, we use the NMSD given by (2.15).

In order to validate (2.18), (2.19), and (2.21), in Fig. 6, we present the steady-state NMSD (in dB) obtained in the simulations considering the noncooperative LMS strategy of (2.5) and the ATC dLMS of (2.9), respectively. For the latter, we consider the Metropolis rule, as well as the ACW algorithm shown in Table 1. In all cases, we employ the same step size $\mu_k = \mu$ for

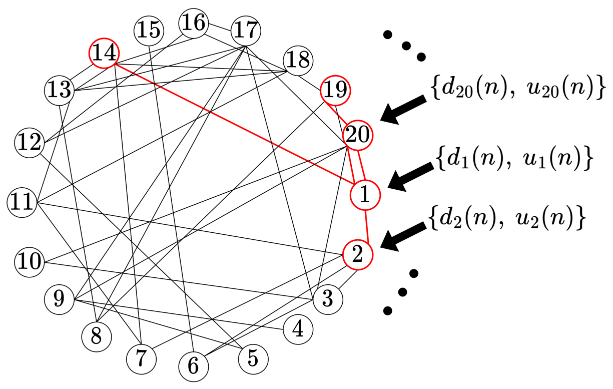


Figure 5: Network used in the simulations of Sec. 2.2.2. The neighborhood of node 1 is highlighted in red.

every node k, $k = 1, \cdots, 20$. Moreover, as a benchmark, we also consider a centralized solution in which a single unit employs the data from all of the nodes to adapt its model, given by

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu \cdot \frac{1}{V} \sum_{k=1}^{V} [d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}(n-1)]\mathbf{u}_k(n).$$

The results are presented for different values of μ in the range $[10^{-4}, 10^{-2}]$. We considered a stationary environment and $120 \cdot 10^3$ iterations per realization. The results presented were obtained by calculating the average NMSD during the last $24 \cdot 10^3$ iterations of each realization, after the algorithms have converged. In addition to the simulation results, we also show the theoretical steady-state NMSD levels obtained from (2.18) and (2.19), which are presented in dashed lines.

We can observe from Fig. 6 that, for every value of μ, the noncooperative strategy is outperformed by the cooperative schemes, as expected. Furthermore, the simulation results match well with Eqs. (2.18) and (2.19), especially for lower values of μ. This is reasonable, since (2.18) and (2.19) were obtained under the assumption of small step sizes. As μ increases, the simulation results for the noncooperative and diffusion strategies deteriorate progressively in comparison with the theoretical predictions. Finally, we remark that the diffusion strategy with ACW clearly outperforms all other techniques for $\mu > 7 \cdot 10^{-4}$. This is in accordance with (2.21), if we take into account the fact that ACW aims at implementing the Hastings weights of Rule 7) of Table 1 without prior knowledge of the noise variance in the network. For lower values of μ, the difference in performance entailed by the adoption of ACW is marginal.

From Fig. 6, we can see that the adoption of small step sizes leads to lower steady-state

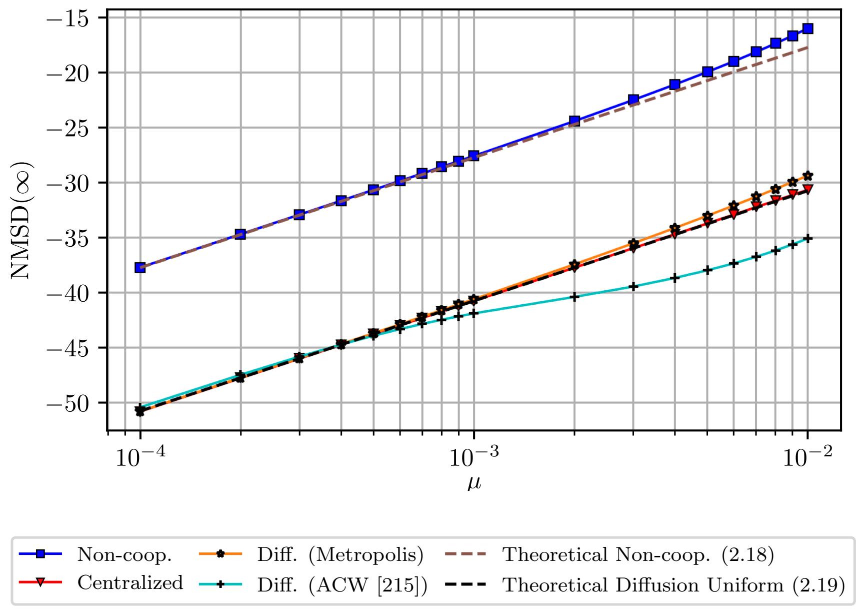


Figure 6: Steady-state NMSD (in dB) versus step size μ for various strategies, as well as the theoretical results from (2.18) and (2.19).

levels of NMSD. On the other hand, this also slows down the convergence rate. To illustrate this, in Fig. 7 we show the NMSD along the iterations obtained with the same techniques used in the simulations of Fig. 6, considering two values for the step sizes: $\mu = 10^{-3}$ in Fig. 7(a) and $\mu = 10^{-2}$ in Fig. 7(b). We consider $50 \cdot 10^3$ iterations in each realization and, to simulate an abrupt change in the environment, in the middle of each experiment we flip the vector $\mathbf{w}^{\text{o}}$. From Figs. 7(a) and (b) we can also see that the diffusion strategy with ACW presents a slightly slower convergence rate in comparison with the same technique with Metropolis weights. Nevertheless, with a step size of $\mu = 10^{-2}$, the diffusion algorithm with ACW achieves a noticeably lower level of steady-state NMSD in comparison with the case in which Metropolis weights are employed, as can be seen from Fig. 7(b).

2.2.3 Restricting Communication Policies

From the previous discussion, we observe from Eqs. (2.18), (2.19) and (2.21) that the exchange of information between the nodes can reduce the steady-state NMSD by a factor of $\frac{1}{V}$ or more. Unfortunately, allowing the permanent cooperation between the nodes can be challenging in practice. This is because energy consumption is usually the most critical constraint in WSNs, and the communication between different agents is frequently the most energy-demanding task associated with the learning process. Furthermore, a high number of communication processes between the nodes demands the allocation of sufficient bandwidth for the exchange of infor-

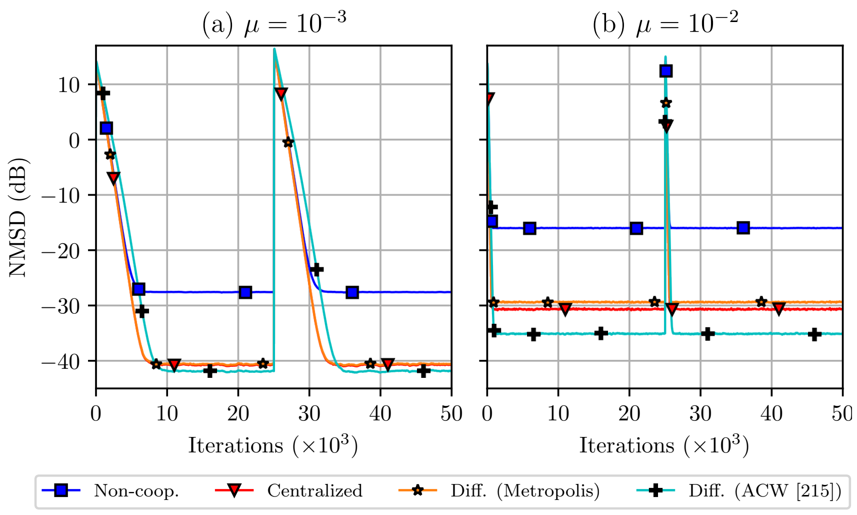


Figure 7: NMSD along the iteration obtained with various strategies, considering two different step sizes.

mation between them. For this reason, several solutions were proposed in the literature over the years to seek a compromise between the energy consumption and the benefits of cooperation to the performance. They allow the nodes to communicate with one another, but attempt to restrict the communication processes across the network in diffusion strategies. Although there is not an “official” distinction in the literature between these solutions, we classify them in three general categories: the *packet size reducing techniques*, the *link selection policies*, and the *censoring strategies*, which we review in detail in the following subsections. The goal of this section is not to provide an exhaustive review of every solution proposed in the literature for restricting communication policies, but rather to showcase some of the main ideas behind them, and to illustrate some of the most prominent characteristics that are common to most of these solutions.

2.2.3.1 Packet Size Reducing Techniques

The packet size reducing schemes aim to reduce the amount of information sent in each transmission by the nodes. Thus, by reducing the length of the messages, they enable a reduction in energy consumption and bandwidth usage, since the energy associated with the transmission of a package often scales linearly with its size in wireless communications [229,230]. Examples of this strategy include, e.g., [54–62].

For instance, in [54], the authors propose to select $L < M$ entries of the local estimates $\boldsymbol{\psi}$ for transmission at every iteration. For this purpose, they introduce an $M \times M$ entry-selection matrix

$\mathbf{S}_k(n)$ in each node k, $k=1,\cdots,V$. The matrix $\mathbf{S}_k(n)$ is diagonal, and its elements are equal to 1 or 0. If $[\mathbf{S}_k(n)]_{m,m}=1$, the node k will send the m-th entry of $\boldsymbol{\psi}_k(n)$ to its neighbors. Otherwise, $[\boldsymbol{\psi}_k(n)]_m$ will not be sent, and is replaced by the corresponding entry of the local estimate of the receiving node. Thus, following an ATC strategy, (2.9b) is replaced with [54]

$$\mathbf{w}_k(n)=c_{kk}\boldsymbol{\psi}_k(n)+\sum_{i\in\mathcal{N}_k\backslash\{k\}} c_{ik}\{\mathbf{S}_i(n)\boldsymbol{\psi}_i(n)+[\mathbf{I}_M-\mathbf{S}_i(n)]\boldsymbol{\psi}_k(n)\}. \tag{2.24}$$

In principle, each node should inform which entries it is sending to its neighbors. However, two methods for the selection of the matrices $\mathbf{S}_k(n)$ are suggested in [54] that eliminate the need for this additional overhead. One of them consists in a stochastic approach in which the nodes use pseudorandom number generators (PRNGs) for the entry selection process, and share their PRNG seeds with their neighbors once, before the adaptation begins. The other approach consists in selecting the entries sequentially in a round-robin manner over the time instants n. Thus, the entries are placed in M groups of size L such that each entry is present in L groups, which are then ordered in a predetermined sequence. These groupings and their sequence are the same across all nodes, which allows them to identify which entries they are receiving [54].

As one could expect, this reduction in data exchange leads to a deterioration in steady-state performance in comparison with the standard diffusion procedure. The lower the value of L in comparison with M, the higher the steady-state NMSD becomes. Thus, there is a trade-off between energy savings and performance. This trait is not specific to the algorithm of [54], but rather common to most restrictive communication policies. The goal of these procedures is to limit the amount of information transmitted across the network while maintaining the performance of the standard diffusion schemes as much as possible.

Furthermore, many of the solutions that fit into this category seek to somehow compress the local estimates before transmitting them [57, 58, 60–62]. For example, in [61, 62], the authors consider an adapt-compress-then-combine protocol. In this solution, the compression stems from the usage of a quantized version of the local estimate. This quantized estimate is initialized in an arbitrary manner at $n=0$, and then updated along the iterations with the usage of a certain compression operator. The choice of this operator is up to the filter designer, with possible solutions such as, e.g., [231, 232]. Moreover, the compression operator is applied to the difference between the current compressed estimate and the local uncompressed one produced by the adaptation step. Then, the output of the compression operator is scaled by a factor between zero and one, and added to the previous compressed estimate.

2.2.3.2 Link Selection Policies

As their name suggests, link selection policies seek to turn certain communication links on or off, according to predefined criteria, in order to reduce the traffic of information throughout the network [63–71]. For this, (2.9b) is replaced by

$$\mathbf{w}_k(n) = \sum_{i \in \mathcal{N}_k(n)} c_{ik}(n)\boldsymbol{\psi}_i(n), \tag{2.25}$$

in which the neighborhood $\mathcal{N}_k(n)$ varies along the iterations due to the possible deactivation of the links. In this case, the combination weights also change from one time instant to another, since (2.10a)–(2.10c) need to hold at every iteration.

The possibility of deactivating communication links was first analyzed in [63], where networks with time-varying topologies were considered. In that work, the availability of the communication links between two neighboring nodes i and k is modeled as Bernoulli random variable with a success probability $p_{ik} = p_{ki}$. Although the main goal of [63] was to analyze the behavior of the network with link failures occurring at random, this idea was later widely employed as a benchmark for comparing the performance of link selection policies. For example, building upon this concept, a mechanism was proposed in [65] for controlling the probability of success of each link in an adaptive manner. In this solution, each node k assigns a probability $p_{\min} \leqslant p_{jk}(n) \leqslant p_{\max}$ to each link associated with the nodes $j \in \mathcal{N}_k(n)\backslash\{k\}$, where $p_{\min}$ and $p_{\max}$ must be selected by the filter designer. It is worth noting that $p_{kk}(n) = 1$ for every n. The probabilities $p_{jk}(n)$ are updated based on the performance gain associated with it and on the resource constraints of the network, such as the amount of energy available at each node.

2.2.3.3 Censoring Strategies

Censoring strategies aim to prevent certain nodes from transmitting their local estimates to any of their neighbors, enabling them to temporarily turn their transmitters off and, consequently, save energy [72–81]. Usually, it is assumed that the node k still receives the data from its uncensored neighbors even when it does not send its own local estimate to them [72, 73, 78, 80, 81]. However, stricter versions in which the nodes cut their communications completely when they are censored have also been proposed [73].

Generically, some of these techniques can also be described by (2.25) with the additional restriction that $c_{jk}(n) = 0$ for every node $j \in \mathcal{N}_k(n)\backslash\{k\}$ if node k is censored at iteration n [72, 78]. Another approach consists in assuming that the nodes can store the past local estimates from their neighbors [73, 81]. In this case, (2.50b) can be interpreted as

$$\mathbf{w}_k(n) = \sum_{i \in \mathcal{N}_k} c_{ik}\zeta_i(n)\boldsymbol{\psi}_i(n) + [1 - \zeta_i(n)]\bar{\boldsymbol{\psi}}_i(n), \tag{2.26}$$

where $\zeta_i(n) = 0$ if node i is censored at iteration n and $\zeta_i(n) = 1$ otherwise, and $\bar{\boldsymbol{\psi}}_i(n)$ is the last estimate received from node i.

One example of this technique was proposed in [73], which uses game theory, as well as information about the energy level at each node and performance indicators to determine whether each node should be censored or not. In their turn, the techniques proposed in [81, 82, 84] will be examined in detail in Chapter 3.

2.2.4 Multitask Adaptive Diffusion Networks

In Sec. 2.2, it was assumed that every node in the network has the common objective of estimating the same parameter vector $\mathbf{w}^{\text{o}}$, as described by (2.1). However, an interesting problem arises when different nodes try to estimate different parameters. Applications of this type are usually referred to as *multitask* estimation problems in the literature, in opposition to the *single-task* scenario modeled by (2.1). Multitask models can be useful for representing situations in which groups of agents have distinct but correlated objectives, or for modeling regional variations in the system to be identified. This diversity in the optimal system can play an important role in, e.g., meteorological applications, where the temperature can be governed by different dynamics at different geographic points [22, 25]. Hence, to reflect this, we recast (2.1) as

$$d_k(n) = \mathbf{u}_k^{\text{T}}(n)\mathbf{w}_k^{\text{o}} + v_k(n). \tag{2.27}$$

Eq. (2.1) can be seen as a special case of (2.27) in which $\mathbf{w}_1^{\text{o}} = \cdots = \mathbf{w}_V^{\text{o}} = \mathbf{w}^{\text{o}}$. This corresponds to an extreme case in which the parameter vector is the same for all nodes. The other extreme occurs when $\mathbf{w}_k^{\text{o}}$ is different for each node k. In many applications of multitask networks, an in-between case is considered, in which there are groups or *clusters* of nodes whose parameter vectors share a certain degree of similarity. In some works, networks that fall into the latter category are referred to as *clustered multitask networks*, and the expression "multitask networks" is reserved for the case where $\mathbf{w}_k^{\mathrm{o}}$ is different for each node k [20,26,30]. For the sake of generality, in this work we apply the term "multitask networks" to any network in which the optimal system is not the same for all nodes, and explicitly differentiate between the clustered and non-clustered scenarios when the context requires.

It can be shown that if the single-task diffusion LMS algorithm with static combination weights is used in a multitask environment, it produces biased estimates of the optimal parameter vectors $\mathbf{w}_k^{\mathrm{o}}$ at each node k [22]. If the vectors $\mathbf{w}_k^{\mathrm{o}}$ are similar across the network, this bias is small in magnitude and may be acceptable in many applications. However, as the spatial variations between the $\mathbf{w}_k^{\mathrm{o}}$ increase in magnitude, the bias introduced by the diffusion can cause a visible deterioration in performance. In fact, depending on the local discrepancies in the optimal system, the noncooperative approach may outperform the diffusion strategies [22]. Intuitively, this is because each node can estimate its own parameter vector much better than its neighbors in this scenario, and the cooperation between nodes does not disseminate any useful information for the learning task of each individual node. For this reason, several different approaches were proposed for the development of efficient diffusion networks for multitask problems in different scenarios [20–29].

In [20], the diffusion LMS algorithm for multitask networks is derived. It is assumed that the network is divided into Q clusters $C_1, \cdots, C_Q$, and that each cluster C_i has the same optimal system $\mathbf{w}_{C_i}^{\mathrm{o}}$, i.e., $\mathbf{w}_k^{\mathrm{o}} = \mathbf{w}_{C_i}^{\mathrm{o}}$ for every node $k \in C_i$. Furthermore, if the clusters C_i and C_j are connected, i.e., there is a link connecting a node belonging to the cluster C_i to another node that belongs to the cluster C_j, it is assumed that their optimal systems are similar in some way. Clusters that are connected to each other are also called neighbors, analogously to the notion of neighborhood between nodes. The diffusion LMS algorithm for multitask networks is obtained by adding a regularization term to the cost function (2.3) to enforce the similarity between neighboring clusters. Using the squared Euclidean distance as a regularizer, this results in the following cost function [20]:

$$J_{\text{global}}^{\text{mult.}}(\mathbf{w}_{C_i}, \cdots, \mathbf{w}_{C_Q}) = \sum_{k=1}^{V} \mathrm{E}\{[d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_{C(k)}]^2\} + \eta \sum_{k=1}^{V} \sum_{i \in \mathcal{N}_k \backslash C(k)} \bar{\rho}_{ki} \left\|\mathbf{w}_{C(k)} - \mathbf{w}_{C(i)}\right\|^2, \quad (2.28)$$

where $C(k)$ denotes the cluster to which node k belongs. An example is shown in Fig. 8, in which, for instance, $C(i) = C_1$ and $C(k) = C_2$. Furthermore, $\mathcal{N}_k \backslash C(k)$ is the set of the neighbors of node k that do not belong to the cluster $C(k)$, $\eta > 0$ is a regularization parameter, and the

$\{\bar{\rho}_{ki}\}$ are non-negative weights that adjust the regularization strength for each pair of nodes k and i. In [20], convex weights $\{\bar{\rho}_{ki}\}$ are adopted.

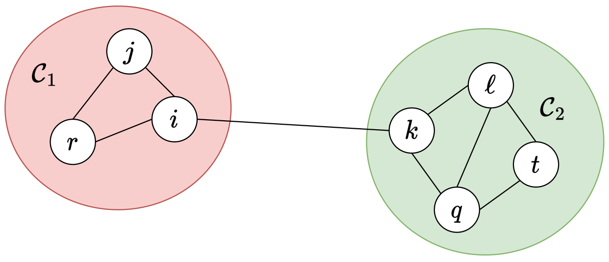


Figure 8: Example of a clustered network structure.

The cost function (2.28) promotes a stronger degree of similarity between the estimates of clusters that have many connections to each other, since in these cases the cardinalities of the sets $\mathcal{N}_k \backslash \mathcal{C}(k)$ are large. Furthermore, it enforces symmetric regularization, i.e., two neighboring clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ promote the same level of similarity between their estimates due to the summation over the V nodes of the regularization term and to the symmetry of the term $\left\|\mathbf{w}_{\mathcal{C}(k)} - \mathbf{w}_{\mathcal{C}(i)}\right\|^2$ with respect to the vectors $\mathbf{w}_{\mathcal{C}(k)}$ and $\mathbf{w}_{\mathcal{C}(i)}$. Since it may be desirable to allow asymmetric regularization terms, (2.28) was modified in [20] and formulated as a sum of Nash equilibrium problems, one for each cluster $\mathcal{C}_i$. By slightly relaxing the cost function to enable its minimization at each node using only the data available locally, and by applying a steepest-descent approach to the estimation of the equilibrium points of these problems and adopting stochastic approximations of the LMS type, the diffusion LMS for clustered multitask networks in an ATC configuration is given by [20]

$$\begin{cases} \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) + \mu_k \left\{ \sum\limits_{i \in \mathcal{N}_k \cap \mathcal{C}(k)} b_{ik} e_{ik}(n) \mathbf{u}_i(n) + \eta \sum\limits_{i \in \mathcal{N}_k \backslash \mathcal{C}(k)} \bar{\rho}_{ki} [\mathbf{w}_i(n-1) - \mathbf{w}_k(n-1)] \right\} & \text{(2.29a)} \\ \mathbf{w}_k(n) = \sum\limits_{i \in \mathcal{N}_k \cap \mathcal{C}(k)} c_{ik} \boldsymbol{\psi}_i(n), & \text{(2.29b)} \end{cases}$$

where we have introduced

$$e_{ik}(n) \triangleq d_i(n) - \mathbf{u}_i^{\mathrm{T}}(n) \mathbf{w}_k(n-1) \tag{2.30}$$

for compactness of notation.

Analyzing (2.29a), we can see that if $\eta = 0$ is chosen, that is equivalent to running the standard ATC dLMS within each cluster $\mathcal{C}_i$, without exchange of information between different clusters. Furthermore, the algorithm for the single-task scenario can be seen as a special case of (2.29) in which $\mathcal{N}_k \cap \mathcal{C}(k) = \mathcal{N}_k$ and $\mathcal{N}_k \backslash \mathcal{C}(k) = \varnothing$ for $k = 1, \cdots, V$. Finally, the non-clustered multitask case can be analyzed by making $\mathcal{N}_k \cap \mathcal{C}(k) = \{k\}$ and $\mathcal{N}_k \backslash \mathcal{C}(k) = \mathcal{N}_k \backslash \{k\}$ for $k = 1, \cdots, V$ [20]. A pseudo-code description of the multitask version of the ATC dLMS for clustered networks is provided next in Algorithm 2.

Algorithm 2 The ATC dLMS Algorithm of Eq. (2.29) for Clustered Multitask Networks.

```
1: % Initialization - For each node $k = 1, \cdots, V$, determine the cluster $\mathcal{C}(k)$ to which it belongs, select a step size $\mu_k$ as well as the parameter $\eta$ and the weights $\bar{\rho}_{ki}$, $b_{ik}$, and $c_{ik}$ for $i = 1, \cdots, V$, and set $\boldsymbol{\psi}_k(0) \leftarrow \mathbf{0}_M$ and $\mathbf{w}_k(0) \leftarrow \mathbf{0}_M$
2: for $n = 1, 2, \cdots$ do
3:     for $k = 1, \cdots, V$ do
4:         Update $\mathbf{u}_k(n)$
5:     end for
6:     % The nodes transmit their local signals $\mathbf{u}$ and $d$ and their combined estimates $\mathbf{w}$ to their neighbors
7:     % Adaptation Step
8:     for $k = 1, \cdots, V$ do
9:         % Adapting the local estimate $\boldsymbol{\psi}_k(n)$:
10:        $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1)$
11:        for $i \in \mathcal{N}_k \cap \mathcal{C}(k)$ do
12:            $\boldsymbol{\psi}_k(n) \leftarrow \boldsymbol{\psi}_k(n) + \mu_k b_{ik} [d_i(n) - \mathbf{u}_i^{\mathrm{T}}(n) \mathbf{w}_k(n-1)] \mathbf{u}_i(n)$
13:        end for
14:        for $i \in \mathcal{N}_k \backslash \mathcal{C}(k)$ do
15:            $\boldsymbol{\psi}_k(n) \leftarrow \boldsymbol{\psi}_k(n) + \mu_k \eta \bar{\rho}_{ki} [\mathbf{w}_i(n-1) - \mathbf{w}_k(n-1)]$
16:        end for
17:    end for
18:    % The nodes transmit their local estimates $\boldsymbol{\psi}$ to their neighbors
19:    % Combination Step
20:    for $k = 1, \cdots, V$ do
21:        % Forming the combined estimate $\mathbf{w}_k(n)$:
22:        $\mathbf{w}_k(n) \leftarrow \mathbf{0}_M$
23:        for $i \in \mathcal{N}_k \cap \mathcal{C}(k)$ do
24:            $\mathbf{w}_k(n) \leftarrow \mathbf{w}_k(n) + c_{ik} \boldsymbol{\psi}_i(n)$
25:        end for
26:    end for
27: end for
```

A different approach was adopted in [22] where it was assumed that there is no prior knowledge about the clusters in the network. Hence, an unsupervised algorithm for the adaptive clustering of diffusion networks was proposed. The concept behind this solution is to only promote the cooperation between each node k and its neighbors $i \in \mathcal{N}_k$ with parameter vectors $\mathbf{w}_i^{\mathrm{o}}$ sufficiently similar to its own $\mathbf{w}_k^{\mathrm{o}}$. Since there is no *a priori* knowledge of the optimal systems $\mathbf{w}_i^{\mathrm{o}}$, $i \in \mathcal{N}_k$, an approximation is obtained by adjusting the combination weights $\{c_{ik}\}$ based on the norm of the difference between their local estimates $\boldsymbol{\psi}$. This is achieved by solving at each node k

$$\mathbf{c}_k = \arg \min_{\mathbf{c}_k \in \mathbb{R}^{V\times 1}} \sum_{i=1}^{V} c_{ik}^2 \left\| \widehat{\mathbf{w}_k^{\mathrm{o}}} - \boldsymbol{\psi}_i(n) \right\|^2 \tag{2.31}$$

$$\text{subject to } c_{ik} \geqslant 0,\ \mathbf{1}_V^{\mathrm{T}} \mathbf{c}_k = 1,\ c_{ik} = 0 \text{ if } i \notin \mathcal{N}_k,$$

where $\widehat{\mathbf{w}_k^{\mathrm{o}}}$ is an estimate of $\mathbf{w}_k^{\mathrm{o}}$ and $\mathbf{c}_k$ is defined as

$$\mathbf{c}_k \triangleq [c_{1k} \ \cdots \ c_{Vk}]^{\mathrm{T}}. \tag{2.32}$$

In particular, the algorithm proposed in [21] uses

$$\widehat{\mathbf{w}_k^{\mathrm{o}}}(n) = \boldsymbol{\psi}_k(n) - \mu_k \widehat{\nabla}_{\mathbf{w}} J_k[\boldsymbol{\psi}_k(n)] = \boldsymbol{\psi}_k(n) + \mu_k \mathbf{g}_k(n), \tag{2.33}$$

with

$$\mathbf{g}_k(n) = [d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\boldsymbol{\psi}_k(n)]\mathbf{u}_k(n). \tag{2.34}$$

Thus, one possible solution for (2.31) is given by

$$c_{ik}(n) = \begin{cases} \dfrac{\left(\left\|\widehat{\mathbf{w}_k^{\mathrm{o}}}(n) - \boldsymbol{\psi}_i(n)\right\|^2\right)^{-1}}{\sum\limits_{i \in \mathcal{N}_k} \left(\left\|\widehat{\mathbf{w}_k^{\mathrm{o}}}(n) - \boldsymbol{\psi}_i(n)\right\|^2\right)^{-1}}, & \text{if } i \in \mathcal{N}_k, \\ 0, & \text{otherwise,} \end{cases} \tag{2.35}$$

with $\widehat{\mathbf{w}_k^{\mathrm{o}}}(n)$ given by (2.33). Incorporating (2.35) into (2.12) yields the ATC diffusion LMS with Adaptive Clustering for multitask problems. It is interesting to notice that there is a certain similarity between (2.35) and the ACW algorithm of Rule 8) in Table 1, despite the differences in context that motivate each solution.

As previously mentioned, multiple solutions were proposed for multitask estimation problems over diffusion networks. In [21], an ATC diffusion LMS algorithm was obtained for the case where the optimal systems $\mathbf{w}_k^{\mathrm{o}}$ can be described by the sum of a common component and a node-specific one for $k = 1, \cdots, V$ under some restrictions so as to ensure a unique optimal solution to the estimation problem. In [23], it is assumed that there are parameters that are of global interest to all nodes in the network, others that are of interest to a subset of nodes, and others that are of local interest to specific nodes, and a diffusion algorithm of the LMS type is derived for this problem. In [24], the authors propose a diffusion strategy that promotes the sparsity of the vector difference $\mathbf{w}_{C_i} - \mathbf{w}_{C_j}$ of neighboring clusters C_i and C_j. Attempting to cover all of these solutions in detail in this work would not be reasonable, as it would expand an already wide scope. An excellent review paper focused specifically on multitask adaptive diffusion networks can be found in [233].

2.2.5 Kernel Adaptive Diffusion Networks

In Secs. 2.2–2.2.4, we have discussed adaptive solutions aimed at solving linear estimation problems, as it is apparent from the modeling of the desired signal in (2.1) and (2.27). However, it is not unusual to encounter problems that are nonlinear in nature. To deal with this, kernel-based adaptive diffusion networks have been proposed in the literature and attracted significant attention [32–36]. In comparison with (2.1), the kernel framework for adaptive diffusion networks considers a modified model for the desired signal $d_k(n)$, given by [32, 35, 234]

$$d_k(n) = \varphi^{\mathrm{o}}\left[\mathbf{u}_k(n)\right] + v_k(n), \tag{2.36}$$

where $\varphi^{\mathrm{o}}(\cdot)$ typically denotes a nonlinear transformation of the input vector $\mathbf{u}_k(n)$. For simplicity, we restrict our review to the single-task scenario, but an analogous model can be obtained for the multitask one if we consider different mapping functions for different nodes [234].

Kernel-based adaptive diffusion networks are largely based on *kernel adaptive filters*, which over the years became established tools for nonlinear signal processing [149–151]. The idea behind these techniques is to apply a nonlinear transformation $\boldsymbol{\varphi} : \mathbb{R}^M \to \mathbb{F}$ to the input vectors, where $\mathbb{F}$ is usually a higher-dimensional space, named *feature space*. Then, linear signal processing techniques are applied to the vectors mapped in $\mathbb{F}$ [149–151, 235]. This way, we can perform filtering tasks that are nonlinear in $\mathbb{R}^M$ while only employing linear procedures in the feature space.

Once a nonlinear mapping function $\boldsymbol{\varphi}(\cdot)$ is adopted, the Mercer kernel $\kappa : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ associated with it, given any two input vectors $\mathbf{u} \in \mathbb{R}^M$ and $\mathbf{u}' \in \mathbb{R}^M$, is defined as the inner product [235]

$$\kappa(\mathbf{u},\mathbf{u}') = \boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{u})\boldsymbol{\varphi}(\mathbf{u}'). \tag{2.37}$$

Depending on how the mapping function $\boldsymbol{\varphi}(\cdot)$ is defined, we may be able to calculate $\kappa(\mathbf{u},\mathbf{u}')$ without explicitly knowing $\boldsymbol{\varphi}(\cdot)$. This is known as the *kernel trick*, which is a cornerstone of many kernel-based methods, such as kernel adaptive filters and support vector machines (SVMs) [149–151, 235, 236]. One of the most widely adopted kernels is the Gaussian kernel, given by

$$\kappa(\mathbf{u},\mathbf{u}') = \exp\left(\frac{-\|\mathbf{u}-\mathbf{u}'\|^2}{2h^2}\right), \tag{2.38}$$

where h is the Gaussian kernel bandwidth [149–151, 235, 236]. It is worth mentioning that the feature space $\mathbb{F}$ associated with the Gaussian kernel is an infinite-dimensional space. Other examples of commonly used kernels include sigmoidal, homogeneous and non-homogeneous polynomial kernels, among many others [149, 235].

In their original forms, the computational costs of these algorithms increase linearly with every iteration, which makes their implementation infeasible [149–151]. To deal with this, the concept of *dictionaries* was introduced in the kernel literature. The idea is to use only a limited set of data in the processing, which restricts the computational cost. Evidently, this leads to a deterioration in performance in comparison with the case where the dictionary grows larger at every iteration, but it is crucial to enable the use of these algorithms in practice. Several techniques have been proposed over the years for the selection of the dictionaries, aiming to limit the growth of the computational cost while maintaining performance as much as possible [149–151].

We then seek to estimate the desired signal by obtaining an approximation φ_k for $\varphi^{o}[\cdot]$ in (2.36) in a distributed manner in each node k. Considering a global cost function of the form (2.8), with [32]

$$J_k(\varphi_k) = \mathrm{E}\left\{\left|d_k(n) - \varphi_k\left[\mathbf{u}_k(n)\right]\right|^2\right\}, \tag{2.39}$$

and applying the stochastic gradient descent, many different algorithms can be obtained. For example, in [32], following an LMS approach and adopting an ATC diffusion strategy with enlarged cooperation, and assuming that there is a dictionary $\mathscr{D} = \{\mathbf{u}_{\mathscr{D}_1}, \cdots, \mathbf{u}_{\mathscr{D}_D}\}$ of cardinality D common to all nodes in the network, the Functional ATC diffusion Kernel LMS algorithm is obtained as [234]

$$\begin{cases} \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) + \mu_k \sum\limits_{i\in\mathcal{N}_k} b_{ik}\Big[d_i(n) - \mathbf{w}_k^{\mathrm{T}}(n-1)\boldsymbol{\kappa}_i(n)\Big]\boldsymbol{\kappa}_i(n) & \text{(2.40a)} \\ \mathbf{w}_k(n) = \sum\limits_{i\in\mathcal{N}_k} c_{ik}\boldsymbol{\psi}_i(n), & \text{(2.40b)} \end{cases}$$

where

$$\boldsymbol{\kappa}_k(n) = \Big[\kappa\left(\mathbf{u}_k(n),\mathbf{u}_{\mathscr{D}_1}\right) \;\cdots\; \kappa\left(\mathbf{u}_k(n),\mathbf{u}_{\mathscr{D}_D}\right)\Big]^{\mathrm{T}} \tag{2.41}$$

is a vector comprised of the kernel values computed between the current input vector $\mathbf{u}_k(n)$ and the elements of the dictionary $\mathscr{D}$ [234].

Multikernel solutions have also been proposed for diffusion-based adaptive learning [35, 237, 238]. As their name suggests, these techniques employ more than one kernel simultaneously in the learning task. The idea is that by using two or more kernels, we may be able to grasp nuances in the desired signal due to, e.g., the existence of high and low frequency components in the nonlinear function. A version of this technique was employed in [35] for environmental monitoring.

The fact that the dictionary $\mathscr{D}$ is shared among all nodes has some implications on the practical implementations of kernel adaptive diffusion networks. The simplest way to ensure all nodes have the same dictionary would be to define its elements and inform them to all nodes before the learning process begins [234]. Nonetheless, a general-case rule for the selection of this dictionary remains an open research topic. Furthermore, this approach can cause a degradation in performance, especially if the scenario changes in such a way that the predefined dictionary ceases to be representative of the streaming data. To circumvent these issues, one might allow a time-varying dictionary using criteria such as the ones presented in [149–151]. However, in this case the necessity of sharing the dictionary with every node in the network in an online and distributed manner poses a challenge for diffusion strategies [37]. A hybrid solution, in which each node has access to both a shared and fixed dictionary and to a local and time-varying one was proposed in [239]. The idea is that the shared dictionary should provide the nodes with a rough estimate of the nonlinear function over the whole network, whereas the local part could allow them to detect particularities of the signals in its vicinity. Despite this, the handling of the dictionary in diffusion strategies is still regarded as an open research topic [37, 234]. In order to circumvent these difficulties, *Random Fourier Features* (RFF) approaches have been proposed for kernel-based adaptive diffusion networks [37–40].

In RFF solutions, instead of using the kernel trick, we apply an RFF map $\mathbf{z} : \mathbb{R}^M \rightarrow \mathbb{R}^D$, with $D > M$, to the regressor vector $\mathbf{u}_k(n)$ based on Bochner's Theorem [37, 240]. This theorem ensures that, if the kernel is shift-invariant and positive definite, i.e., $\kappa(\mathbf{u},\mathbf{u}')$ depends exclusively on $\mathbf{u} - \mathbf{u}'$, and $\kappa(\mathbf{u},\mathbf{u}') > 0$ for any $\mathbf{u}$ and $\mathbf{u}'$, the Fourier transform $p(\omega)$ of the kernel is a probability density function such that [37, 240]

$$\kappa(\mathbf{u}-\mathbf{u}') = \int p(\omega) e^{j\omega^{\mathrm{T}}(\mathbf{u}-\mathbf{u}')} d\omega, \tag{2.42}$$

where we have written $\kappa(\mathbf{u},\mathbf{u}')$ as $\kappa(\mathbf{u}-\mathbf{u}')$ for compactness [37]. We should notice that the Gaussian kernel given by (2.38) is shift-invariant and positive definite.

Since $p(\boldsymbol{\omega})$ and $\kappa(\mathbf{u}-\mathbf{u}')$ are real, the integral (2.42) converges when the complex exponentials are replaced by cosines. Hence, the real-valued mapping $z_{\omega,\theta}[\mathbf{u}] = \sqrt{2}\cos(\boldsymbol{\omega}^{\mathrm{T}}\mathbf{u}+\theta)$ also satisfies (2.42) if $\boldsymbol{\omega}$ is drawn from $p(\boldsymbol{\omega})$ and θ is uniformly distributed in the range $[0,2\pi]$ [37]. Thus, $\kappa(\mathbf{u}-\mathbf{u}')$ can be computed as $\kappa(\mathbf{u}-\mathbf{u}') = \mathrm{E}\{z_{\omega,\theta}[\mathbf{u}]z_{\omega,\theta}[\mathbf{u}']\}$. To reduce the variance of this estimate, a sample average of D randomly chosen $z_{\omega,b}[\cdot]$ is used, i.e.,

$$\kappa(\mathbf{u},\mathbf{u}') \approx \frac{1}{D}\sum_{i=1}^{D} z_{\omega_i,\theta_i}[\mathbf{u}]z_{\omega_i,\theta_i}[\mathbf{u}']. \tag{2.43}$$

Thus, the vector $\mathbf{u}_k(n)$ can be mapped to the following D-dimensional RFF vector [37,38]:

$$\mathbf{z}_k(n) = \sqrt{\frac{2}{D}}\begin{bmatrix}\cos[\boldsymbol{\omega}_1^{\mathrm{T}}\mathbf{u}_k(n)+\theta_1]\\ \vdots \\ \cos[\boldsymbol{\omega}_D^{\mathrm{T}}\mathbf{u}_k(n)+\theta_D]\end{bmatrix}, \tag{2.44}$$

For the Gaussian kernel, $\boldsymbol{\omega}_i$, $i=1,\cdots,D$ are drawn from the multivariate Gaussian distribution with zero mean and covariance matrix $\mathbf{I}_D/h^2$ [37,240].

Since the RFF space has a finite dimension D, we can estimate $d_k(n)$ at node k by directly using a similar strategy to that of the linear ATC dLMS of (2.9). Thus, the ATC RFF-dKLMS consists in two steps given by [37,38]

$$\begin{cases}\boldsymbol{\psi}_k(n)=\mathbf{w}_k(n-1)+\mu_k[d_k(n)-\mathbf{z}_k^{\mathrm{T}}(n)\mathbf{w}_k(n-1)]\mathbf{z}_k(n) & \text{(2.45a)}\\ \mathbf{w}_k(n)=\sum_{i\in\mathcal{N}_k}c_{ik}\boldsymbol{\psi}_i(n). & \text{(2.45b)}\end{cases}$$

A pseudo-code description of ATC RFF-dKLMS is presented for convenience as Algorithm 3.

The idea behind the RFF approach is that the features are selected randomly according to the aforementioned distributions. This only needs to be done once, before the beginning of the learning task. Subsequently, the selected features are informed to the nodes. Due to Bochner's Theorem, as long as $\boldsymbol{\omega}$ and b are drawn from the appropriate distributions, this approach avoids the issue of how to select the elements of the dictionary $\mathscr{D}$ of the non-RFF kernel solutions. As one might expect, the more features are used, the better the performance of the RFF diffusion algorithms tends to be [37–40]. On the other hand, the computational cost also increases as more features are used.

Algorithm 3 The ATC RFF-dLMS Algorithm of Eq. (2.45).

```
 1: % Initialization - draw D scalars $\theta_1, \cdots, \theta_D$ from a Uniform distribution in the range $[0, 2\pi]$
    and D vectors $\omega_1, \cdots, \omega_D$ from a multivariate Gaussian distribution with zero mean and
    covariance matrix $\mathbf{I}_D/h^2$. These parameters shall be common to every node in the net-
    work. Then, for each node $k = 1, \cdots, V$, select a step size $\mu_k$ and combination weights $c_{ik}$
    satisfying (2.10) for $i = 1, \cdots, V$, and set $\boldsymbol{\psi}_k(0) \leftarrow \mathbf{0}_M$ and $\mathbf{w}_k(0) \leftarrow \mathbf{0}_M$
 2: for $n = 1, 2, \cdots$ do
 3:     % Adaptation Step
 4:     for $k = 1, \cdots, V$ do
 5:         Update $\mathbf{u}_k(n)$
 6:         % Mapping $\mathbf{u}_k(n)$ to the RFF vector $\mathbf{z}_k(n)$:
 7:         $\mathbf{z}_k(n) \leftarrow \sqrt{\frac{2}{D}} \begin{bmatrix} \cos[\omega_1^{\mathrm{T}}\mathbf{u}_k(n) + \theta_1] \\ \vdots \\ \cos[\omega_D^{\mathrm{T}}\mathbf{u}_k(n) + \theta_D] \end{bmatrix}$
 8:         % Adapting the local estimate $\boldsymbol{\psi}_k(n)$:
 9:         $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1) + \mu_k[d_k(n) - \mathbf{z}_k^{\mathrm{T}}(n)\mathbf{w}_k(n-1)]\mathbf{z}_k(n)$
10:     end for
11:     % The nodes transmit their local estimates $\boldsymbol{\psi}$ to their neighbors
12:     % Combination Step
13:     for $k = 1, \cdots, V$ do
14:         % Forming the combined estimate $\mathbf{w}_k(n)$:
15:         $\mathbf{w}_k(n) \leftarrow \mathbf{0}_M$
16:         for $i \in \mathcal{N}_k$ do
17:             $\mathbf{w}_k(n) \leftarrow \mathbf{w}_k(n) + c_{ik}\boldsymbol{\psi}_i(n)$
18:         end for
19:     end for
20: end for
```

2.2.6 Graph Signal Processing and Adaptive Diffusion Networks

So far, we have been assuming that the signals at a certain node do not influence the desired signal at the remainder of the network in any way. For instance, even if nodes k and ℓ are neighbors, $u_k(n)$ only influences $d_k(n)$, and $u_\ell(n)$ will affect $d_\ell(n)$, in its turn. This becomes clear when we analyze Eqs. (2.1), (2.27), and (2.36). However, there may be situations in which the signals measured at a node do have an impact on its neighborhood. Broadly speaking, the goal of Graph Signal Processing (GSP) is exactly to model scenarios in which there is an underlying relationship between the data distributed over a certain domain (such as space). For this reason, as we will see next, extensions of GSP have also been applied to adaptive diffusion networks, in order to address problems in which the signals are spatially related to one another.

Before going forward, it is important to remark that GSP has become a broad field, with many ramifications. Extensions of several classical ideas of the signal processing field to the graph framework have been proposed in the literature, such as the graph Fourier transform, graph signal convolution, graph filters, among many others [41, 160–167]. Rather than providing a comprehensive overview of GSP, we focus on how it can be used in conjunction with adaptive diffusion networks to model certain situations that cannot be easily represented using the theory of Secs. 2.2–2.2.5. For a much broader view of the GSP field, we suggest that the reader refers to, e.g., [162, 163, 167, 241] and their references. With this in mind, we begin our exposition on diffusion solutions for GSP in the following manner. Firstly, in Sec. 2.2.6.1, we provide some preliminary concepts that are fundamental for the understanding of the adaptive diffusion networks for GSP, which are reviewed in Sec. 2.2.6.2.

2.2.6.1 Preliminaries to Adaptive Diffusion over Graphs

Graphs are structures commonly used to represent interactions between objects of interest, consisting of a set of points, called *nodes*, and a set of lines connecting certain pairs of points, called *edges* [242]. Typically, nodes in a graph represent objects and/or agents, edges describe relationships between objects or agents, and weights represent the strength of the relation [242, 243]. Moreover, weights can be assigned to each edge, to represent the "strength" of the link between any pair of nodes. Finally, a graph is said to be *directed* if there is an orientation associated with each edge, i.e., from node i to node j or vice-versa, and if the weights associated with one direction are different from those associated with the other [243].

The adaptive networks presented in Section 2.2 can be considered examples of graphs. This is not a coincidence. Graphs are mathematical abstractions capable of representing a wide variety of real-world situations [243]. A relevant property of graphs is that they admit matrix representations, which makes them computationally manipulable [243]. One of the ways to represent them in this way is through the *adjacency matrix* of the graph. For a simple graph with V nodes, the adjacency matrix $\mathbf{A}$ is a $V \times V$ matrix whose element $[\mathbf{A}]_{ij}$ is equal to the value of the weight associated with the edge connecting the nodes i and j. It is interesting to note that the adjacency matrix of an undirected graph is symmetric with respect to its main diagonal.

The field of GSP is based upon the assumption that, at each node k and iteration n, there is a value of interest, whose meaning may vary from one application to the other. For example, these values can represent the temperature measured at a certain location, the power spectral density of a radio signal that a device receives from a certain source, the affinity of a social network user for certain types of content, and so on. The collection of these values of interest can be understood as a *signal defined over a graph*, or *graph signal* for brevity [162]. Thus, given a graph $\mathcal{G}$ with V nodes labeled $k = 1, 2, \cdots, V$ and adjacency matrix $\mathbf{A}$, a signal defined over $\mathcal{G}$ is represented by a vector of the form $\mathbf{u}(n) = [u_1(n)\ u_2(n)\ \cdots\ u_V(n)]$, where each element $u_k(n)$ is indexed by a node k [160].

Based on this definition, the concept of filters for graph signals, or *graph filters* for short, was proposed in [160]. For this, an analogy is drawn between the adjacency matrix and the unit delay operator in discrete time signal processing. To illustrate this idea, let us consider a periodic sequence with V elements $u_1, u_2, \cdots, u_V$. This sequence could be represented by a graph in which each sample u_k is associated with the corresponding node k, and there is a directed edge from node k to its successor, node $k+1$ (unless $k = V$, in which case there is an edge connecting it to node 1). This concept is depicted in Fig. 9.

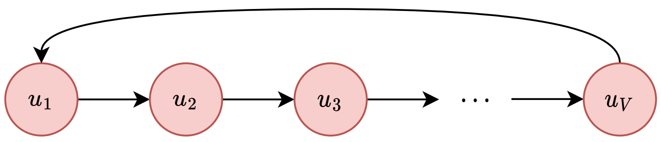


Figure 9: A discrete periodic time sequence represented as a graph.

Assigning unit weights to the edges of the graph of Fig. 9, its adjacency matrix is given by [160]

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

Collecting the samples of the sequence into a vector $\mathbf{u} = [u_1\ u_2\ \cdots\ u_V]$, we observe that by left-multiplying $\mathbf{u}$ by $\mathbf{A}$, we obtain the vector $\bar{\mathbf{u}} = \mathbf{A}\mathbf{u} = [u_V\ u_1\ \cdots\ u_2]$. Taking into account that $\mathbf{u}$ is a vector representation of a periodic discrete time sequence of finite length, $\bar{\mathbf{u}}$ can be understood as a version of $\mathbf{u}$ delayed by one time unit. Thus, the adjacency matrix fulfills a role similar to the time delay operator in discrete time signal processing. Although this analogy is based specifically on the graph of Fig. 9, this notion is generalized for any adjacency matrix $\mathbf{A}$ in the GSP framework. In the general case, it can be understood that the multiplication of a graph signal $\mathbf{u}$ by $\mathbf{A}$ represents a spatial shift along the graph. For this reason, the adjacency matrix is considered a *graph-shift operator*. Over time, the usage of other matrices as graph shift operators was proposed in the literature, such as the Laplacian matrix [47, 48]. Although the analogy with discrete-time signal processing is less clear in this case, it allows for different forms of modeling the effects of the graph on the evolution of the signals defined over it. For the sake of generality, we denote the graph shift operator by $\mathbf{A}$, but we assume that it can represent any suitable choice for this operator, not just the adjacency matrix.

Thus, we can define a *linear shift-invariant graph filter* as a system given by [160]

$$\mathbf{H} \triangleq \sum_{m=0}^{M-1} w_m^{\text{o}} \mathbf{A}^m, \tag{2.46}$$

where M is the filter length and $\{w_m^{\text{o}}\}_{m=0}^{M-1}$ denotes its coefficients. Hence, if a graph signal $\mathbf{u}(n)$ is processed by this system, its output $\mathbf{h}(n)$ can be described as [160]

$$\mathbf{h}(n) = \sum_{m=0}^{M-1} w_m^{\text{o}} \mathbf{A}^m \mathbf{u}(n), \tag{2.47}$$

as illustrated in Fig. 10.

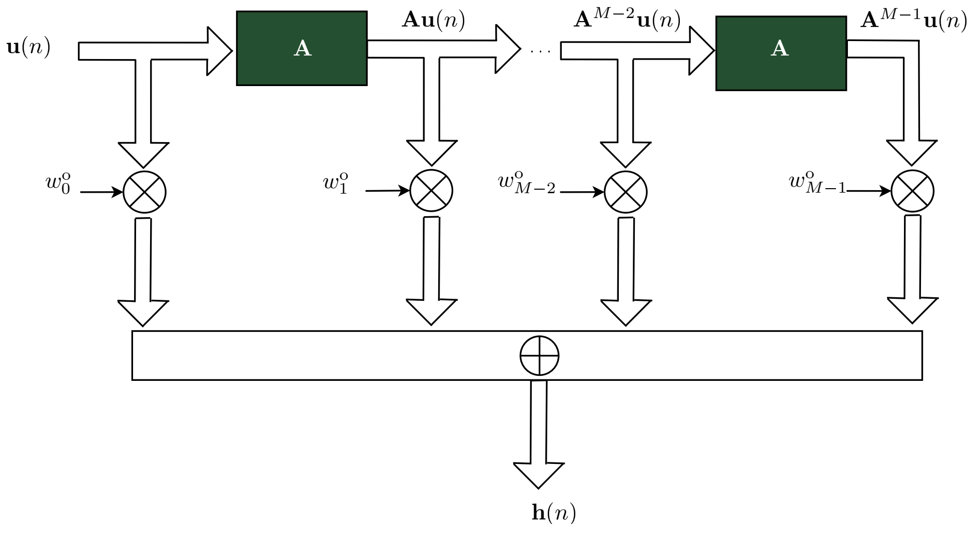


Figure 10: Schematic representation of a linear shift-invariant graph filter whose output is given by (2.47). The output of each block with the matrix $\mathbf{A}$, having the vector $\mathbf{u}(n)$ as its input, is given by the left-multiplication of $\mathbf{u}(n)$ by $\mathbf{A}$, i.e., by the vector $\mathbf{Au}(n)$.

We should notice that there is a clear analogy to the tapped delay line commonly found in discrete-time filters [125]. This motivated the proposal of adaptive filters for GSP [47,48] based on this concept, which will be explained in Sec. 2.2.6.2

2.2.6.2 Diffusion Algorithms for Graph Signal Processing

Let us consider a graph with a predefined topology and V nodes labeled $1, \cdots, k, \cdots, V$. Each node k has access at each time instant n to an input signal $u_k(n)$ and to a desired signal $d_k(n)$, modeled as [39,47,48,244,245]

$$d_k(n) = \mathbf{x}_k^{\mathrm{T}}(n)\mathbf{w}^{\mathrm{o}} + v_k(n), \tag{2.48}$$

where $v_k(n)$ is the measurement noise at node k, $\mathbf{w}^{\mathrm{o}}$ is the optimal system, and the input $\mathbf{x}_k(n)$ is given by [39,47,48,244,245]

$$\mathbf{x}_k(n) = \Big[[\mathbf{u}(n)]_k \; [\mathbf{A}\mathbf{u}(n-1)]_k \; \cdots \; [\mathbf{A}^{M-1}\mathbf{u}(n-M+1)]_k\Big]^{\mathrm{T}}. \tag{2.49}$$

Eq. (2.49) can be interpreted as follows. The vector $\mathbf{u}(n)$ represents the "raw" information available at each node of the network at the iteration n, whereas $\mathbf{x}_k(n)$ models the spreading of that information throughout the graph, which is the result of both a temporal and spatial shift, or "delay."

In light of (2.49), it may be interesting to compare (2.48) with (2.1). In the model of Eq. (2.1), only the input signal $u_k(n)$ influences $d_k(n)$, whereas in the model of Eq. (2.48), $d_k(n)$ is influenced by $u_k(n)$, by $u_i(n)$ for $i \in \mathcal{N}_k$, by the input signal at the two-hop neighbors of node k, and so on. Hence, unlike (2.1), the model of (2.48) can be used to represent situations in which the desired signal at a node is affected by what happens around it, and, therefore, on the measurements collected by its neighbors. In other words, the defining difference between the "classical" model and the GSP framework lies in the role of the spatial aspect of the problem. In the former, the topology of the network does not influence the dynamics of the desired signal. We see from (2.1) that $d_k(n)$ depends only on the signal $u_k(n)$ and on the measurement noise $v_k(n)$, and is independent of $u_i(n)$ for all $i = 1, \cdots, V$, $i \neq k$. In this case, the information does not propagate from one node to another. In the context of GSP, we see from (2.48) and (2.49) that if the nodes i and k are immediate neighbors, $d_k(n)$ depends on $u_i(n-1)$, since the information from one node spreads to its neighbors over time. Moreover, if nodes j and k are two-hop neighbors (i.e., it is possible to travel from node j to node k in two hops), $d_k(n)$ also depends on $u_j(n-2)$, and so on. Hence, the topology of the network plays a major role in how the desired signal $d_k(n)$ unfolds at each node k. In its turn, the optimal system $\mathbf{w}^{\mathrm{o}}$ models how exactly the graph topology and time lag affect the spreading of information in (2.48). This makes the graph-based framework well suited for distributed problems where both time and space must be taken into consideration, as in meteorological applications [47,48,81].

Furthermore, comparing (2.48) and (2.49) with (2.47), we can see that the model presented in this section modifies the original concept of graph filter by incorporating a time lag. In this case, rather than spreading instantaneously throughout the network as in (2.47), the information takes time to arrive at other nodes, with distant locations being influenced later than closer ones.

Apart from these differences between the GSP problem from the original distributed signal processing framework, the derivation of the dLMS algorithm for GSP in [47] follows an analogous path to what was done in Sec. 2.2. Replacing $\mathbf{u}_k(n)$ with $\mathbf{x}_k(n)$ in (2.3) and considering (2.48), the adoption of an approach of the LMS type to the minimization of (2.8) with an ATC configuration leads to [47,48,244,245]

$$\begin{cases} \boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) + \mu_k[d_k(n) - \mathbf{x}_k^{\mathrm{T}}(n)\mathbf{w}_k(n-1)]\mathbf{x}_k(n) & \text{(2.50a)} \\ \mathbf{w}_k(n) = \sum_{i \in \mathcal{N}_k} c_{ik}\boldsymbol{\psi}_i(n). & \text{(2.50b)} \end{cases}$$

Eqs. (2.50a) and (2.50b) form the basis for the ATC dLMS algorithm for GSP. For ease of reference, we shall refer to the algorithm of (2.50) as the diffusion Graph LMS (dGLMS) algorithm, whose pseudo-code description is presented in Algorithm 4. Comparing (2.50) with (2.12), we see that there is a clear analogy between these solutions, with the main difference residing on the meaning of their inputs. Furthermore, it is important to notice that the neighborhood $\mathcal{N}_k$ of node k in (2.50b) refers to the communication network of the diffusion algorithm, which does not necessarily coincide with the graph represented by $\mathbf{A}$. For example, we may allow nodes to communicate with farther agents, whose effects on their signals are negligible. The inverse may also happen, with nodes being unable to communicate with other agents that do affect their dynamics.

Finally, it is worth mentioning that multitask [233, 244] and kernel-based [39, 246, 247] versions of graph diffusion algorithms have also been proposed in the literature. They apply analogous lines of reasoning to those expounded in Secs. 2.2.4 and 2.2.5 to diffusion graph adaptive filters such as (2.50). For instance, in [39], a Graph Diffusion KLMS filter with a pre-selected dictionary is proposed, as well as an RFF Graph diffusion KLMS algorithm. In this latter case, the algorithm is similar to the one described by (2.45), with the difference that the mapped vector $\mathbf{z}_k$ is calculated using $\mathbf{x}_k$ instead of $\mathbf{u}_k$ in (2.44), with $\mathbf{x}_k$ given by (2.49) [39].

Algorithm 4 The ATC dGLMS Algorithm of Eq. (2.50).

```
1: % Initialization - for each node $k = 1, \cdots, V$, select a step size $\mu_k$ and combination weights
   $c_{ik}$ satisfying (2.10) for $i = 1, \cdots, V$, and set $\boldsymbol{\psi}_k(0) \leftarrow \mathbf{0}_M$ and $\mathbf{w}_k(0) \leftarrow \mathbf{0}_M$
2: for $n = 1, 2, \cdots$ do
3:     % Adaptation Step
4:     for $k = 1, \cdots, V$ do
5:         Update $\mathbf{u}_k(n)$
6:         % Calculating the vector $\mathbf{x}_k$:
7:         $\mathbf{x}_k(n) \leftarrow \big[[\mathbf{u}(n)]_k\ [\mathbf{A}\mathbf{u}(n-1)]_k\ \cdots\ [\mathbf{A}^{M-1}\mathbf{u}(n-M+1)]_k\big]^{\mathrm{T}}$.
8:         % Adapting the local estimate $\boldsymbol{\psi}_k(n)$:
9:         $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1) + \mu_k[d_k(n) - \mathbf{x}_k^{\mathrm{T}}(n)\mathbf{w}_k(n-1)]\mathbf{x}_k(n)$
10:    end for
11:    % The nodes transmit their local estimates $\boldsymbol{\psi}$ to their neighbors
12:    % Combination Step
13:    for $k = 1, \cdots, V$ do
14:        % Forming the combined estimate $\mathbf{w}_k(n)$:
15:        $\mathbf{w}_k(n) \leftarrow \mathbf{0}_M$
16:        for $i \in \mathcal{N}_k$ do
17:            $\mathbf{w}_k(n) \leftarrow \mathbf{w}_k(n) + c_{ik}\boldsymbol{\psi}_i(n)$
18:        end for
19:    end for
20: end for
```

Besides the dLMS of (2.12), other types of diffusion algorithms can also be extended to the system identification problem in GSP. For example, in [48], building from (2.50), an LMS-Newton type of diffusion algorithm was proposed to improve the convergence rate of the solution.

2.2.7 Application Example: Temperature Prediction

In this section, we consider a temperature dataset of daily average measurements (in °F) from 12/25/2001 to 12/21/2012 at $V = 100$ weather stations across Brazil [248]. We consider that each station corresponds to a node of a network. To determine the communication links, we applied the following procedure. Firstly, we represent the network as a directed weighted graph in which each node k is connected to the six nearest stations. Denoting this set by $\mathcal{N}_{\mathbf{A}_k}$, each element $[\mathbf{A}]_{kj}$ of the adjacency matrix $\mathbf{A}$ is given by [47]

$$[\mathbf{A}]_{kj} = \begin{cases} \dfrac{\exp(-g_{kj}^2)}{\displaystyle\sum_{\ell \in \mathcal{N}_{\mathbf{A}_k}} \exp(-g_{\ell k}^2) \sum_{i \in \mathcal{N}_{\mathbf{A}_j}} \exp(-g_{ji}^2)}, \text{if } j \in \mathcal{N}_{\mathbf{A}_k} \\ 0, \text{ otherwise} \end{cases}, \qquad (2.51)$$

where g_{kj} is the geodesical distance between nodes k and j. Thus, we consider that nodes k and j can communicate if $k \in \mathcal{N}_{\mathbf{A}_j}$ and $j \in \mathcal{N}_{\mathbf{A}_k}$ simultaneously. The resulting network is depicted in Fig. 11a, along with the temperature measured in each station on 06/21/2002.

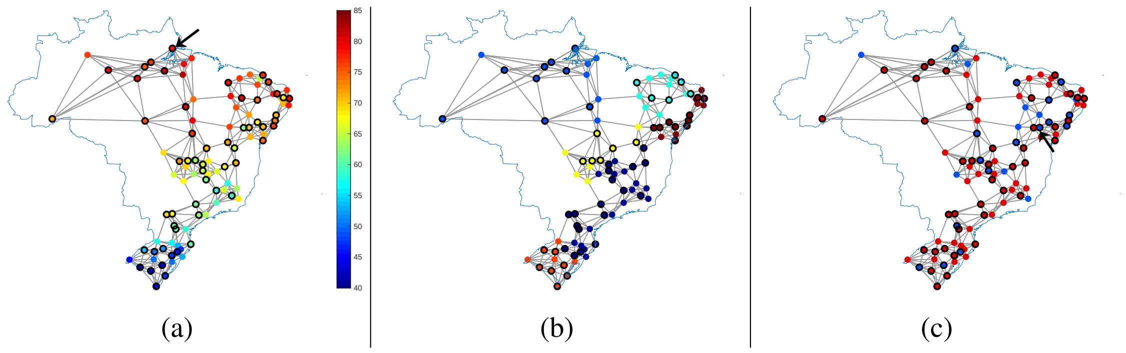


Figure 11: Network used in the simulations of Secs. 2.2.7.1 and 2.2.7.2. Edges represent communication links. The nodes that are circled in black use a normalized step size $\widetilde{\mu}_k = 1$, whereas the others use $\widetilde{\mu}_k = 0.1$ in (2.53). (a) Daily average temperature measured by 100 weather stations on 06/21/2002 (°F). The arrow points to the station whose data are used in Fig. 13. (b) Clusters adopted for the multitask algorithms in Secs. 2.2.7.1 and 2.2.7.2. (c) Scenario studied in Sec. 2.2.7.2. Blue nodes are unobserved, whereas the red ones are observed. Edges represent communication links. The arrow points to the station whose data are used in Fig. 15. This figure was created using the GSPBOX toolkit [249].

We divided our dataset into training and testing sets. The former consists of $N_{\text{tr.}} = 3650$ measurements from 12/25/2001 to 12/22/2011, which were periodically replicated to form a number of training epochs, depending on the experiment. The testing set consists of the measurements from 12/23/2011 to 12/21/2012. In both periods, we consider that $d_k(n) = u_k(n+1)$, where $u_k(n)$ denotes the temperature measurement at node k and time instant n. Hence, the simulation scenario could be interpreted as follows. At the dawn of each day, the algorithms predict the average temperature for that date at all the nodes using the last measurements available and the combined estimate calculated at the end of the previous day. At the end of the day, after the average temperature for that certain date has been calculated, the algorithms compute their estimation errors and perform their adaptation and combination steps to update their parameters $\boldsymbol{\psi}_k$ and $\mathbf{w}_k$.

In both the training and testing periods, we conduct the adaptation of the algorithms as usual. The idea behind the division of the data in these sets is to enable us to test the algorithms using data different from the ones employed in the training, and also to examine the steady-state performance along 365 days. As a performance indicator, we adopt the squared relative reconstruction error (SSRE), given by [47]

$$\mathrm{SRRE} = \frac{1}{V}\sum_{k=1}^{V}\frac{\left[u_k(n+1)-\mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n)\right]^2}{u_k^2(n+1)}. \tag{2.52}$$

We consider two types of application. In Sec. 2.2.7.1, we study the behavior of the techniques studied in Secs. 2.2–2.2.6 in a scenario in which the temperature is measured daily at every node. On the other hand, in Sec. 2.2.7.2, we consider that some of the nodes are not capable of measuring the temperature. Thus, the data remain unobserved at these nodes throughout the training and testing sets. This model could be used to, e.g., predict the temperature in a region where there are no weather stations. To cope with the lack of information at these nodes, we consider different diffusion techniques that utilize the GSP framework.

2.2.7.1 Temporal Prediction with Fully Observed Nodes

In this scenario, we use the dNLMS, RFF-dKNLMS, multitask dNLMS, and the diffusion Graph NLMS (dGNLMS) algorithms to predict the current temperature at every station, assuming that we have access to a few past measurements at every node. We opted to use the normalized version of each algorithm due to the fact that the input signal is not white in this scenario, which could hinder the performance otherwise [125, 126]. Hence, each node k uses

$$\mu_k(n) = \frac{\widetilde{\mu}_k}{\delta_r + \left\|\mathbf{y}_k(n)\right\|^2}, \tag{2.53}$$

where $\delta_r > 0$ is a small regularization factor, and $\mathbf{y}_k(n) = \mathbf{u}_k(n)$ for the single-task and multitask dNLMS algorithms, $\mathbf{y}_k(n) = \mathbf{z}_k(n)$ for RFF-dKNLMS, and $\mathbf{y}_k(n) = \mathbf{x}_k(n)$ for the dGNLMS algorithm. We adopted $\widetilde{\mu}_k = 0.1$ for half of the nodes, and $\widetilde{\mu}_k = 1$ for the other half. This is depicted in Fig. 11, in which the nodes that use $\widetilde{\mu}_k = 1$ are circled in black, whereas the ones that use $\widetilde{\mu}_k = 0.1$ are not. For every node, we set $\delta_r = 10^{-5}$. Furthermore, instead of directly using $u_k(n)$ as the input signal for each node k, we divided by the highest temperature measured registered during the training period, so as to ensure that $|u_k(n)| \leqslant 1$ for every k, $k = 1, \cdots, V$. Later, the outputs of the algorithms were re-scaled. In every case, we adopted the ATC configuration, and the ACW algorithm of Table 1 for the selection of the combination weights $c_{ik}(n)$, with $\nu_{k_{\mathrm{ACW}}} = 0.2$. To avoid potential problems due to a division by zero in the ACW algorithm, we also added a regularization factor δ_c to $\widehat{\sigma}_{ik}^2(n)$ before calculating its reciprocal in (2.23). In this experiment, we set $\delta_c = \delta_r = 10^{-5}$. We also set $M = 5$ for every solution. For the

RFF-dKNLMS algorithm, we used the Gaussian kernel with $D = 20$. Increasing D further did not improve the performance significantly, while raising the computational cost noticeably. Furthermore, the value of $h^2 = 0.1$ was selected because it led to the best steady-state performance. For the multitask case, we considered the approach of [20] with $\eta = 0$ and $b_{ik} = 1$ if $i = k$ and $b_{ik} = 0$ otherwise. Hence, the nodes do not exchange information during the adaptation step, only in the combination step, and nodes from different clusters do not communicate with each other. In order to determine the clusters, we calculated the Wiener solution for each node and ran the k-means algorithm. Then, a few manual adjustments were made to ensure that nodes belonging to the same clusters are indeed neighbors. The resulting clusters are depicted in Fig. 11b. For the dGNLMS algorithm, we considered the adjacency matrix given by (2.51) as the graph shift operator. This matrix was normalized by its greatest absolute eigenvalue, which is a common practice in the literature [47, 48]. Finally, for comparison, we also included the noncooperative NLMS approach in the simulations. In the training period, we consider a total of 40 epochs.

In Fig. 12 we show the SRRE curves yielded by the solutions in the training period. For the sake of visualization, these curves were filtered by a moving-average filter with 2048 coefficients. In Table 2, we present the average SRRE obtained with each solution during the testing period, as well as the approximate number of multiplications required per iteration by each solution. From Fig. 12 and Table 2, we can see that the noncooperative NLMS achieves a higher level of SRRE in steady state in comparison with all of the other solutions. The dNLMS, dGNLMS, and multitask dNLMS algorithms achieved similar performances, whereas the RFF-dKNLMS algorithm slightly outperforms them in steady-state and in the testing period. Nonetheless, it is worth noting that this improvement comes at the expense of a greater computational cost, as is evident from Table 2. Furthermore, the usage of information from neighboring nodes in the dGNLMS algorithm did not improve the performance in this particular scenario. Analyzing the behavior of the single-task and multitask dNLMS algorithms, we can see that, in this case, the communication solely within each cluster was sufficient to achieve the same performance in comparison with the case in which each node was allowed to exchange information with all of its neighbors. In other words, the multitask dNLMS algorithm achieved a similar performance in comparison with the single-task dNLMS while presenting lower communication and computational burdens, as shown in Table 2.

Lastly, in order to enable a direct comparison in terms of degrees Fahrenheit, in Fig. 13, we

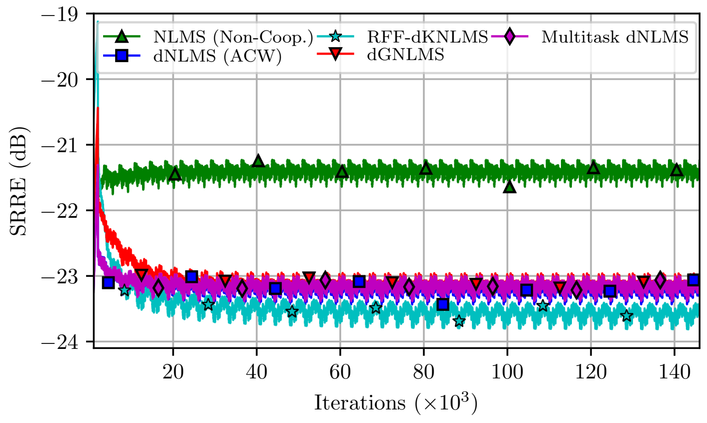


Figure 12: SRRE curves along the iterations for the training dataset obtained with the dNLMS, RFF-dKNLMS, dGNLMS, multitask dNLMS, and noncooperative NLMS algorithms.

Table 2: Comparison between the dNLMS, RFF-dKNLMS, dGNLMS, multitask dNLMS, and noncooperative NLMS algorithms in terms of the performance on the testing set and computational cost.

Solutions	Average SRRE in the testing set (dB)	Approximate number of multiplications per iteration ($\times 10^3$)
NLMS (noncooperative)	−20.6910	3.99
dNLMS (2.12)	−22.8933	7.90
Multitask dNLMS (2.29)	−22.8338	6.48
RFF-dKNLMS (2.45)	−23.5502	39.54
dGNLMS (2.50)	−22.8725	14.10

illustrate the behavior of the solutions by showing the estimates provided by each of them for the temperature at a single node, which is indicated by an arrow in Fig. 11a, in the testing set. The actual temperature, i.e. the desired signal, is also shown. We can see that the noncooperative approach produces a noisier estimate in comparison with the other solutions, which match the desired signal reasonably well.

2.2.7.2 Prediction at Unobserved Nodes

In this subsection, we consider that there are 75 observed nodes, which can measure their data daily, and 25 unobserved nodes, which are not capable of performing that measurement at all. This is depicted in Fig. 11c, in which observed nodes are shown in red, and unobserved ones are depicted in blue. This situation could be used to model a scenario in which we would like to estimate the temperature in a region in which there is no weather station, based on the measurements from the surrounding region.

In order to predict the temperature at unobserved nodes, we must rely on the information at their neighbors. To this end, we employ the dGNLMS algorithm, similarly to what was done in

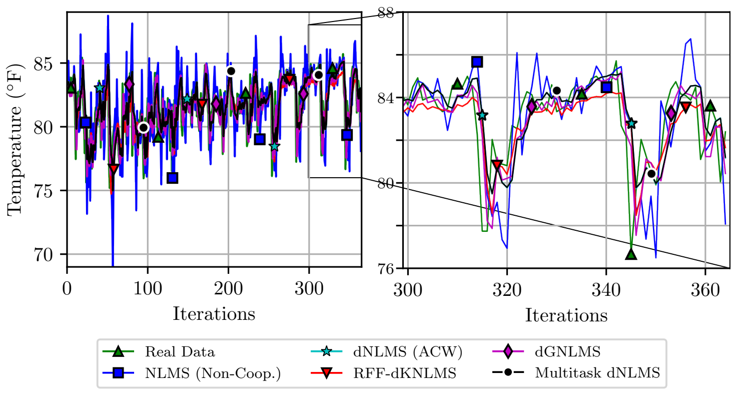


Figure 13: Comparison between the temperature measured at the station indicated by an arrow in Fig. 11 and the estimates provided by the dNLMS, RFF-dKNLMS, dGNLMS, multitask dNLMS, and noncooperative NLMS algorithms.

Sec. 2.2.7.1, as well as GSP-based versions of the multitask and kernel solutions of Secs. 2.2.4 and 2.2.5, respectively.

To cope with the unavailability of the information at the unobserved nodes, the vector $\mathbf{x}_k(n)$ of (2.49) is slightly modified to [244]

$$\mathbf{x}'_k(n) = \left[[\mathbf{A}\mathbf{S}\mathbf{u}(n)]_k \cdots [\mathbf{A}^M\mathbf{S}\mathbf{u}(n-M+1)]_k\right]^{\mathrm{T}}, \tag{2.54}$$

where $\mathbf{S}$ is a diagonal matrix such that $[\mathbf{S}]_{kk} = 1$ if node k is observed and $[\mathbf{S}]_{kk} = 0$ otherwise.

We employ a normalized version of the RFF kernel solution of [246], resulting in the RFF-dGKNLMS algorithm. For the multitask approach, we consider the solution of (2.29) with the same clusters that were used in the simulations of Sec. 2.2.7.1. In comparison with the original solutions of (2.29) and (2.45), the main differences between the algorithms employed in this section and the ones studied previously lie in the usage of the vector given by (2.54) instead of the regressor vector $\mathbf{u}_k(n)$ in the adaptation step, and in the adoption of a normalized step size as in (2.53). It is worth noting that an unsupervised clustering algorithm for multitask diffusion solutions was proposed in [244]. Based on this method, several multitask algorithms were proposed in the aforementioned paper specifically for diffusion adaptive graph filtering. Due to space limitations, and since in this case we can group the nodes in clusters based on the knowledge of the Wiener solution, we have opted to restrict our analysis to the modified version of (2.29). Nonetheless, we encourage the reader interested in distributed GSP to consult [244] for the multitask solutions with unsupervised clustering.

Similarly to what was done in Sec. 2.2.7.1, we adopt a length of $M = 5$ for $\mathbf{u}_k(n)$. We also use normalized step size $\widetilde{\mu}_k = 1$ for half of the nodes, whereas the other half employs $\widetilde{\mu}_k = 0.1$, as shown in Fig. 11b. Furthermore, we consider an ATC configuration and seek to obtain the best performance possible with each solution. To this end, we have adopted ACW to select the weights of the dGNLMS and multitask dGNLMS algorithms. For the latter, we adopted $\eta = 0$ and $b_{ik} = c_{ki}$ for every k, $k = 1, \cdots, 100$ and i, $i = 1, \cdots, 100$. Adopting $\eta \neq 0$ with weights $\bar{\rho}_{ik}$ did not affect the performance significantly. For the RFF-dGKNLMS algorithm, we adopted $h^2 = 10$ and $D = 20$. Increasing D further did not improve the results noticeably. Finally, specifically for the RFF-dGKNLMS algorithm we adopted Metropolis weights, since the adoption of ACW did not improve the performance significantly, while increasing a computational cost that was already comparatively high. We also tested multitask versions of RFF-dGKNLMS following different configurations for the weights b_{ik}, c_{ik}, $\bar{\rho}_{ik}$ and different values for η, but did not observe any significant improvement in comparison with the single-task RFF-dGKNLMS algorithm. For this reason, these results are not depicted in the following figures and tables.

We consider 20 epochs in the training set, and evaluate the performance of each solution by applying the SRRE given by (2.52) to the unobserved nodes only. The resulting SRRE curves are presented in Fig. 14. For the sake of visualization, these curves were filtered by a moving-average filter with 1024 coefficients. In Table 3, we present the average SRRE measured at the unobserved nodes in the testing period, as well as the number of multiplications required by each solution. Analyzing Fig. 14, we can see that the multitask dGNLMS outperforms the single-task dGNLMS algorithm, but both are outperformed by the RFF-GKNLMS solution. The same holds in the testing set, as evidenced in Table 3. On the other hand, we can see from the same table that the computational cost of RFF-dGKNLMS is considerably greater than that of the other solutions. It is interesting to notice that the multitask dGNLMS outperforms the single-task version while presenting only a slightly higher computational burden.

Table 3: Comparison between the dGNLMS, multitask dGNLMS, and RFF-dGKNLMS algorithms in terms of the performance the testing set and computational cost.

Solutions	Average SRRE in the testing set (dB)	Approximate number of multiplications per iteration ($\times 10^3$)
dGNLMS	-20.2872	17.17
Multitask dGNLMS (2.29)	-23.8701	20.14
RFF-dGKNLMS (2.45)	-26.2089	299.47

Finally, similarly to what was done in Fig. 13, in Fig. 15, we show the estimates provided by the dGNLMS, multitask dGNLMS, and RFF-dGKNLMS algorithms for the temperature at the

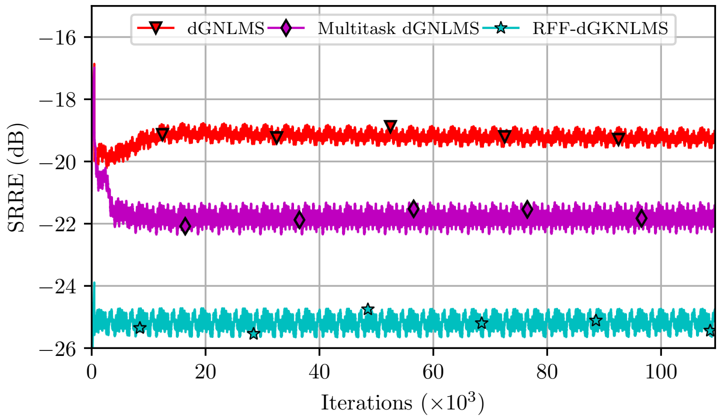


Figure 14: SRRE curves for the unobserved nodes along the iterations for the training dataset obtained with the dGNLMS, multitask dGNLMS, and RFF-dGKNLMS algorithms.

unobserved node indicated by an arrow in Fig. 11b in the testing set. We also show the actual temperature measured at that station. We can see that the estimates match the actual temperature signal reasonably accurately. This is especially true for the RFF-dGKNLMS algorithm, which supports the results of Fig. 14 and Table 3.

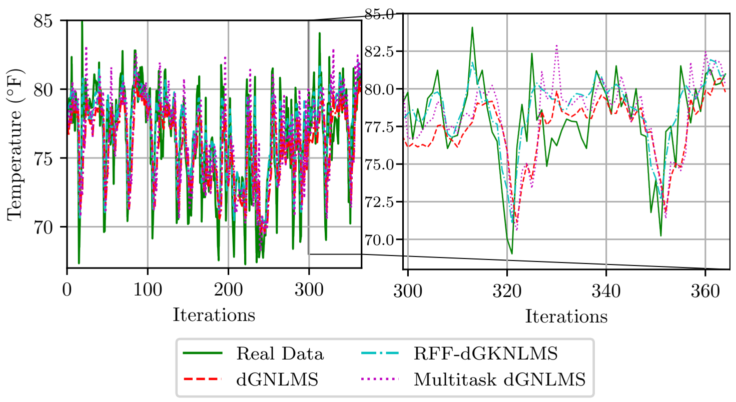


Figure 15: Comparison between the temperature measured at the station indicated by an arrow in Fig. 11b and the estimates provided by the dGNLMS, multitask dGNLMS, and RFF-dGKNLMS algorithms.

2.3 Conclusions

Starting with the technological developments that motivated the emergence of adaptive diffusion networks, we have presented the progress that has occurred in the area throughout the past fifteen years or so. Some theoretical results were presented in order to provide some insights into their behavior. On the other hand, a few considerations were presented regarding the feasibility of these solutions in practical applications. In particular, significant attention was devoted to the necessity of restricting the amount of communication between nodes, even if at the expense of some deterioration in performance, due to energy constraints. We have also looked at advances that have been proposed in the literature and developed into mature research topics in their own right, such as the multitask and kernel-based diffusion networks. Furthermore, diffusion-based adaptive algorithms for Graph Signal Processing were also addressed, and the similarities and differences in comparison with the more conventional diffusion solutions were highlighted. In each case, we have highlighted what type of problems each of these solutions is intended for. Finally, simulations carried out with real-world data exemplify the opportunities and challenges that can arise from the usage of adaptive diffusion networks in practice.

Despite the maturity of adaptive diffusion networks as a research field, there are still many open questions and challenges in the area. As mentioned in Sec. 2.2.3, the search for restrictive communication policies that impact network performance as little as possible continues to spark research and publications in the area, such as, e.g., [61, 62, 82–84]. The research efforts in this area will likely continue as long as energy remains a significant constraint for battery-operated WSNs. Moreover, the computational cost of the solutions continues to be an object of concern in many scenarios. This aspect is applicable to adaptive diffusion networks as a whole, due to the potentially low computational power of each node in IoT applications [81, 99, 100, 250]. For kernel-based adaptive networks, this is an especially pressing issue, due to the high computational burden associated with these tools, as can be seen from Tables 2 and 3. The reduction of the computational cost of kernel methods has been a topic of intense research in general [251–253], as well as estimating the number of RFFs required for satisfactory performance [254]. In the GSP field, the idea of sampling the nodes – i.e., measuring the data only at a subset of nodes and seeking to estimate the behavior of the network as a whole from the acquired information – has also been a topic of constant research [255–257]. This is important because the computational cost of GSP solutions increases with the number of nodes, and may become prohibitively high if the network is too large [41, 42].

In an effort to address these concerns, in Chapter 3, we present several solutions for the adaptive sampling and censoring of diffusion networks, which seek to reduce the computational cost and energy consumption with as little negative impacts on algorithm performance as possible.

3 PROPOSED ALGORITHMS

In this chapter, we present algorithms for the sampling and censoring of adaptive diffusion networks. Moreover, theoretical analyses are presented in order to estimate the expected number of sampled or censored nodes per iteration, and simulation results are shown to illustrate the behavior of each proposed solution. These results indicate the good performance of the algorithms in comparison with other techniques found in the literature in a wide range of scenarios.

As mentioned in Chapter 1, the goal of sampling is to reduce the amount of data measured and processed by the nodes. As we shall explain in this chapter, if we say that a node is "not sampled," this means that it is exempt from measuring its desired signal and from adding the correction term in the adaptation step of diffusion algorithms. This enables a reduction in the computational cost and can lead to a slight decrease in the power consumption, since the energy required to sense the desired signal is saved. On the other hand, the goal of censoring is to restrict the number of broadcasts performed by the nodes, as explained in Sec. 2.2.3.3. Censored nodes do not transmit their local estimates to their neighbors in diffusion algorithms, allowing them to save a significant amount of energy. The techniques proposed throughout this chapter enable each node in the network to decide whether it should be sampled and or not, and then act accordingly. For the censoring techniques, an analogous line of thought holds.

This chapter is organized as follows. The first solution, named as "Adaptive-Sampling" (AS) algorithm, is presented in Sec. 3.1, along with the "Adaptive-Sampling-and-Censoring" technique. These techniques were proposed in [81], although preliminary versions had appeared in [258–260]. In Sec. 3.2, two modified versions of the AS algorithm are presented, which seek to address its main weaknesses. These modifications were proposed in [82]. In Secs. 3.3 and 3.4, we show how the algorithms of the previous sections can be applied to the RFF adaptive diffusion networks of Sec. 2.2.5, as was done in [84], and to the multitask networks of Sec. 2.2.4, as in [261], respectively. Finally, Sec. 3.5 closes the chapter with the main conclusions from the previous sections.

3.1 The Adaptive Sampling Algorithm

To facilitate the selection of the step size, we consider henceforth a normalized version of the ATC dLMS algorithm of Eqs. (2.12), with μ_k replaced by $\mu_k(n)$ given by (2.53). $\{d_k(n), u_k(n)\}$. Mathematically, the modifications that the AS-dNLMS algorithm applies to the original dNLMS can be described as follows. We begin by introducing a binary variable $\zeta_k(n) \in \{0,1\}$ in (2.12a) as

$$\boxed{\boldsymbol{\psi}_k(n+1) = \mathbf{w}_k(n) + \zeta_k(n)\mu_k(n)\mathbf{u}_k(n)e_k(n).} \tag{3.1}$$

If $\zeta_k(n) = 1$, $d_k(n)$ is sampled, $e_k(n)$ is computed as in (2.7) and (3.1) coincides with (2.12a). In contrast, if $\zeta_k(n) = 0$, $d_k(n)$ is not sampled, $\mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n)$, $e_k(n)$ and $\mu_k(n)$ are not computed, and $\boldsymbol{\psi}_k(n+1) = \mathbf{w}_k(n)$. The input signal $u_k(n)$ is always sampled, even if $\zeta_k(n) = 0$, since otherwise it would be impossible to adequately update the regressor vector $\mathbf{u}_k(n)$ at future iterations due to the missing data. We remark that the adaptive sampling mechanism can be straightforwardly extended to the GSP-based version of the diffusion algorithms by replacing $\mathbf{u}_k(n)$ by $\mathbf{x}_k(n)$ given by (2.49) or (2.54) in (3.1). In this case, the vector $\mathbf{x}_k(n)$ does not need to be computed if $\zeta_k(n) = 0$, but we still sample the input signal $u_k(n)$, as in the non-GSP approach.

Since $\zeta_k(n)$ is a binary variable, directly adapting it might be somewhat difficult. For this reason, we introduce an auxiliary continuous variable $\bar{\zeta}_k(n) \in [0,1]$, such that

$$\zeta_k(n) = \begin{cases} 1, \text{ if } \bar{\zeta}_k(n) \geqslant 0.5, \\ 0, \text{ otherwise} \end{cases} . \tag{3.2}$$

We then minimize

$$J_{\zeta,k}(n) = \left[\bar{\zeta}_k(n)\right]\beta\zeta_k(n) + \left[1 - \bar{\zeta}_k(n)\right]\sum_{i \in \mathcal{N}_k} c_{ik}(n)e_i^2(n), \tag{3.3}$$

with respect to $\bar{\zeta}_k(n)$, where $\beta > 0$ is a parameter introduced to control how much the sampling of the nodes is penalized. When the error is high in magnitude or when node k is not being sampled ($\zeta_k = 0$), $J_{\zeta,k}(n)$ is minimized by making $\bar{\zeta}_k(n)$ closer to one, leading to the sampling of node k. This ensures that the algorithm keeps sampling the nodes while the error is high and resumes the sampling of idle nodes at some point, enabling it to detect changes in the environment. In contrast, when node k is being sampled ($\zeta_k = 1$) and the error is small in magnitude in comparison to β, $J_{\zeta,k}(n)$ is minimized by making $\bar{\zeta}_k(n)$ closer to zero, which leads the algorithm to stop sampling node k. This desirable behavior depends on a proper choice for β, which is addressed in Section 3.1.2.

Inspired by convex combination of adaptive filters (see [262, 263] and their references), rather than directly adjusting $\bar{\zeta}_k(n)$, we introduce yet another auxiliary variable $\alpha_k(n)$, which will be directly updated in its stead. The variables $\bar{\zeta}_k(n)$ and $\alpha_k(n)$ are related to one another via [263]

$$\bar{\zeta}_k(n) = \phi[\alpha_k(n)] \triangleq \frac{\operatorname{sgm}[\alpha_k(n)] - \operatorname{sgm}[-\alpha^+]}{\operatorname{sgm}[\alpha^+] - \operatorname{sgm}[-\alpha^+]}, \tag{3.4}$$

where $\operatorname{sgm}[x] = (1+e^{-x})^{-1}$ is a sigmoidal function and α^+ is the maximum value α_k can assume. The function $\phi[\cdot]$ of (3.4) is a scaled and shifted version of $\operatorname{sgm}[\cdot]$. It was proposed in [263] to prevent the adaptation process from stopping if $\alpha_k(n)$ becomes too large or too negative. We should notice that $\phi[\alpha^+] = 1$, $\phi[0] = 0.5$, and $\phi[-\alpha^+] = 0$. In the literature, $\alpha^+ = 4$ is usually adopted [263]. For compactness, henceforth we shall write $\phi[\alpha_k(n)]$ simply as $\phi_k(n)$.

By taking the derivative of (3.3) with respect to $\alpha_k(n)$, we obtain the following stochastic gradient descent rule:

$$\alpha_k(n+1) = \alpha_k(n) + \mu_\zeta \phi'_k(n) \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n) e_i^2(n) - \beta \zeta_k(n) \right], \tag{3.5}$$

where $\mu_\zeta > 0$ is a step size and

$$\phi'_k(n) \triangleq \frac{d\bar{\zeta}_k(n)}{d\alpha_k(n)} = \frac{\operatorname{sgm}[\alpha_k(n)]\{1 - \operatorname{sgm}[\alpha_k(n)]\}}{\operatorname{sgm}[\alpha^+] - \operatorname{sgm}[-\alpha^+]}. \tag{3.6}$$

It is interesting to notice that although we used $\bar{\zeta}_k(n)$ in the derivation of the algorithm, it does not have to be calculated explicitly, since it does not arise in (3.9). Instead, only $\zeta_k(n)$ and $\frac{d\bar{\zeta}_k(n)}{d\alpha_k(n)}$ appear. The latter can be stored in a look-up table, and the former is related to $\alpha_k(n)$ by

$$\zeta_k(n) = \begin{cases} 1, \text{ if } \alpha_k(n) \geqslant 0, \\ 0, \text{ otherwise} \end{cases}, \tag{3.7}$$

as can be seen from (3.2) and (3.4). To ensure the sampling of the nodes in the transient phase, the idea is to initialize the sampling algorithm with $\alpha_k(0) = \alpha^+$ for $k = 1, \cdots, V$.

We remark that we do not take the partial derivative of $\zeta_k(n)$ with respect to $\alpha_k(n)$ into account in the derivation of (3.5). Otherwise, we would obtain

$$\alpha_k(n+1) = \alpha_k(n) + \mu_\zeta \left\{ \phi'_k(n) \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n) e_i^2(n) - \beta \zeta_k(n) \right] - \frac{\beta}{2} \delta[\alpha_k(n)] \right\}. \tag{3.8}$$

This is due to the fact that

$$\frac{\partial \zeta_k(n)}{\partial \alpha_k(n)} = \delta[\alpha_k(n)],$$

as can be seen from (3.7), where $\delta[\cdot]$ denotes the Dirac delta function, and $\bar{\zeta}_k(n) = \frac{1}{2}$ if $\alpha_k(n) = 0$. Therefore, if we did incorporate the term $-\frac{\beta}{2}\delta[\alpha_k(n)]$ into the algorithm, its effect would be to set $\alpha_k(n+1) = -\alpha^+$ whenever $\alpha_k(n) = 0$, due to the imposed constraint that $\alpha_k(n) \in [-\alpha^+, \alpha^+]$. However, we should notice that during a normal execution of the algorithm with $\alpha_k(0) = \alpha^+$, the probability of obtaining exactly $\alpha_k(n) = 0$ at any iteration is zero. In other words, under normal conditions, this term is equal to zero with probability one. For this reason, we have opted to neglect it in (3.5).

Equation (3.5) cannot be used for sampling since it requires the errors to be computed to decide if the nodes should be sampled or not, which is contradictory. To address this issue, we replace $e_i(n)$ in (3.5) by its latest measurement we have access to, which is denoted by $\varepsilon_i(n)$. When the node is sampled, $\varepsilon_i(n) = e_i(n)$. Otherwise, it is kept fixed, i.e., $\varepsilon_i(n) = \varepsilon_i(n-1)$. We thus obtain

$$\boxed{\alpha_k(n+1) = \alpha_k(n) + \mu_\zeta \phi'_k(n) \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n) \varepsilon_i^2(n) - \beta \zeta_k(n) \right].} \tag{3.9}$$

Equation (3.9) is the foundation of the adaptive sampling mechanism. In conjunction with (3.1), it leads to an adaptive-sampling version of the dNLMS algorithm, named as adaptive-sampling diffusion NLMS (AS-dNLMS). It is summarized as Algorithm 5. Since (3.9) depends only on the estimation error at each sampled node, the proposed sampling technique can be extended to any adaptive diffusion algorithm.

The proposed mechanism reduces the number of sampled nodes in steady state, decreasing the computational cost. If β is chosen appropriately, this reduction does not occur in the transient and the adaptive-sampling version of the algorithm maintains the same convergence rate as that of the original with no sampling mechanism. This comes at the expense of a slight increase of the cost during the transient, since the sampling algorithm requires the computation of an additional update equation per node per iteration. Furthermore, we should mention that when

Algorithm 5 The ATC AS-dNLMS Algorithm.

```
1: % Initialization - for each node $k = 1, \cdots, V$, select a step size $\widetilde{\mu}_k$, a regularization factor $\delta_r$, and combination weights $c_{ik}$ satisfying (2.10) for $i = 1, \cdots, V$, and set $\boldsymbol{\psi}_k(0) \leftarrow \mathbf{0}_M$, $\mathbf{w}_k(0) \leftarrow \mathbf{0}_M$, $\alpha_k(0) \leftarrow \alpha^+$, $\zeta_k(0) \leftarrow 1$, and $\varepsilon_k(n) \leftarrow 0$
2: for $n = 1, 2, \cdots$ do
3:     % Adaptation Step
4:     for $k = 1, \cdots, V$ do
5:         if $\alpha_k(n) \geqslant 0$ then
6:             $\zeta_k(n) \leftarrow 1$
7:         else
8:             $\zeta_k(n) \leftarrow 0$
9:         end if
10:        if $\zeta_k(n) = 1$ then
11:            Update $\mathbf{u}_k(n)$
12:            % Calculating the estimation error:
13:            $e_k(n) \leftarrow d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n-1)$
14:            % Updating $\varepsilon_k$:
15:            $\varepsilon_k(n) \leftarrow e_k(n)$
16:            % Calculating the normalized step size $\mu_k(n)$:
17:            $\mu_k(n) \leftarrow \frac{\widetilde{\mu}_k}{\delta_r + \|\mathbf{u}_k(n)\|^2}$
18:            % Adapting the local estimate $\boldsymbol{\psi}_k(n)$:
19:            $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1) + \mu_k(n)e_k(n)\mathbf{u}_k(n)$
20:        else
21:            $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1)$
22:        end if
23:    end for
24:    % The nodes $\boldsymbol{\psi}$ and $\varepsilon$ to their neighbors
25:    % Combination Step
26:    for $k = 1, \cdots, V$ do
27:        % Forming the combined estimate $\mathbf{w}_k(n)$ and updating $\alpha_k$:
28:        $\mathbf{w}_k(n) \leftarrow \mathbf{0}_M$
29:        $\alpha_k(n+1) \leftarrow \alpha_k(n)$
30:        for $i \in \mathcal{N}_k$ do
31:            $\mathbf{w}_k(n) \leftarrow \mathbf{w}_k(n) + c_{ik}\boldsymbol{\psi}_i(n)$
32:            $\alpha_k(n+1) \leftarrow \alpha_k(n) + \mu_\zeta \phi'_k(n)[c_{ik}\varepsilon_i^2(n) - \beta\zeta_k(n)]$
33:        end for
34:    end for
35: end for
```

the node i is sampled, it is required to transmit $\varepsilon_i^2(n) = e_i^2(n)$ to its neighbors. Nonetheless, this information can be sent bundled with the local estimates $\boldsymbol{\psi}_i$ so as to not increase the number of transmissions.

Finally, we remark that the Algorithm 5 can be implemented in conjunction with any rule for the selection of combination weights. If an adaptive scheme for such selection is employed, c_{ik} should be replaced by $c_{ik}(n)$, and the update equations of the combination weights should also be included in Algorithm 5. Particularly, if the ACW method of Table 1 is considered in conjunction with AS-dNLMS and the sampling of node k ceased for a long period of time, the sampling mechanism could potentially harm the update of the combination weights. This occurs since in this case $\widehat{\sigma}_{kk}$ could tend toward zero in (2.23) due to ζ_k being equal to zero in (3.1). To avoid this, for $j = k$, we replace $\boldsymbol{\psi}_j(n+1)$ in (2.23) by $\bar{\boldsymbol{\psi}}_k(n+1) \triangleq \zeta_k(n)\boldsymbol{\psi}_k(n+1) + [1-\zeta_k(n)]\bar{\boldsymbol{\psi}}_k(n)$.

3.1.1 The Adaptive Sampling Algorithm as a Censoring Strategy

With a very simple modification, the proposed adaptive sampling mechanism can also be used as a censoring strategy. This alternate version of AS-dNLMS is obtained by not updating $\boldsymbol{\psi}_k$ at all when node k is not sampled. In other words, instead of using (3.1), we apply

$$\boxed{\boldsymbol{\psi}_k(n+1) = [1-\zeta_k(n)]\boldsymbol{\psi}_k(n) + \zeta_k(n)\left[\mathbf{w}_k(n)+\mu_k(n)\mathbf{u}_k(n)e_k(n)\right].} \tag{3.10}$$

Assuming that the nodes can store past information from their neighbors, this allows us to cut the number of communications between nodes, since in this case $\boldsymbol{\psi}_k$ and ε_k remain static when $\zeta_k = 0$ and there is no need for node k to retransmit them. Thus, when node k is not sampled in this version of the algorithm, it only receives data and carries out (2.12b), and can therefore turn its transmitter off. This version of the proposed algorithm is named as adaptive-sampling-and-censoring diffusion NLMS (ASC-dNLMS), and it features a lower energy consumption as well as a computational cost reduction in comparison with the original dNLMS algorithm. Its pseudocode is identical to that of Algorithm 5, except for lines 20 and 21, which simply do not exist in this version.

3.1.2 Theoretical Analysis

In the current section, we conduct a theoretical analysis of the proposed sampling mechanism. In Sec. 3.1.3, we show how to choose β so as to ensure that the nodes cease to be sampled during steady state. Then, in Sec. 3.1.4 we study how its choice influences the expected number of sampled nodes per iteration. Finally, in Sec. 3.1.5, we analyze how fast the nodes cease to be sampled depending on the choice for μ_ζ, and how to select this parameter appropriately based on that information.

3.1.3 The parameter β and its effects on the algorithm

The parameter β plays a crucial role in the behavior of the AS-dNLMS. It influences the expected number of sampled nodes during steady state, and determines when the sampling mechanism begins to act.

Firstly, we study how to choose β so that we can ensure that every node will cease to be sampled at some point during steady state. To do so, we examine (3.9) while node k is being sampled. In this case, $\varepsilon_i^2(n)$ and $\beta\zeta_k(n)$ can be replaced by $e_i^2(n)$ and β, respectively. Then, subtracting $\alpha_k(n)$ from both sides in (3.9) and taking expectations, we get

$$\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n) \geqslant 0\} = \mu_\zeta \mathrm{E}\left\{\phi_k'(n)\left[\sum_{i\in\mathcal{N}_k} c_{ik}(n)e_i^2(n) - \beta\right]\right\}. \tag{3.11}$$

where $\Delta\alpha_k(n) \triangleq \alpha_k(n+1) - \alpha_k(n)$. To make the analysis more tractable, $\phi_k'(n)$ and the term between brackets in (3.11) are assumed to be statistically independent. Simulation results suggest it is a reasonable approximation. Furthermore, for simplicity, we consider henceforth in our analysis that the combination weights are static and deterministic. Thus, we can write

$$\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n) \geqslant 0\} = \mu_\zeta \mathrm{E}\{\phi_k'(n)\}\left[\sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{e_i^2(n)\} - \beta\right]. \tag{3.12}$$

In order to stop sampling node k, $\alpha_k(n)$ should decrease along the iterations until it becomes negative. Since $\phi_k'(n)$ is always positive, to enforce $\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n) \geqslant 0\}$ to be negative, β must satisfy

$$\beta > \sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{e_i^2(n)\}. \tag{3.13}$$

Assuming that the order of the adaptive filter is sufficient and that $\tilde{\mu}_k$, $k = 1, \cdots, V$, are chosen properly so that the gradient noise can be disregarded, it is reasonable to assume that, during steady state, $\mathrm{E}\{e_i^2(n)\} \approx \sigma_{v_i}^2$, which leads to

$$\sigma_{\min}^2 \leqslant \sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{e_i^2(n)\} \leqslant \sigma_{\max}^2, \tag{3.14}$$

where

$$\sigma_{\min}^2 \triangleq \min_i \sigma_{v_i}^2 \tag{3.15}$$

and

$$\sigma^2_{\max} \triangleq \max_i \sigma^2_{v_i} \tag{3.16}$$

for $i=1,\cdots,V$. Thus, the condition

$$\boxed{\beta > \sigma^2_{\min}} \tag{3.17}$$

is necessary (but not sufficient) if we wish to stop sampling the nodes at some point during steady state. On the other hand,

$$\boxed{\beta > \sigma^2_{\max}} \tag{3.18}$$

is a sufficient (although not necessary) condition to ensure a reduction in the number of sampled nodes. Moreover, this ensures that $\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n) \geqslant 0\} < 0$, which means that, if a certain node is being sampled, it will eventually cease to be after some iterations. This is true even though the value of $\alpha_k(n+1)$ may occasionally increase at certain time instants depending on the realization. When $\sigma^2_{\min} = \sigma^2_{\max}$, i.e., every node is subject to the same level of noise power, (3.17) and (3.18) coincide and form a necessary and sufficient condition.

Moreover, given a certain value of β, we can analyze when the sampling mechanism will begin to act in terms of the MSE. From (3.12) we observe that $\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n) \geqslant 0\} \geqslant 0$ as long as $\sum_{i\in\mathcal{N}_k} c_{ik}(n)\mathrm{E}\{e_i^2(n)\} \geqslant \beta$. Since we do not allow $\alpha_k(n)$ to become greater than α^+, we conclude that $\mathrm{E}\{\alpha_k(n)\} = \alpha^+$ for $k = 1,\cdots,V$ as long as $\mathrm{MSE}_{\min}(n) > \beta$, where $\mathrm{MSE}_{\min}(n) \triangleq \min_{i=1,\cdots,V} \mathrm{E}\{e_i^2(n)\}$. In terms of mean-square deviation, this can be translated as follows. Applying the independence theory [125] to the estimation errors $e_i(n)$, $i = 1,\cdots,V$ and assuming that $\mathbf{u}_i(n)$ is wide-sense stationary and statistically independent from $\mathbf{w}_i(n)$, after some algebraic manipulations we can conclude that $\mathrm{E}\{\alpha_k(n)\}=\alpha^+$ as long as

$$\mathrm{MSD}_{\min}(n) > \frac{\beta - \sigma^2_{\min}}{\min_{i=1,\cdots,V} \mathrm{E}\left\{\|\mathbf{u}_i(n)\|^2\right\}},$$

where $\mathrm{MSD}_{\min} \triangleq \min_{i=1,\cdots,V} \mathrm{E}\{\|\mathbf{w}^{\mathrm{o}}(n)-\mathbf{w}_j(n)\|^2\}$ denotes the lowest MSD in the network. This indicates that, in the mean, the sampling mechanism does not act as long as $\mathrm{MSD}_{\min}$ remains greater than a threshold that increases with β. Consequently, we can be sure that no node will cease to be sampled in the mean during that period. Furthermore, the higher the value of β we adopt, the sooner $\mathrm{E}\{\alpha_k(n)\}$ begins to decrease for $k = 1,\cdots,V$, and thus the sooner the nodes cease to be sampled for a fixed step size μ_ζ.

3.1.4 The expected number of sampled nodes

Based on the previous section, we can estimate upper and lower bounds for the expected number V_s of sampled nodes in steady state. For this purpose, we consider each $\zeta_k(n)$ as an independent Bernoulli random variable during steady state that is equal to one with probability p_{ζ_k} or to zero with probability $1-p_{\zeta_k}$ for $k=1,\cdots,V$, with $0\leqslant p_{\zeta_k}\leqslant 1$. Thus,

$$Vp_{\zeta_{\min}} \leqslant \mathrm{E}\{V_s\} \leqslant Vp_{\zeta_{\max}}, \tag{3.19}$$

where $p_{\zeta_{\min}}$ and $p_{\zeta_{\max}}$ are upper and lower bounds for p_{ζ_k}, $k = 1,\cdots,V$.

It is useful to note that the sampling mechanism exhibits a cyclic behavior in steady state. After all, the algorithm eventually resumes the sampling of the nodes that are not currently being sampled, and ceases to sample nodes that are currently sampled. Hence, we could approximate p_{ζ_k} by the expected "duty cycle" of the mechanism, i.e.,

$$\widehat{p}_{\zeta_k} = \frac{\breve{\eta}_k}{\breve{\eta}_k + \overline{\eta}_k}, \tag{3.20}$$

where $\breve{\eta}_k$ denotes the expected number of iterations per cycle in which node k is sampled and $\overline{\eta}_k$ is the expected number of iterations in which it is not. Since we are only interested in estimating $p_{\zeta_{\min}}$ and $p_{\zeta_{\max}}$, we do not need to evaluate (3.20) for every k. Instead, we only need to estimate upper and lower bounds for $\breve{\eta}_k$ and $\overline{\eta}_k$, which we respectively denote by $\breve{\eta}_{\max}$, $\breve{\eta}_{\min}$, $\overline{\eta}_{\max}$ and $\overline{\eta}_{\min}$.

For the sake of brevity, in this section we omit the intermediate calculations and skip to the final results concerning the estimation of these parameters. Nonetheless, a complete demonstration is provided in Appendix A.

Assuming that we can write

$$\sigma^2_{\min} \leqslant \sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{\varepsilon_i^2(n)\} \leqslant \sigma^2_{\max} \tag{3.21}$$

for $k = 1,\cdots,V$ during steady state, we can estimate $\breve{\eta}_{\max}$ by finding the maximum number of iterations any node can remain sampled in the mean. Considering a worst-case scenario, as well as the fact that every node must be sampled at least once during each cycle, and assuming that (3.18) is satisfied, we obtain after some approximations

$$\breve{\eta}_{\max} = \max\left\{\frac{\sigma^2_{\max}}{\beta - \sigma^2_{\max}},\ 1\right\}. \tag{3.22}$$

Following an analogous procedure, the estimated lower bound $\breve{\eta}_{\min}$ of $\breve{\eta}_k$ can be obtained as

$$\breve{\eta}_{\min} = \max\left\{\frac{\sigma^2_{\min}}{\beta - \sigma^2_{\min}},\ 1\right\}. \tag{3.23}$$

Lastly, for $\overline{\eta}_{\max}$ and $\overline{\eta}_{\min}$, we respectively obtain

$$\overline{\eta}_{\max} = \max\left\{\frac{\beta - \sigma^2_{\min}}{\sigma^2_{\min}},\ 1\right\} \tag{3.24}$$

and

$$\overline{\eta}_{\min} = \max\left\{\frac{\beta - \sigma^2_{\max}}{\sigma^2_{\max}},\ 1\right\}. \tag{3.25}$$

Thus, using (3.20), we can now estimate $p_{\zeta_{\min}}$ and $p_{\zeta_{\max}}$ as

$$\widehat{p}_{\min} = \frac{\breve{\eta}_{\min}}{\breve{\eta}_{\min} + \overline{\eta}_{\max}} \tag{3.26}$$

and

$$\widehat{p}_{\max} = \frac{\breve{\eta}_{\max}}{\breve{\eta}_{\max} + \overline{\eta}_{\min}}. \tag{3.27}$$

When $\beta < 2\sigma^2_{\min}$, we observe from (3.23) and (3.24) that $\breve{\eta}_{\min} = \sigma^2_{\min}/(\beta - \sigma^2_{\min})$ and $\overline{\eta}_{\max} = 1$. On the other hand, for $\beta \geqslant 2\sigma^2_{\min}$, (3.23) and (3.24) yield $\breve{\eta}_{\min} = 1$ and $\overline{\eta}_{\max} = (\beta - \sigma^2_{\min})/\sigma^2_{\min}$, respectively. In both cases, making these replacements in (3.26), we get

$$\widehat{p}_{\min} = \frac{\sigma^2_{\min}}{\beta}. \tag{3.28}$$

Analogously, from (3.22), (3.25), and (3.27) we obtain

$$\widehat{p}_{\max} = \frac{\sigma^2_{\max}}{\beta}. \tag{3.29}$$

Thus, replacing (3.28) and (3.29) in (3.19), we finally get

$$V\frac{\sigma^2_{\min}}{\beta} \leqslant \mathrm{E}\{V_s\} \leqslant V\frac{\sigma^2_{\max}}{\beta}. \tag{3.30}$$

For $\beta < \sigma^2_{\max}$, (F.20) yields an upper bound that is greater than the total number V of nodes, which is not informative. However, we can generalize it for all $\beta > 0$ by recasting it as

$$\boxed{V \cdot \min\left\{1,\ \frac{\sigma^2_{\min}}{\beta}\right\} \leqslant \mathrm{E}\{V_s\} \leqslant V \cdot \min\left\{1,\ \frac{\sigma^2_{\max}}{\beta}\right\}.} \tag{3.31}$$

Replacing $\beta < \sigma^2_{\min}$ in (3.31) implies $\mathrm{E}\{V_s\} = V$, which agrees with (3.17) being a necessary condition to ensure a reduction in the number of sampled nodes. In fact, it is an even stronger result. Taking into account the fact that V_s is a discrete random variable that cannot be greater than V, this implies that the probability of having $V_s < V$ is zero in this case. Analogously, replacing $\beta > \sigma^2_{\max}$ we conclude that $\mathrm{E}\{V_s\} < V$, which is in accordance with (3.18) being a sufficient condition. Moreover, the higher the parameter β, the smaller the amount of nodes sampled in the mean during steady state, as expected. Since there is a trade-off between the tracking capability and the gains in terms of computational cost provided by the sampling mechanism, we should care not to choose excessively high values for β, since they can deteriorate the performance in non-stationary environments. Simulation results suggest that if $\beta \leqslant 5\sigma^2_{\max}$, the good behavior of the algorithm is maintained. Moreover, the upper and lower bounds coincide when $\sigma^2_{\min} = \sigma^2_{\max}$. Finally, the step size $\mu_\zeta > 0$ does not affect the number of sampled nodes in the mean.

3.1.5 Choosing the step size μ_ζ

In this section, we show how to choose a proper value for the parameter μ_ζ. To do so, we study how fast the nodes cease to be sampled (i.e., how fast we arrive at $\mathrm{E}\{\alpha_k(n)\} \leqslant 0$) after the algorithm's initialization with $\alpha_k(0) = \alpha^+$ for $k = 1, \cdots, V$. From (3.12) and (3.14), we can write

$$\mathrm{E}\{\Delta\alpha_k(n)\} \leqslant \mu_\zeta \mathrm{E}\{\phi'_k(n)\}(\sigma^2_{\max} - \beta). \tag{3.32}$$

Since in this case we consider $\alpha_k(n) \in [0, \alpha^+]$, approximating $\phi'[\alpha_k(n)]$ by its first-order Taylor expansion around $\alpha_k(n) = 0$ is not a suitable approach. Instead, we now approximate $\phi'_k(n)$ in that interval by a straight line that crosses the points $(0, \phi'_0)$ and $(\alpha^+, \phi'_{\alpha^+})$, in which ϕ'_0 and ϕ'_{α^+} respectively denote the value of $\phi'_k(n)$ evaluated at $\alpha_k(n) = 0$ and $\alpha_k(n) = \alpha^+$. This approximation is given by

$$\phi'_k(n) \approx \vartheta_1 \alpha_k(n) + \phi'_0, \tag{3.33}$$

where $\vartheta_1 = \dfrac{\phi'_{\alpha^+} - \phi'_0}{\alpha^+}$. For $\alpha^+ = 4$, this is a good approximation since its mean-squared error in $[0, \alpha^+]$ is of the order of 5×10^{-4}.

Replacing (3.33) in (3.32), we obtain

$$\mathrm{E}\{\alpha_k(n+1)\} \lessapprox \mathrm{E}\{\alpha_k(n)\}(1 + \vartheta_1\vartheta_2) + \phi'_0\vartheta_2, \tag{3.34}$$

where $\vartheta_2 = \mu_\zeta(\sigma^2_{\max} - \beta)$. Since we assumed $\mathrm{E}\{\alpha_k(n)\} \approx \alpha^+$ during transient, we denote the first iteration of the steady state by n_0. Then, considering $\mathrm{E}\{\alpha_k(n_0)\} \approx \alpha^+$ in (3.34) and applying it recursively, we conclude that Δn iterations after the beginning of the steady state, we should obtain

$$\mathrm{E}\{\alpha_k(n_0+\Delta n)\} \lessapprox \alpha^+(1+\vartheta_1\vartheta_2)^{\Delta n} + \phi_0'\vartheta_2 \sum_{n_i=0}^{\Delta n-1}(1+\vartheta_1\vartheta_2)^{n_i}. \tag{3.35}$$

After some algebraic manipulations, we arrive at

$$\mathrm{E}\{\alpha_k(n_0+\Delta n)\} \lessapprox \frac{(\vartheta_1\alpha^+ + \phi_0')(1+\vartheta_1\vartheta_2)^{\Delta n} - \phi_0'}{\vartheta_1}. \tag{3.36}$$

Since we are interested in studying how fast we arrive at $\mathrm{E}\{\alpha_k(n)\} \leqslant 0$ depending on our choice of μ_ζ, we set $\mathrm{E}\{\alpha_k(n_0+\Delta n)\}$ to zero in (3.36). Thus, for a desired value of Δn and $\beta > \sigma^2_{\max}$, we should choose

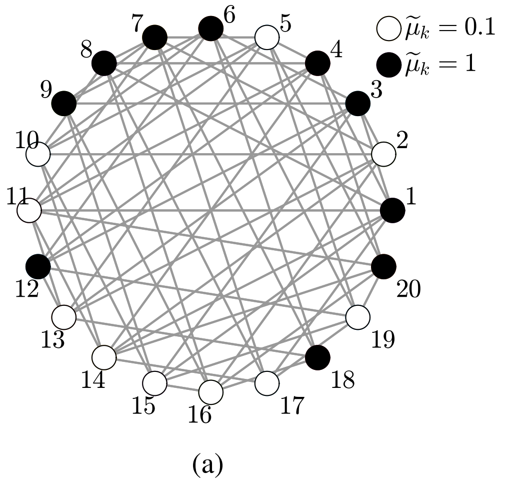
$$\boxed{\mu_\zeta > \frac{\alpha^+}{(\beta-\sigma^2_{\max})(\phi_0'-\phi_{\alpha^+}')}\left[\left(\frac{\phi_o'}{\phi_{\alpha^+}'}\right)^{\frac{1}{\Delta n}} - 1\right]}. \tag{3.37}$$

From (3.37), we observe that the smaller the Δn, the larger the value of μ_ζ, which is reasonable. Moreover, as β approaches $\sigma^2_{\max}$, (3.37) yields increasingly large values for μ_ζ. Since (3.18) is a sufficient condition, the nodes may cease to be sampled even for $\beta \leqslant \sigma^2_{\max}$. When $\beta \approx \sigma^2_{\max}$ and $\sigma^2_{\min} < \sigma^2_{\max}$, (3.37) may overestimate the value of μ_ζ required to cease the sampling of the nodes within Δn iterations. Nonetheless, this does not invalidate (3.37), since we are only interested in ensuring that the sampling will cease in at most Δn iterations in the mean.

3.1.6 Simulation Results

In this section, we present simulation results to illustrate the behavior of the proposed sampling mechanism and to validate the results of Section 3.1.2. The results presented were obtained over an average of 100 independent realizations. For better visualization, we filtered the curves by a moving-average filter with 64 coefficients, unless otherwise stated.

We consider the ATC dNLMS algorithm and a heterogeneous network with 20 nodes. Half of the them use $\widetilde{\mu}_k = 0.1$, while the other half uses $\widetilde{\mu}_k = 1$, as depicted in Fig. 16(a). The network was generated randomly according to an Erdös-Renyi model [47], and the average degree of the nodes is approximately 6.1. Furthermore, each node k is subject to a different noise variance $\sigma^2_{v_k}$, as shown in Fig. 16(b). For the optimal system $\mathbf{w}^{\mathrm{o}}$, we consider a random vector with $M = 50$ coefficients uniformly distributed in $[-1,1]$, which was the same for every realization.



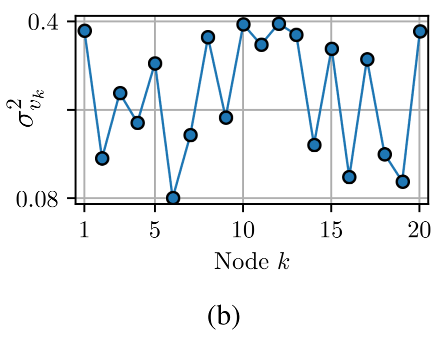


Figure 16: (a) Network used in the simulations of Sections 3.1.6.1 to 3.1.6.4. (b) Noise variance $\sigma^2_{v_k}$ for $k = 1, \cdots, V$.

We use $\delta_\mu = 10^{-5}$ as a regularization factor in (2.53) and the ACW algorithm for the update of the combination weights. To avoid potential problems due to a division by zero in the ACW algorithm, we also added the regularization factor $\delta_c = 10^{-8}$ to $\hat{\sigma}^2_{ik}(n)$ before calculating its reciprocal, similarly to what was done in Sec. 2.2.7. As a performance indicator, we adopt the NMSD. Moreover, in some situations we also analyze the network mean-square-error (NMSE), given by

$$\text{NMSE}(n) = \frac{1}{V}\sum_{k=1}^{V} \text{E}\{e_k^2(n)\}. \tag{3.38}$$

This subsection is divided as follows. In 3.1.6.1, we compare AS-dNLMS with a random sampling technique. The theoretical results of Section 3.1.2 are validated in Sec. 2.2.2, and in Sec. 3.1.6.3 we compare ASC-dNLMS to other censoring techniques. Next, in Sec. 3.1.6.4, we study the tracking capability of the proposed techniques. It is worth noting that in these subsections the input signal $u_k(n)$ is white Gaussian with $\sigma^2_{u_k}$ for $k = 1, \cdots, V$. Finally, in Sec. 3.1.6.3, we employ AS-dNLMS in the context of graph distributed adaptive filtering with the real-world temperature dataset considered in Sec. 2.2.7.

3.1.6.1 Comparison with Random Sampling

Firstly, we compare the behavior of AS-dNLMS to that of the original dNLMS with a random sampling technique and different numbers of sampled nodes $V_s \in \{5, 10, 15, 20\}$. In this case, V_s nodes are randomly selected every iteration to be sampled, whereas the remainder of the network is not sampled. It is worth noting that the case in which $V_s = 20$ corresponds to a scenario in which every node is sampled at every iteration. Moreover, we simulate a change in the environment by flipping the parameter vector $\mathbf{w}^{\mathrm{o}}$ in the middle of each realization. We adjusted AS-dNLMS to obtain approximately the same computational cost as that of dNLMS with $V_s = 5$ nodes sampled while maintaining a good performance. For this purpose, we adopted $\beta = 1.6\sigma_{\max}^2$ and $\mu_\zeta = 0.06$ based on preliminary simulation results. Figs. 17(a) and 17(b) present respectively the NMSD performance and the average number of multiplications per iteration. We observe that the more nodes are sampled during the transient, the faster the convergence rate. Moreover, we notice that AS-dNLMS is able to detect the change in the optimal system and, since all nodes are sampled during the transients, it converges as fast as the dNLMS algorithm with all nodes sampled. It is interesting to note that the sampling of less nodes per iteration leads to a slight reduction of the steady-state NMSD in comparison with the original algorithm. The dNLMS algorithm with $V_s = 5$ nodes sampled achieves a steady-state NMSD that is approximately 1.3 dB lower than the one presented by the version with all nodes sampled. In its turn, AS-dNLMS reaches a steady-state NMSD that is 0.3 dB higher than that of the algorithm with five nodes sampled, but with a much faster convergence rate. This phenomenon will be studied in Chapter 4. Thus, by reducing the sampling rate during steady state, there may be a slight reduction in the NMSD. However, we remark that this reduction can be considered marginal in the case of Fig. 17(a). From Fig. 17(b) we observe that during the transients the computational cost of AS-dNLMS is slightly higher than that of the dNLMS algorithm with all nodes sampled, but decreases significantly in steady-state. A similar behavior is observed for the number of sums.

3.1.6.2 Validation of the Theoretical Analysis

In order to validate (3.31), we also tested the AS-dNLMS algorithm in a stationary environment with different values of $\beta \geqslant \sigma_{\min}^2$ and three methods for the selection of the combination weighs: the Uniform and Metropolis rules [1], and the ACW algorithm [215]. Two scenarios were considered: one with the noise power in the network distributed as in Fig. 16b, and another where $\sigma_{v_k}^2 = 0.4$ for $k = 1, \cdots, V$. The results are shown in Fig. 18(a) and 18(b), respectively. For the ease of visualization, they are presented in terms of β_r, defined as $\beta_r \triangleq \beta/\sigma_{\max}^2$. Along with the experimental data, the predicted upper and lower bounds $\widehat{V}_{s_{\max}}$ and $\widehat{V}_{s_{\min}}$ are presented for each β_r using dashed lines. We should notice that these bounds coincide in Fig. 18(b), since $\sigma_{\min}^2 = \sigma_{\max}^2$ in this case. Moreover, in Fig. 18(a), the upper bound remains fixed at $V_{s_{\max}} = V = 20$

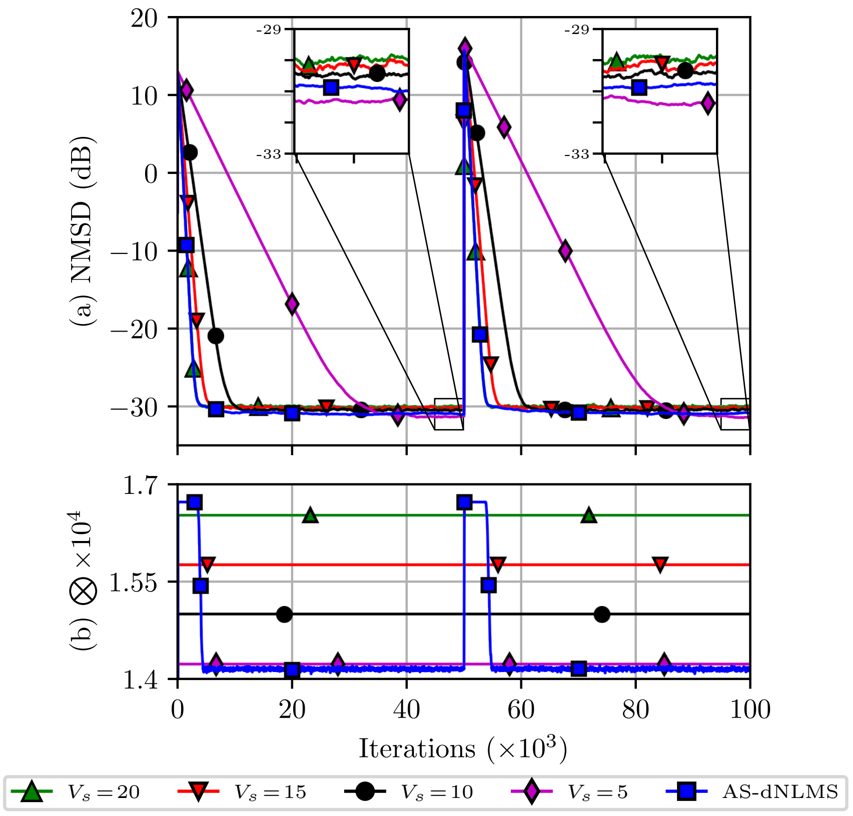


Figure 17: Comparison between dNLMS with V_s nodes randomly sampled per iteration and AS-dNLMS ($\beta = 1.6\sigma^2_{\max}$, $\mu_\zeta = 0.06$). (a) NMSD curves and (b) Multiplications per iteration.

for $\beta \leqslant \sigma^2_{\max}$. We also observe from Fig. 18 that the higher β is, the less nodes are sampled in both scenarios, as expected. Furthermore, the experimental data lie between the theoretical bounds for all combination rules and for all values of β_r in Fig. 18(a). On the other hand, from Fig. 18(b) we notice that the theoretical model slightly overestimates the number of sampled nodes for $1 < \beta_r \leqslant 20$. One possible explanation for this resides in the fact that the Assumption (3.21) translates to $\mathrm{E}\{\varepsilon_k^2(n)\} = \sigma^2_{\max}$ for $k = 1, \cdots, V$ in the case of Fig. 18(b). However, since $\zeta_k(n)$ and $\varepsilon_k(n)$ are not independent, $\mathrm{E}\{\varepsilon_k^2(n)\} \neq \sigma^2_{v_k} = \sigma^2_{\max}$. In general, we observe that $\mathrm{E}\{\varepsilon_k^2(n)\} \leqslant \sigma^2_{v_k}$, which can be attributed to ε_k remaining at a fixed value for possibly long periods of time, which tends to reduce its variance. Furthermore, the difference between the theoretical and analytical results in Fig. 18(b) is especially noticeable for the ACW algorithm. This can be attributed to the fact that the analysis presented in Section 3.1.2 was derived considering static combination weights, which is not the case of ACW. Nonetheless, the scenario considered in Fig. 18 is not realistic, since some level of noise power discrepancy across the network is expected in most environments [1, 4–6, 185]. Lastly, in both Fig. 18(a) and Fig. 18(b), the adoption of the ACW algorithm led to a smaller number of sampled nodes in comparison with the Uniform and Metropolis rules.

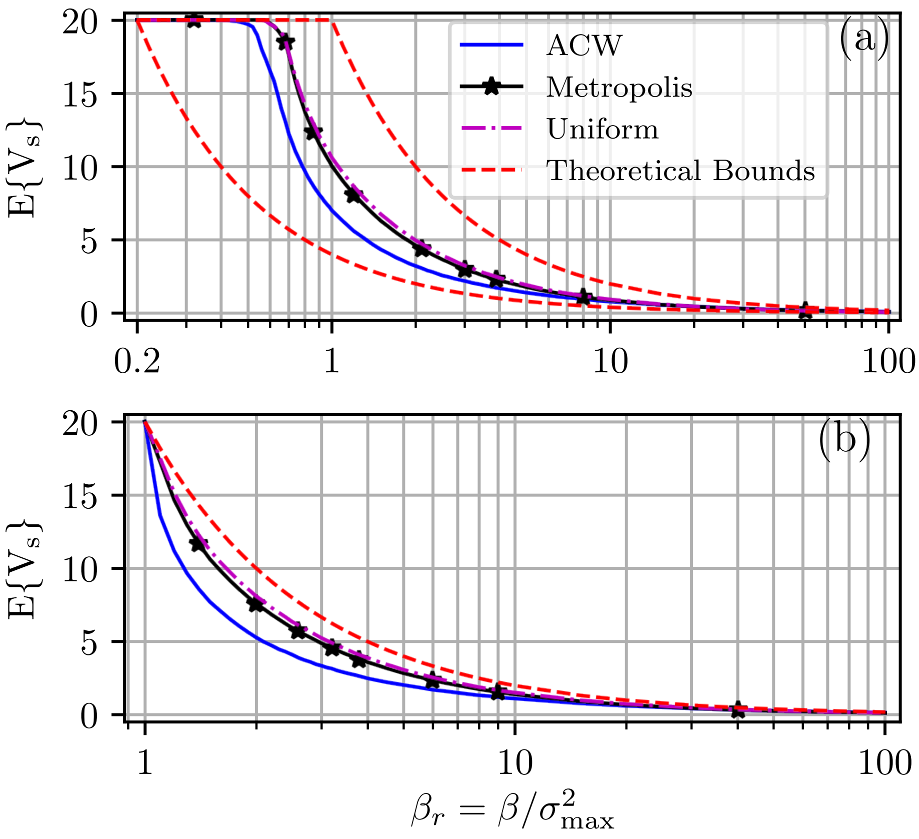


Figure 18: Theoretical bounds and average number of nodes sampled by AS-dNLMS with three combination rules as a function of $\beta \geqslant \sigma^2_{\text{min}}$. (a) $\sigma^2_{v_k}$ as in Fig. 16. (b) $\sigma^2_{v_k} = 0.4$ for $k = 1, \cdots, V$.

In Fig. 19, we test (3.37) by using it to set the step size μ_ζ for different values of β with $\Delta n = 3000$. In Fig. 19(a) we show the NMSD curves, in Fig. 19(b) the number of sampled nodes per iteration, and in Fig. 19(c) the NMSE.

From Figs. 19(b) and 19(c) we observe that, before the abrupt change in the optimal system, the number of sampled nodes stabilizes at approximately the same time for all $\beta_r > 1.1$. For $\beta_r = 1.1$, we can notice that (3.37) slightly overestimates μ_ζ. This is expected for $\beta_r \gtrapprox 1$, as discussed in Section 3.1.5. In this case, AS-dNLMS ceased to sample the nodes before reaching the steady state in terms of NMSD, which compromised the convergence rate. This illustrates the importance of a proper choice for μ_ζ as well as β. Nonetheless, since the sampling of the nodes ceased in less than Δn iterations after the beginning of the steady state in terms of NMSE, the results obtained support the validity of (3.37). However, this shows that some care must be taken when using (3.37) for $\beta \gtrapprox \sigma^2_{\text{max}}$. Lastly, we observe that the sampling of less nodes leads to a slight reduction in the steady-state NMSD, as in Fig. 17(a).

In Fig. 20 we repeated the experiments of Fig. 19 with higher values of β_r. We observe that

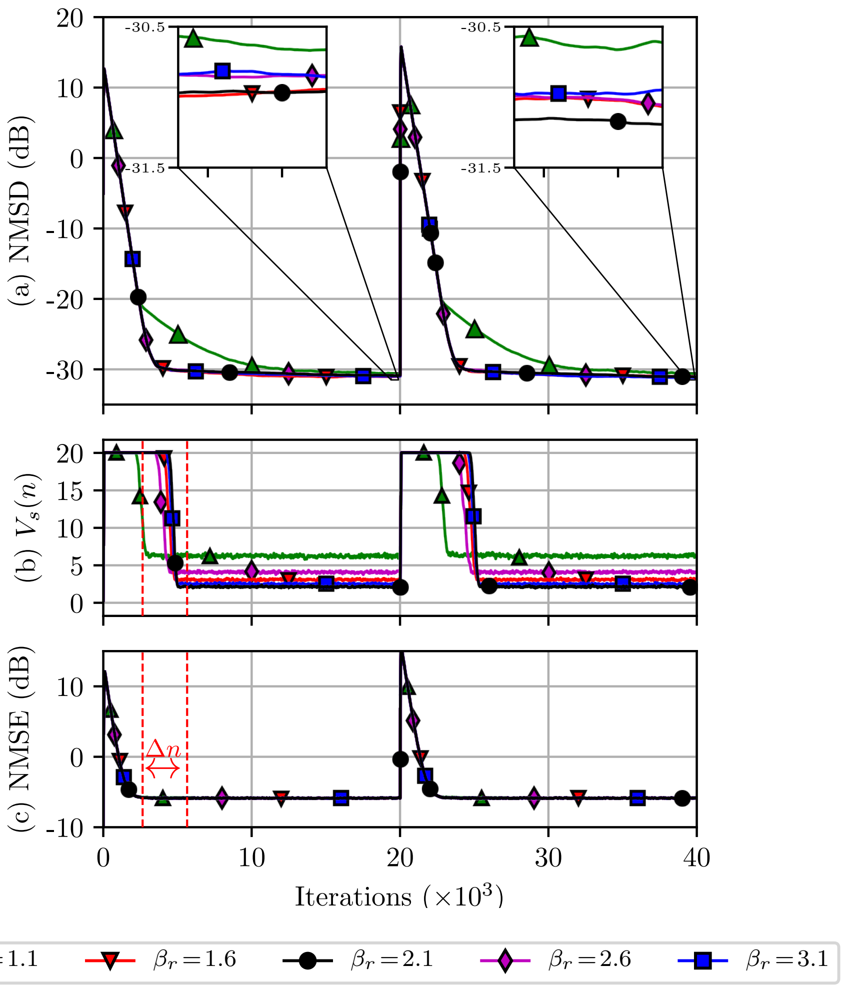


Figure 19: Simulation results obtained with $1.1\sigma^2_{\max} \leqslant \beta \leqslant 3.1\sigma^2_{\max}$ and μ_ζ adjusted by (3.37) for each case. (a) NMSD curves, (b) Number of sampled nodes per iteration, and (c) NMSE curves.

the number of sampled nodes stabilizes almost simultaneously for all values of β_r before the abrupt change and that the performance of AS-dNLMS is maintained before the change in the optimal system. Nonetheless, after the change occurs, the NMSD is affected for $\beta_r \geqslant 8$. The higher the parameter β, the more intense the deterioration in performance. The difference in the behavior of the algorithm before and after the change in the optimal system can be explained by the initialization with $\alpha_k(0) = \alpha^+$ for $k = 1, \cdots, V$. In contrast, right before the abrupt change, we have $\alpha_k(n) \ll \alpha^+$. Thus, the algorithm ceases to sample the nodes earlier in this case, as can be seen in Fig. 20(b). We recall that $\beta \leqslant 5\sigma^2_{\max}$ seems to be a safe interval for the choice of β, according to various simulations results.

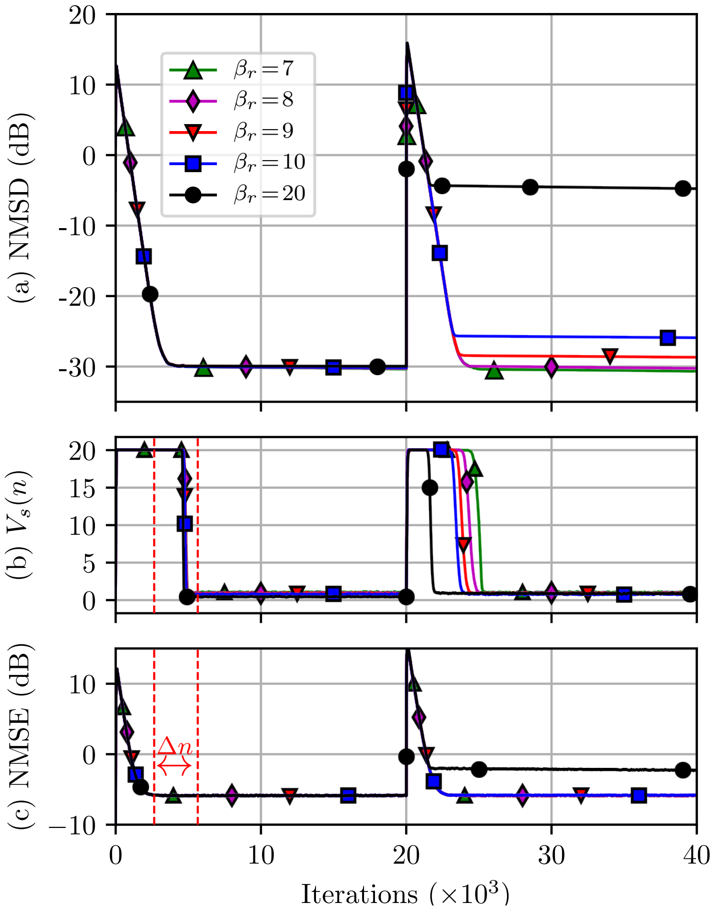


Figure 20: Simulation results obtained with $7\sigma^2_{\max} \leqslant \beta \leqslant 20\sigma^2_{\max}$ and μ_ζ adjusted by (3.37) for each case. (a) NMSD curves, (b) Number of sampled nodes per iteration, and (c) NMSE curves.

3.1.6.3 Application as a Censoring Technique

In this section, we test the ASC-dNLMS algorithm and compare it to other techniques found in the literature, namely, the ACW-Selective (ACW-S) algorithm of [72] and the energy-aware diffusion algorithm (EA-dNLMS) of [73], all arranged according to an ATC configuration. Assuming that the nodes can broadcast their data to all of their neighbors at once, we present in Fig. 21(a) the NMSD curves, in Fig. 21(b), the number $V_t(n)$ of transmitting nodes per iteration, i.e. the amount of broadcasts in the network, and in Fig. 21(c) the average number of multiplications per iteration. It is worth noting that, in the simulations of this section, we do not consider the AS-dNLMS algorithm due to the fact that this version does not restrict the amount of data transmitted by the nodes, unlike ASC-dNLMS.

The algorithms were adjusted to achieve approximately the same level of steady-state NMSD. Table 4 shows the adopted values for the parameters of each solution. For ease of reference, we have maintained the notation adopted in [72, 73] for the respective parameters of each solution. In this regard, it is worth noting that EA-dNLMS presents a high number of parameters, which may be difficult to adjust. We consider the version of EA-dNLMNS that allows node k to receive and combine the estimates from its neighbors even when it is not transmitting [73], and we adopt a normalized step size following (2.53). For comparison, we also present results obtained with the original dNLMS and with the noncooperative case.

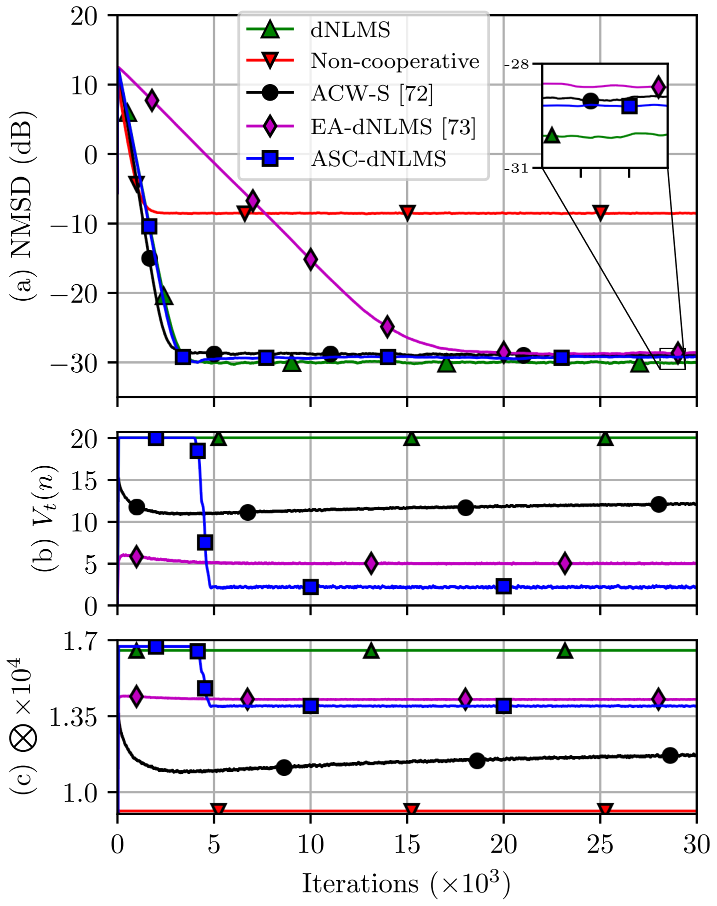


Figure 21: Comparison between the ASC-dNLMS, ACW-S and EA-dNLMS algorithms. The parameters adopted are shown in Table 4. (a) NMSD curves. (b) Number of broadcasts and (c) multiplications per iteration.

Unlike AS-dNLMS, which led to a slight reduction in the steady-state NMSD in comparison to dNLMS with all nodes sampled, ASC-dNLMS achieves a slightly higher level of NMSD in steady state in comparison with the original algorithm. The same occurs for the ACW-S and EA-dNLMS algorithms, as can be seen in Fig. 21(a). We observe that EA-dNLMS presents a notably slower convergence rate in comparison with ACW-S and ASC-dNLMS, which converge at a rate similar to that of dNLMS. On the other hand, from Fig. 21 we see that ACW-S utilizes a comparatively high number of broadcasts, thus saving less energy. We remark that ASC-dNLMS does require each uncensored node k to transmit $e_k^2(n)$ to its neighbors, which means that there is a slight communication overhead in comparison with the other techniques when the nodes are not censored. However, since $e_k^2(n)$ is a scalar, this overhead can be considered negligible, especially when the number of coefficients M of the local estimates $\boldsymbol{\psi}_k$ is high. During steady state, both ACW-S and EA-dNLMS transmit more than the proposed ASC-dNLMS, which maintains all transmissions during the transient but drastically reduces the number of broadcasts after converging. Thus, the proposed technique saves more energy in steady state while preserving the convergence rate.

Lastly, it should be noted from Fig. 21(c) that ASC-dNLMS requires approximately the same number of multiplications per iteration as EA-dNLMS, while leading to 50% less broadcasts. In comparison with ACW-S, ASC-dNLMS requires 21% more multiplications, but leads to 81% less broadcasts. This gap in the number of multiplications is due to the fact that ASC-dNLMS considers the local estimates in the combination step even if they were not updated in the current iteration. In contrast, ACW-S considers that the neighborhood of each node only includes its neighbors that have broadcast their estimate in the current iteration. Despite this, we recall that in censoring applications we are mostly concerned with the number of transmissions, since they are usually the main responsible for energy consumption in the network [72–74]. Moreover, given the percentage differences, ASC-dNLMS can be deemed an efficient solution for censoring.

Table 4: Parameters used in the simulations of Fig. 21

ACW-S [72]	$E_T = 1$, $E_R = 2$
EA-dNLMS [73]	$E_{\text{Act}} = 33.5966 \cdot 10^{-3}$, $E_{\text{Tx}} = 15.16 \cdot 10^{-3}$, $K_{\ell,1} = 2$, $K_{\ell,2} = 0.5$, $K_g = 2$, $\gamma_g = 2$,$\gamma_\ell = 2$, $\delta = 0.5$, $\rho = 0.01$, $r = 2$
ASC-dNLMS	$\beta = 2.1\sigma^2_{\max}$, $\mu_\zeta = 0.0333$

3.1.6.4 Random-Walk Tracking

As can be observed from Fig. 20, increased values of β may hinder the tracking capability of AS-dNLMS. Thus, in this section, we investigate the behavior of the algorithm in nonstationary environments following a random-walk model, in which the optimal solution $\mathbf{w}^{\text{o}}(n)$ varies according to

$$\mathbf{w}^{\text{o}}(n) = \mathbf{w}^{\text{o}}(n-1) + \mathbf{q}(n), \tag{3.39}$$

where $\mathbf{q}(n)$ is a zero-mean random column vector with length M and autocovariance matrix $\mathbf{Q} = \mathrm{E}\{\mathbf{q}(n)\mathbf{q}^{\mathrm{T}}(n)\}$ independent of any other signal [125, 264]. Moreover, we consider that the vector $\mathbf{q}(n)$ is i.i.d. for $n = 1, 2, \cdots$, with a Gaussian distribution such that $\mathbf{Q} = \sigma_q^2\mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix. In Fig. 22, we present the results obtained with the AS-dNLMS algorithm and different values of β as a function of Tr$[\mathbf{Q}]$. For each β_r, we maintained the corresponding step size μ_ζ used in the simulations of Fig. 19. For comparison, we also show the results obtained with the dNLMS algorithm with all nodes sampled. In Fig. 22(a), we present the steady-state levels of NMSD, in Fig. 22(b) the average number of sampled nodes per iteration, and in Fig. 22(c) the steady-state NMSE. The results presented were obtained by averaging the data over the last 600 iterations of each realization, after all the algorithms achieved steady state.

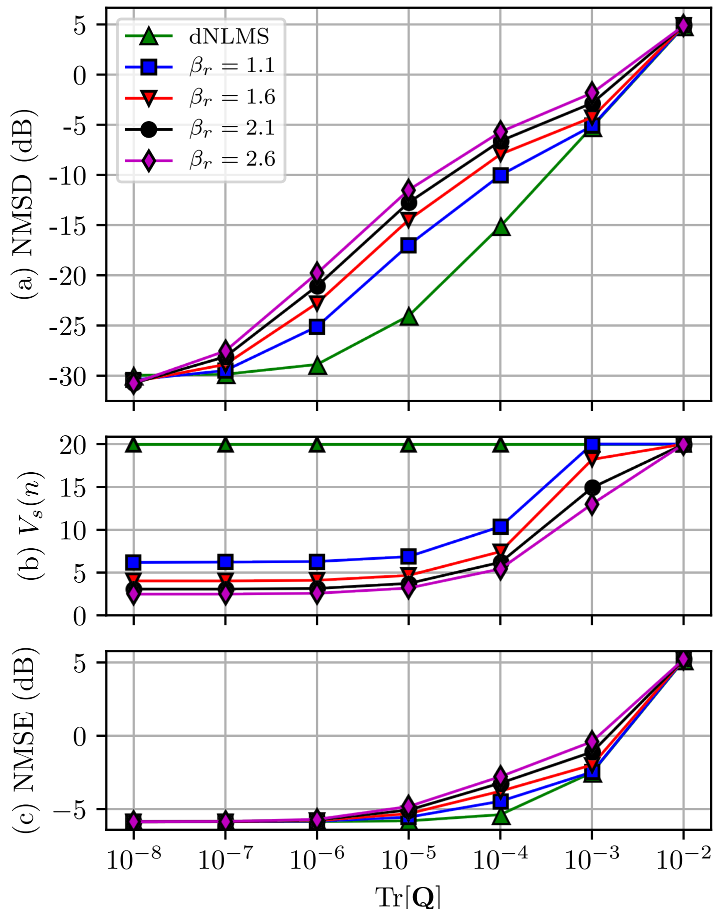


Figure 22: Simulation results in a nonstationary environment following Model (3.39). (a) Steady-state NMSD, (b) Number of nodes sampled per iteration, and (c) Steady-state NMSE.

From Fig. 22(a) we can observe that, in slowly-varying environments (Tr$[\mathbf{Q}] = 10^{-8}$), the performance of AS-dNLMS is similar to that of dNLMS with all nodes sampled. However, for $10^{-7} \leqslant \text{Tr}[\mathbf{Q}] \leqslant 10^{-3}$, there is a degradation in performance in comparison with dNLMS. The higher the parameter β, the more intense this deterioration becomes for a fixed value of Tr$[\mathbf{Q}]$. For Tr$[\mathbf{Q}] \leqslant 10^{-5}$ and a fixed β, this deterioration in comparison with dNLMS intensifies with the increase of $\text{Tr}[\mathbf{Q}]$. On the other hand, for $\text{Tr}[\mathbf{Q}] > 10^{-5}$, the difference in performance begins to decrease as the variations in the optimal system become faster. This can be explained by analyzing Figs. 22(b) and 22(c). We observe that, when the environment varies slowly or moderately, the number of nodes sampled by the AS-dNLMS is not significantly affected by the increase of $\text{Tr}[\mathbf{Q}]$. This occurs since the effects of the changes in the optimal system are small in comparison with those of the measurement noise for $\text{Tr}[\mathbf{Q}] < 10^{-5}$, and thus the NMSE does not increase noticeably, as seen in Fig. 22(c). However, as these variations become faster, they begin to affect the estimation error more intensely, and the NMSE starts to increase for $\text{Tr}[\mathbf{Q}] \geqslant 10^{-5}$, leading to a gradual rise in the number of sampled nodes in Fig. 22(b). For $\text{Tr}[\mathbf{Q}] = 10^{-2}$, the algorithm does not cease to sample any of the nodes for $\beta_r \leqslant 2.6$, and thus its performance matches that of dNLMS.

Next, we repeated the experiment of Fig. 22 for ASC-dNLMS, ACW-S and EA-dNLMS with the parameters of Table 4. The results are shown in Fig. 23. We also present the results obtained with ASC-dNLMS with $\beta_r = 1.3$ and $\beta_r = 0.71$, which were respectively adjusted to lead to the same number of broadcasts as those of EA-dNLMS and ACW-S for $\text{Tr}[\mathbf{Q}] \leqslant 10^{-6}$. Finally, we also show results obtained with the dNLMS algorithm. We observe from Fig. 23(a) that ASC-dNLMS with $\beta_r = 2.1$ achieves a performance similar to that of the other solutions for $\text{Tr}[\mathbf{Q}] = 10^{-8}$ and $\text{Tr}[\mathbf{Q}] = 10^{-7}$. However, it is outperformed for $\text{Tr}[\mathbf{Q}] \geqslant 10^{-6}$. It also employs less transmissions than any other solution in these scenarios. With $\beta_r = 1.3$, ASC-dNLMS outperforms EA-dNLMS for $\text{Tr}[\mathbf{Q}] \leqslant 10^{-7}$ and $\text{Tr}[\mathbf{Q}] = 10^{-4}$, although its NMSD is higher for $\text{Tr}[\mathbf{Q}] = 10^{-6}$ and $\text{Tr}[\mathbf{Q}] = 10^{-5}$. With $\beta_r = 0.71$, ASC-dNLMS outperforms ACW-S for $\text{Tr}[\mathbf{Q}] \leqslant 10^{-7}$, while the opposite occurs for other values of $\text{Tr}[\mathbf{Q}]$. The results suggest that ASC-dNLMS generally outperforms ACW-S and EA-dNLMS in stationary or slowly-varying environments while utilizing the same number of transmissions. Moreover, in these cases it can achieve a comparatively similar performance while transmitting less. However, ASC-dNLMS must be employed with caution in scenarios in which the optimal system changes rapidly. Finally, we can control the trade-off between energy saving and performance by adjusting β.

3.1.6.5 Application in Graph Adaptive Filtering

In this section, we test the proposed sampling algorithm in the same scenario as that of Sec. 2.2.7. We consider the GSP-based version of the dNLMS and AS-dNLMS algorithms, as

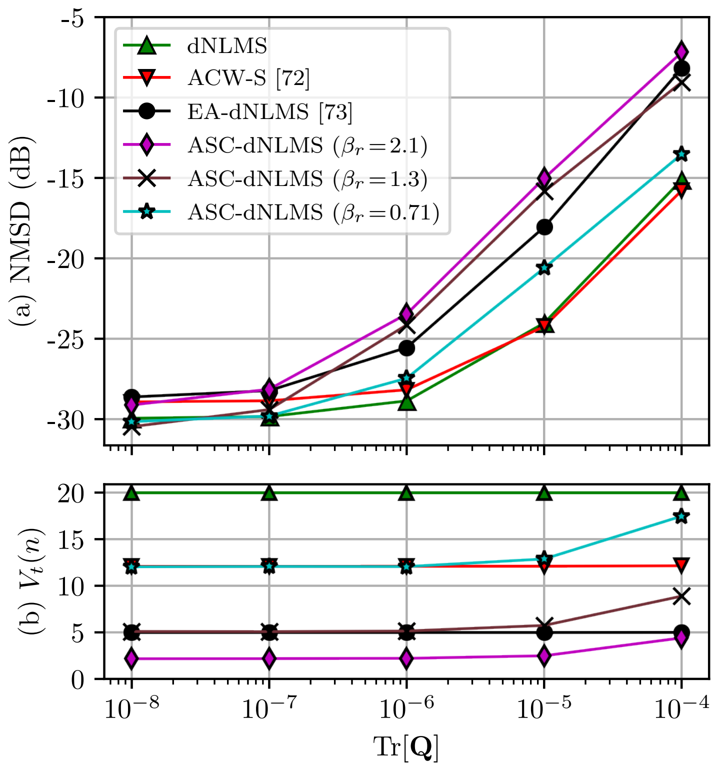


Figure 23: Simulation results in a nonstationary environment following Model (3.39) with the algorithms listed in Table 4. (a) Steady-state NMSD, and (b) Broadcasts per iteration.

described by Eq. (2.50). Moreover, we adopt $M = 5$ and $\widetilde{\mu}_k = 1$ for half of the nodes, while the other half utilizes $\widetilde{\mu}_k = 0.1$. Finally, we use the ACW algorithm with $\nu_{k_{\text{ACW}}} = 0.2$ and $\delta_c = 10^{-5}$.

We divided our dataset into training and testing sets. The former consists of $N_{\text{tr.}} = 3650$ measurements from 12/25/2001 to 12/22/2011, which were periodically replicated to form 20 training epochs. During this period, we consider that $d_k(n) = u_k(n+1)$, where $u_k(n)$ denotes the temperature measurement at node k and time instant n. The vector $\mathbf{x}_k(n)$ is formed as in (2.49). The testing set consists of the measurements from 12/23/2011 to 12/21/2012. In this case, we do not have access to the temperatures registered on the following day. To keep the adaptation going, we use the last estimate of the algorithm as the desired signal. Thus, we set $d_k(n) = \mathbf{x}_k^{\mathrm{T}}(n-1)\mathbf{w}_k(n-1)$. As a performance indicator, we adopt the squared relative reconstruction error (SSRE), given by (2.52) [47]. Similarly to what was done in Sec. 2.2.7, we converted the temperature to degrees Fahrenheit in our experiments to avoid division by zero in (2.52).

It should be noted that, in this scenario, $\sigma^2_{\max}$ is not known *a priori*, making it hard to choose β beforehand. To set β, we ran AS-dNLMS like the original dNLMS in the first epoch, calculated the average NMSE during the last 730 days, and multiplied the result by 3. Then, we used (3.37) with $\Delta_n = 9N_{\text{tr.}}$ to set μ_ζ, and ran the algorithm normally as in Table 5.

In Figs. 24(a) and (b) we present the SRRE obtained in the training and testing periods, respectively. Similarly, Figs. 24(c) and (d) respectively show the number of multiplications per iteration during training and testing. For the ease of visualization, and due to the noisy nature of the data, the curves of Figs. 24(a) and (c) were filtered by a moving-average filter with 1024 coefficients. Nonetheless, in Fig. 24(a) we also show the envelope of the unfiltered curves as dashed lines.

We can observe that AS-dNLMS and the original dNLMS algorithm achieved similar performances during both periods. From the envelope displayed in Fig. 24(a) and the curves of Fig. 24(b), we can see that the SRRE of both algorithms during the test phase is slightly higher than the one observed during training, as expected. Furthermore, from Fig. 24(c) we see that the computational cost of AS-dNLMS remains slightly higher than that of dNLMS during transient, but falls significantly after converging. In the test phase, AS-dNLMS sampled 20 nodes on average per iteration, and performed 45% less multiplications than the original dNLMS while preserving the performance. Considering both the training and test phases, AS-dNLMS reduced the number of multiplications by 17%.

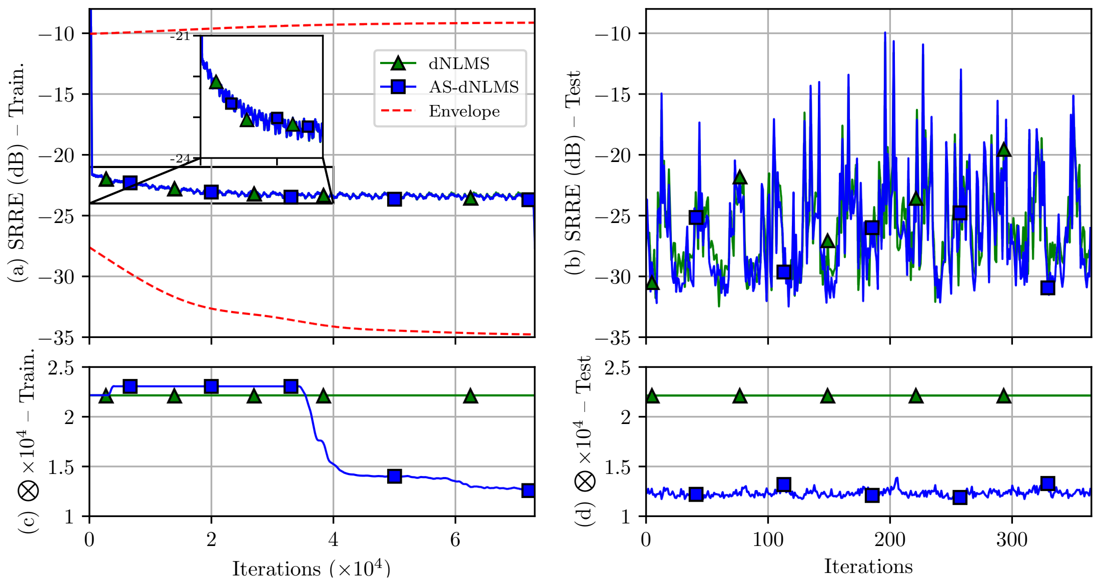


Figure 24: Comparison between dNLMS and AS-dNLMS ($\beta = 80.49$, $\mu_\zeta = 2.5 \cdot 10^{-5}$). (a) and (b): SRRE in the training and testing periods, respectively. (c) and (d): Multiplications per iteration during training and testing, respectively.

Lastly, as an illustrative example, in Fig. 25 we present the estimates provided by AS-dNLMS for the temperature at the stations of Fig. 26 indicated by the arrows, along with the measured data at these locations. We observe that the estimates of the algorithm follow closely the patterns of the real data even with a reduced number of sampled nodes, as desired.

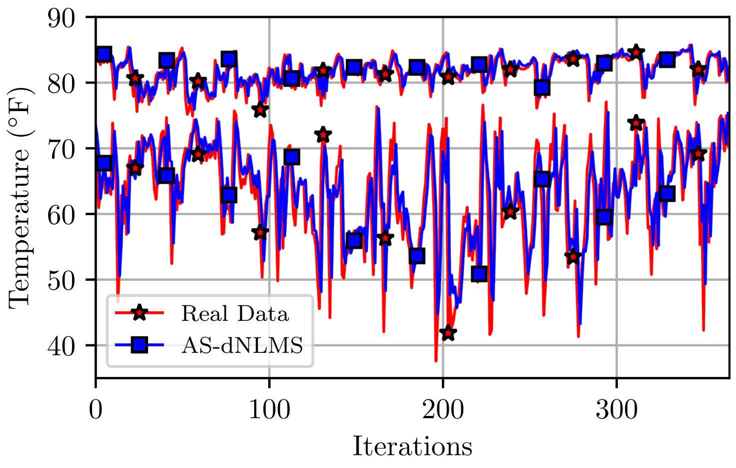


Figure 25: Comparison between the temperature measured at two stations and the estimates provided by AS-dNLMS for them.

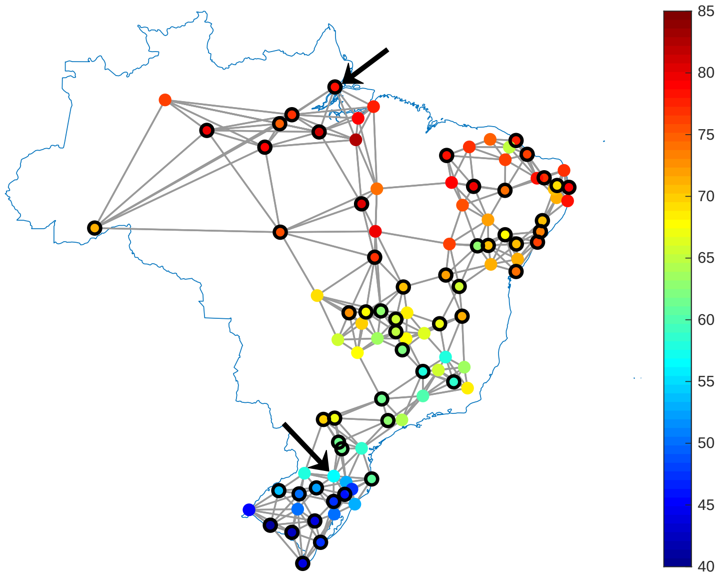


Figure 26: Daily average temperature measured by 100 weather stations on 06/21/2002 (°F). Circled nodes use $\widetilde{\mu}_k = 1$, whereas the others use $\widetilde{\mu}_k = 0.1$. Each edge is a communication link. The arrows point to the stations whose data are used in Fig. 13.

3.2 Modifications for the AS Algorithm

In Sec. 3.1.6, we attested the good behavior of the AS-dNLMS algorithm in a wide range of scenarios, including a simulation with real-world data. However, we also noticed that its tracking capability may be deemed as a potential weakness. This is especially clear in the simulations of Sec. 3.1.6.4, in which the optimal system varies over time according to a random-walk model. Moreover, although the algorithm responds quickly to the abrupt change in the environment in the simulations of Sec. 3.1.6.1, there are scenarios in which an abrupt change can lead to a deterioration in its performance – for instance, when the SNR decreases as a result of the modifications. An example is shown in Fig. 28, considering the network of Fig. 27, which has $V = 25$ nodes, and the Scenario 1 described in Sec. 3.2.6. For comparison, we included the results of the original dNLMS algorithm with all 25 nodes sampled, and with the sampling technique with $V_s = 5$ nodes sampled randomly per iteration. In Fig. 28(a) we present NMSD, and in Figs. 28(b) and (c) the number of nodes sampled and of multiplications per iteration, respectively. For the parameters of the AS-dNLMS algorithm, we selected $\beta = 3.8\sigma^2_{\max}$, which in this scenario yields $\beta = 1.9$, and $\mu_\zeta = 0.0045$. At the middle of each experiment, we abruptly modify the system to be identified in such a way that the Signal-to-Noise Ratio (SNR) drops. We can observe that the AS-dNLMS algorithm behaves well before the abrupt change occurs. Initially, the sampling of the nodes is maintained, and thus the algorithm preserves the convergence rate of the original dNLMS solution. However, after the abrupt change, the convergence rate of AS-dNLMS becomes slower than that of the dNLMS with $V_s = 5$ nodes sampled per iteration. As can be seen from Fig. 28(b), AS-dNLMS does slightly increase the number of nodes sampled per iteration after the change in the environment, but this increase is insufficient to improve the convergence rate.

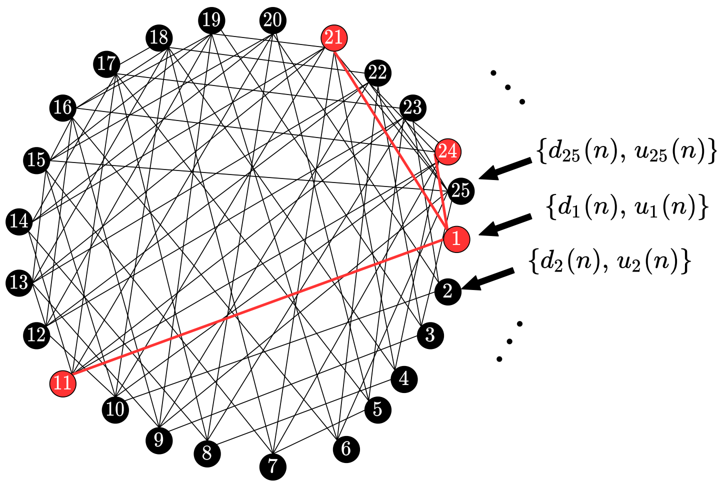


Figure 27: Example of an adaptive diffusion network and its inputs. The neighborhood of node 1 is highlighted in red.

This problem can be aggravated when one of the nodes of the network is much noisier than the others. In this scenario, it may be challenging to select the parameters of the AS-dNLMS algorithm, since from Eq. (3.18) we notice that we must choose $\beta > \sigma^2_{\max}$ in order to ensure that the sampling of the nodes will cease at some point in steady state. However,

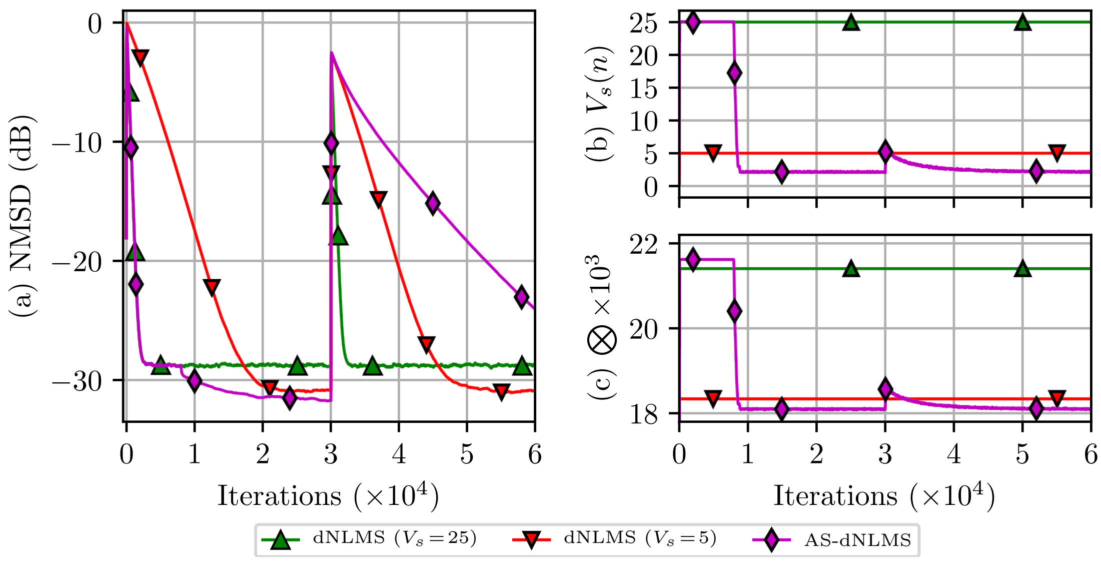


Figure 28: Comparison between dNLMS with V_s nodes randomly sampled per iteration and AS-dNLMS ($\beta = 3.8\sigma^2_{\text{max}} = 1.9$, $\mu_\zeta = 0.0045$). (a) NMSD along the iterations, (b) number of nodes sampled, and (c) multiplications per iteration.

if σ^2_{max} is much greater than the average noise power in the network, the adoption of this rule may lead to excessively large values for β. An example is depicted in Fig. 29, in which we set $\beta = 3.8\sigma^2_{\text{max}}$, as was done in the simulations of Fig. 28. However, in comparison with the scenario considered in Fig. 28, the noise power at the noisiest node was multiplied by ten, which leads to $\beta = 19$. Thus, the noise variance at this node is between 10 and 100 times greater than the noise power in the remainder of the network. Comparing Figs. 28 and 29, we can see that the convergence rate of AS-dNLMS after the abrupt change is even more severely hampered. This is because the largest noise variance in this scenario is much greater than the average noise power. Hence, the selection of the parameter β can become more complicated under these circumstances, requiring a more refined knowledge of the noise power profile throughout the network. Moreover, since prior knowledge of the noise variance is not always available, we can see that it may be interesting to incorporate some sort of mechanism into the AS-dNLMS algorithm for the tuning of the parameter β.

Hence, the goal of this section is to propose modifications for the AS-dNLMS algorithm in order to address its main weaknesses, namely, its tracking capability and the need for prior knowledge on the noise variance for the selection of its parameters. To this end, the following measures are taken:

1. Instead of using global parameters for the entire network, we allow each node to have its own local set of parameters, which enables the algorithm to cope with diversity in the

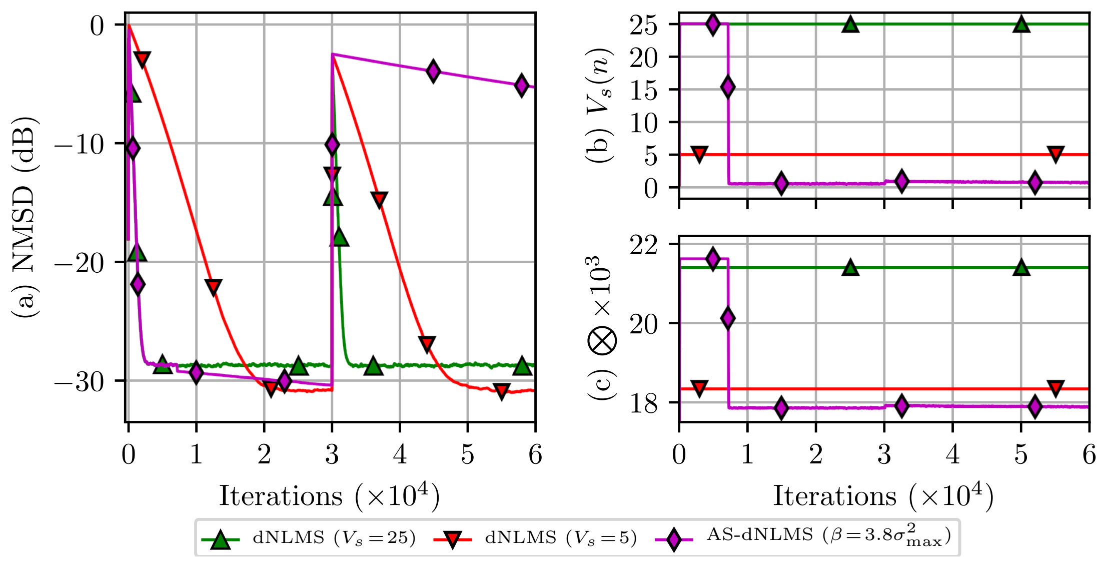


Figure 29: Comparison between dNLMS with V_s nodes randomly sampled per iteration and AS-dNLMS ($\beta = 3.8\sigma^2_{\max} = 19$, $\mu_\zeta = 0.0045$) in Scenario 2 described in Sec. 3.2.6, in which one of the nodes is much noisier than the others. (a) NMSD curves, and (b) number of nodes sampled per iteration.

network, e.g., due to significant variations in the measurement noise power from one node to the other, as in the simulations of Fig. 29.

2. The proposed algorithm estimates the noise variance in each node and modifies their local parameters accordingly in an online and distributed manner, thus eliminating the need for *a priori* knowledge of the noise variance in the network.

3. The proposed algorithm incorporates a change-detection device that allows for the resetting of the sampling mechanism, drastically improving the tracking capability of the algorithm.

3.2.1 Dynamic Tuning of the Parameters

In order to enable the dynamic tuning of the parameters in a local fashion, we now allow each node k to have a local parameter β_k. Thus, replacing β by β_k in (3.13) and maintaining the assumption that $\mathrm{E}\{e_i^2(n)\} \approx \sigma^2_{v_i}$ in steady state, we conclude that $\beta_k > \sum_{i\in\mathcal{N}_k} c_{ik}\sigma^2_{v_i} \triangleq \sigma^2_{\mathcal{N}_k}$ is a necessary and sufficient condition in order to stop the sampling of node k at some point in steady state. Assuming that each node k can calculate an estimate $\widehat{\sigma}^2_{v_k}(n)$ of $\sigma^2_{v_k}$ in an online manner and that they can exchange such estimates with their neighbors, we can write

$$\beta_k(n) = \gamma \sum_{i\in\mathcal{N}_k} c_{ik}\widehat{\sigma}^2_{v_i}(n) \triangleq \gamma\widehat{\sigma}^2_{\mathcal{N}_k}(n), \tag{3.40}$$

where $\gamma > 1$ is a parameter that the designer must choose and whose selection will be discussed in more detail in Sec. 3.2.2. Like the parameter β of the AS-dNLMS, its role is to control how much the sampling of the nodes is penalized in the steady state. Thus, (3.9) can be recast as

$$\alpha_k(n+1) = \alpha_k(n) + \mu_\zeta \phi_k'(n) \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n)\varepsilon_i^2(n) - \gamma \widehat{\sigma}^2_{\mathcal{N}_k}(n)\zeta_k(n) \right]. \tag{3.41}$$

In this work we consider the algorithm proposed in [265] for adaptive noise power estimation. It presents a faster convergence rate in comparison with other methods and is not greatly affected by alterations in the environment. In other words, changes in the optimal system have little impact on the estimate of $\sigma^2_{v_k}$ produced by it, so long as $v_k(n)$ is wide-sense stationary for $k = 1, \cdots, V$ [265]. This is an important trait, because it mitigates the impact of the convergence of the noise power estimation on the sampling mechanism. Since some aspects of this algorithm will serve as the basis for the proposed change-detection mechanism as well as the noise power estimator, we summarize its operation in the following.

The algorithm of [265] uses information from $e_k^2(n)$ at every iteration to estimate the noise variance at node k. Thus, whenever node k is sampled, three low-pass filters with different forgetting factors ν_f, ν_m and ν_s are employed to calculate our estimate:

$$\widehat{\sigma}^2_{f_k}(n) = \nu_f \widehat{\sigma}^2_{f_k}(n-1) + (1-\nu_f)e_k^2(n), \tag{3.42}$$

$$\widehat{\sigma}^2_{m_k}(n) = \nu_m \widehat{\sigma}^2_{m_k}(n-1) + (1-\nu_m)e_k^2(n), \tag{3.43}$$

$$\widehat{\sigma}^2_{s_k}(n) = \nu_s \widehat{\sigma}^2_{s_k}(n-1) + (1-\nu_s)e_k^2(n). \tag{3.44}$$

On the other hand, when node k is not sampled, $\widehat{\sigma}^2_{f_k}$, $\widehat{\sigma}^2_{m_k}$ and $\widehat{\sigma}^2_{s_k}$ are kept fixed. In [265], the following choices are suggested for the forgetting factors: $\nu_f = 1 - \frac{1}{5M}$, $\nu_m = 1 - \frac{1}{15M}$, and $\nu_s = 1 - \frac{1}{45M}$. Since $\nu_f > \nu_m > \nu_s$, $\widehat{\sigma}^2_{f_k}$ converges quickly, which enables it to swiftly respond to changes. However, this estimate is more noticeably affected by fluctuations in $e_k^2(n)$. In contrast, $\widehat{\sigma}^2_{s_k}$ provides a smoother and more accurate estimate in steady state, but takes longer to converge and to detect changes in the environment. In its turn, $\widehat{\sigma}^2_{m_k}$ shows an intermediate behavior [265]. If no change in the environment has been detected recently, i.e., the algorithm is in "normal mode," an intermediate estimate $\widehat{\sigma}^2_{\iota_k}$ is calculated as [265]

$$\widehat{\sigma}^2_{\iota_k}(n) = \nu_f \widehat{\sigma}^2_{\iota_k}(n-1) + (1-\nu_f)\widehat{\sigma}^2_{\min_k}(n), \tag{3.45}$$

where

$$\widehat{\sigma}^2_{\min_k}(n) \triangleq \min\{\widehat{\sigma}^2_{f_k}(n), \widehat{\sigma}^2_{m_k}(n), \widehat{\sigma}^2_{s_k}(n)\}. \tag{3.46}$$

Regardless of the current mode, the consolidated estimate $\widehat{\sigma}^2_{v_k}(n)$ is obtained by [265]

$$\widehat{\sigma}^2_{v_k}(n) = \min\{\widehat{\sigma}^2_{\iota_k}(n), \widehat{\sigma}^2_{f_k}(n)\}. \tag{3.47}$$

The mechanism of [265] enters "change mode", i.e., it considers that a change in the environment has been detected whenever

$$\widehat{\sigma}^2_{f_k}(n) > \widehat{\sigma}^2_{s_k}(n), \tag{3.48}$$

unless the algorithm is still in transient. A flag is used for this purpose, which indicates whether this state has been entered before or not. Then, if this mode is entered for the first time or if $\widehat{\sigma}^2_{f_k}(n) < \widehat{\sigma}^2_{m_k}(n)$, $\widehat{\sigma}^2_{\iota_k}$ is updated as

$$\widehat{\sigma}^2_{\iota_k}(n) = \nu_m \widehat{\sigma}^2_{\iota_k}(n-1) + (1-\nu_m)\widehat{\sigma}^2_{\min_k}(n). \tag{3.49}$$

Otherwise, it is kept fixed until $\widehat{\sigma}^2_{m_k}(n) < \widehat{\sigma}^2_{s_k}(n)$, upon which the algorithm returns to normal mode [265]. A summary of this solution is presented as Algorithm 6 for clarity. The estimate $\widehat{\sigma}^2_{v_k}(n)$ can be calculated locally, since it only uses information available at node k. However, in order to calculate (3.40), each sampled node i must send $\widehat{\sigma}^2_{v_i}(n)$ to its neighbors. This does lead to a small communication overhead in the transmissions by the sampled nodes. Nonetheless, if this information is sent bundled with $\varepsilon_i^2(n)$ and $\boldsymbol{\psi}_i(n)$, no extra broadcasts are required. Furthermore, since $\widehat{\sigma}^2_{v_i}(n)$ is a scalar, the communication overhead associated with its transmission can be considered negligible under most circumstances, especially if the number of coefficients in $\boldsymbol{\psi}_i(n)$ is high.

Since we now have a different $\beta_k(n)$ for each node instead of a global parameter β, we make this replacement in (3.37), allowing the nodes to have distinct step sizes $\mu_{\zeta_k}(n)$. Moreover, since we do not assume the prior knowledge of $\sigma^2_{\max}$ in this approach, we replace it by $\widehat{\sigma}^2_{\mathcal{N}_k}(n)$ in (3.37). Using (3.40), we finally conclude after some algebraic manipulations that we must choose

$$\mu_{\zeta_k}(n) > \frac{1}{\widehat{\sigma}^2_{\mathcal{N}_k}(n)} \left\{ \frac{\alpha^+}{(\gamma-1)(\phi'_0 - \phi'_{\alpha^+})} \left[\left(\frac{\phi'_o}{\phi'_{\alpha^+}} \right)^{\frac{1}{\Delta n}} - 1 \right] \right\} \tag{3.50}$$

Algorithm 6 The noise power estimation algorithm proposed in [265].

```
1:  % Initialization – for each node $k = 1, \cdots, V$, set $\widehat{\sigma}^2_{f_k}(-1) \leftarrow 0$, $\widehat{\sigma}^2_{m_k}(-1) \leftarrow 0$, $\widehat{\sigma}^2_{s_k}(-1) \leftarrow$
    $0$, $\widehat{\sigma}^2_{\iota_k}(-1) \leftarrow 0$, flag$_k$ $\leftarrow$ false, mode$_k$ $\leftarrow$ “normal”
2:  for $n = 1, 2, \cdots$ do
3:      for $k = 1, \cdots, V$ do
4:          $\widehat{\sigma}^2_{f_k}(n) \leftarrow \nu_f \widehat{\sigma}^2_{f_k}(n-1) + (1-\nu_f)e^2_k(n)$
5:
6:          $\widehat{\sigma}^2_{m_k}(n) \leftarrow \nu_m \widehat{\sigma}^2_{m_k}(n-1) + (1-\nu_m)e^2_k(n)$
7:
8:          $\widehat{\sigma}^2_{s_k}(n) \leftarrow \nu_s \widehat{\sigma}^2_{s_k}(n-1) + (1-\nu_s)e^2_k(n)$
9:          $\widehat{\sigma}^2_{\min_k}(n) \leftarrow \min\{\widehat{\sigma}^2_{f_k}(n), \widehat{\sigma}^2_{m_k}(n), \widehat{\sigma}^2_{s_k}(n)\}$
10:         if mode$_k$ = “normal” then
11:             $\widehat{\sigma}^2_{\iota_k}(n) \leftarrow \nu_f \widehat{\sigma}^2_{\iota_k}(n-1) + (1-\nu_f)\widehat{\sigma}^2_{\min_k}(n)$
12:             if $\widehat{\sigma}^2_{f_k} > \widehat{\sigma}^2_{s_k}$ then
13:                 mode$_k$ $\leftarrow$ “change”
14:                 flag$_k$ $\leftarrow$ true
15:             end if
16:         else
17:             if $\widehat{\sigma}^2_{m_k} < \widehat{\sigma}^2_{s_k}$ then
18:                 mode$_k$ $\leftarrow$ “normal”
19:             else
20:                 if $\widehat{\sigma}^2_{f_k} < \widehat{\sigma}^2_{M_k}$ or flag$_k$ = false then
21:                     $\widehat{\sigma}^2_{\iota_k}(n) \leftarrow \nu_m \widehat{\sigma}^2_{\iota_k}(n-1) + (1-\nu_m)\widehat{\sigma}^2_{\min_k}(n)$
22:                 end if
23:             end if
24:         end if
25:         $\widehat{\sigma}^2_{v_k}(n) \leftarrow \min\{\widehat{\sigma}^2_{\iota_k}(n), \widehat{\sigma}^2_{f_k}(n)\}$
26:     end for
27: end for
```

if we wish the sampling of the nodes to cease in at most Δn iterations after the steady state is achieved in terms of the MSE. The term between braces in the right-hand side of (3.50) is a constant once the filter designer chooses the values for Δn and γ. Thus, the tuning of μ_{ζ_k} only requires one extra division per iteration at each node and one extra sum if a regularization term is added to $\widehat{\sigma}^2_{N_k}(n)$ in (3.50).

Incorporating Algorithm 6 as well as (3.40) and (3.50) into AS-dNLMS, we obtain an algorithm where each node k dynamically tunes its own parameters $\beta_k(n)$ and $\mu_{\zeta_k}(n)$. We name the resulting algorithm as Dynamic-Tuning AS-dNLMS (DTAS-dNLMS). For convenience, a pseudocode is as Algorithm 7. It should be mentioned that the same modifications can be straightforwardly applied to ASC-dNLMS. Finally, although we considered static combination weights while deriving DTAS-dNLMS, the resulting algorithm can be used in conjunction with an adaptive rule for the selection of combination weights. In this case, the update of $\{c_{ik}(n)\}$ should also be included in Algorithm 7.

The DTAS-dNLMS algorithm addresses the need for the prior knowledge of $\sigma^2_{\max}$ and increases the flexibility of the sampling mechanism by allowing different values for the parameters in each node. Thus, it may be interesting to compare it to AS-dNLMS. For this reason, in Fig. 30 we resume the simulation of Fig. 29, considering Scenario 2 of Sec. 3.2.6. For AS-dNLMS, we consider two sets of parameters. The curves with diamond-shaped (♦) markers depict the results obtained with $\beta = 3.8\sigma^2_{\max}$ and $\mu_\zeta = 0.0045$, which were obtained following the same rules that were used in the simulations of Figs. 28 and 29. On the other hand, the curves with star-shaped (★) markers show the results obtained with $\beta = 0.7\sigma^2_{\max}$ and $\mu_\zeta = 0.0025$, which were chosen in order to obtain roughly the same number of nodes sampled per iteration as observed in Fig. 28 and a good performance prior to the abrupt change in the optimal system. It should be noted that in this case the choice of the step size μ_ζ is complicated, since the rule for its selection proposed in [81] only applies when $\beta > \sigma^2_{\max}$. Moreover, it is interesting to mention that the AS-dNLMS algorithm is not guaranteed to cease the sampling of the nodes when $\beta < \sigma^2_{\max}$ is chosen [81]. For DTAS-dNLMS, we consider $\gamma = 9$ and $\Delta n = 7 \cdot 10^4$. From Fig. 30(a) we observe that, much like AS-dNLMS, DTAS-dNLMS presents a similar convergence rate to that of dNLMS with every node sampled during the first transient. In steady-state, we see from Fig. 30(b) DTAS-dNLMS samples roughly the same number of nodes as AS-dNLMS with $\beta = 0.7\sigma^2_{\max}$, although its computational cost is slightly higher, as seen from Fig. 30(c). On the other hand, after the abrupt change, DTAS-dNLMS presents a faster convergence rate than AS-dNLMS, albeit slower than dNLMS with $V_s = 5$ nodes sampled per iteration. DTAS-dNLMS outperforms AS-dNLMS when their parameters are adjusted to obtain the same number of nodes sampled per iteration. This can be attributed to the capability of DTAS-dNLMS to adjust the values of the local parameters $\beta_k(n)$ at each node k accordingly, which enables it to maintain and resume the sampling of the noisier nodes faster in comparison with AS-dNLMS. However, the comparison with dNLMS with $V_s = 5$ nodes sampled shows that further modifications are necessary if we desire to improve the tracking capability of the algorithm. For this reason, in Sec. 3.2.3 we incorporate a reset tool for the sampling mechanism in DTAS-dNLMS, which addresses this issue. Before that, however, we present in Sec. 3.2.2 an analysis on the effects of the parameter γ, thus aiding the filter designer in its selection.

Algorithm 7 The ATC DTAS-dNLMS Algorithm.

```
1: % Initialization - for each node $k = 1, \cdots, V$, select a step size $\widetilde{\mu}_k$, a regularization factor
   $\delta_r$, and combination weights $c_{ik}$ satisfying (2.10) for $i = 1, \cdots, V$, and set $\boldsymbol{\psi}_k(0) \leftarrow \mathbf{0}_M$,
   $\mathbf{w}_k(0) \leftarrow \mathbf{0}_M$, $\alpha_k(0) \leftarrow \alpha^+, \zeta_k(0) \leftarrow 1$, and $\varepsilon_k(n) \leftarrow 0$
2: for $n = 1, 2, \cdots$ do
3:     % Adaptation Step
4:     for $k = 1, \cdots, V$ do
5:         if $\alpha_k(n) \geqslant 0$ then
6:             $\zeta_k(n) \leftarrow 1$
7:         else
8:             $\zeta_k(n) \leftarrow 0$
9:         end if
10:        if $\zeta_k(n) = 1$ then
11:            Update $\mathbf{u}_k(n)$
12:            % Calculating the estimation error:
13:            $e_k(n) \leftarrow d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n-1)$
14:            % Updating $\varepsilon_k$:
15:            $\varepsilon_k(n) \leftarrow e_k(n)$
16:            % Calculating the normalized step size $\mu_k(n)$:
17:            $\mu_k(n) \leftarrow \frac{\widetilde{\mu}_k}{\delta_r + \|\mathbf{u}_k(n)\|^2}$
18:            % Adapting the local estimate $\boldsymbol{\psi}_k(n)$:
19:            $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1) + \mu_k(n)e_k(n)\mathbf{u}_k(n)$
20:            Run lines 4-25 of the Algorithm 6, thus obtaining $\widehat{\sigma}^2_{v_k}(n)$
21:        else
22:            $\boldsymbol{\psi}_k(n) \leftarrow \mathbf{w}_k(n-1)$
23:            $\widehat{\sigma}^2_{v_k}(n) \leftarrow \widehat{\sigma}^2_{v_k}(n-1)$
24:        end if
25:    end for
26:    % The nodes transmit $\boldsymbol{\psi}$, $\varepsilon$, and $\widehat{\sigma}_v$ to their neighbors
27:    % Combination Step
28:    for $k = 1, \cdots, V$ do
29:        % Forming the combined estimate $\mathbf{w}_k(n)$ and updating $\alpha_k$:
30:        $\mathbf{w}_k(n) \leftarrow \mathbf{0}_M$
31:        $\alpha_k(n+1) \leftarrow \alpha_k(n)$
32:        $\widehat{\sigma}^2_{\mathcal{N}_k}(n) \leftarrow 0$
33:        for $i \in \mathcal{N}_k$ do
34:            $\widehat{\sigma}^2_{\mathcal{N}_k}(n) \leftarrow \widehat{\sigma}^2_{\mathcal{N}_k}(n) + c_{ik}\widehat{\sigma}^2_{v_i}(n)$
35:            $\beta_k(n) \leftarrow \gamma\widehat{\sigma}^2_{\mathcal{N}_k}(n)$
36:            $\mu_{\zeta_k}(n) \leftarrow \frac{1}{\widehat{\sigma}^2_{\mathcal{N}_k}(n)} \left\{ \frac{\alpha^+}{(\gamma-1)(\phi'_0 - \phi'_{\alpha^+})} \left[ \left( \frac{\phi'_o}{\phi'_{\alpha^+}} \right)^{\frac{1}{\Delta n}} - 1 \right] \right\}$
37:            $\alpha_k(n+1) \leftarrow \alpha_k(n) + \mu_{\zeta_k}(n)\phi'_k(n) \times [\sum_{i \in \mathcal{N}_k} c_{ik}\varepsilon_i^2(n) - \beta_k(n)\zeta_k(n)]$
38:            $\mathbf{w}_k(n) \leftarrow \mathbf{w}_k(n) + c_{ik}\boldsymbol{\psi}_i(n)$
39:        end for
40:    end for
41: end for
```

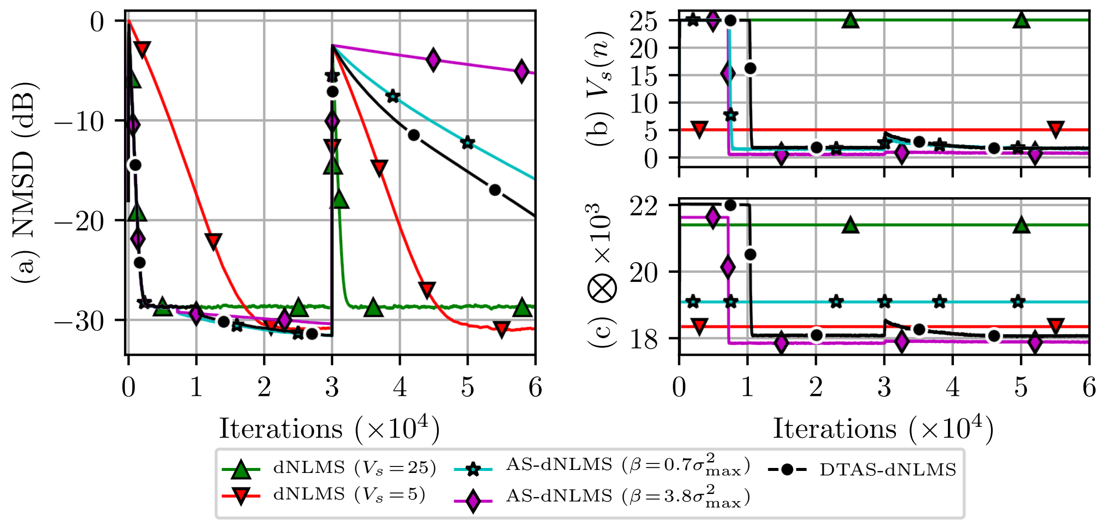


Figure 30: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS and DTAS-dNLMS in Scenario 2 described in Sec. 3.2.6, in which one of the nodes is much noisier than the others. (a) NMSD curves, and (b) number of nodes sampled per iteration.

3.2.2 Selection of the parameter γ

In the simulations of Fig. 30, we adopted $\gamma = 9$ in order to achieve an average of two nodes sampled per iteration at steady-state. However, it is not obvious at first how many nodes will be sampled based on our choice for γ, or, conversely, how we should select this parameter so that we obtain a certain number of nodes sampled per iteration. Intuitively, the influence of γ on the behavior of the algorithm comes from the fact that it controls the penalization of the sampling, similarly to the parameter β in (3.3). Since β was replaced by $\beta_k(n) = \gamma\widehat{\sigma}^2(n)$ in (3.41), it is straightforward to see that γ should affect the number of nodes sampled per iteration. Thus, in this section, we aim to study this influence in detail, which will aid in the selection of γ. We limit our analysis to stationary environments in the absence of impulsive noise for the sake of simplicity, but the results can also be useful in nonstationary environments or in the presence of impulsive noise as well.

For this purpose, we remark that each $\zeta_k(n)$ can be viewed as Bernoulli random variable during steady state that is equal to one with probability p_{ζ_k} or to zero with probability $1-p_{\zeta_k}$ for $k = 1, \cdots, V$, with $0 \leqslant p_{\zeta_k} \leqslant 1$. In this case, the expected number V_s of sampled nodes can be calculated as

$$\mathrm{E}\{V_s\} = \sum_{k=1}^{V} p_{\zeta_k}. \tag{3.51}$$

Thus, we now seek to obtain an estimate $\widehat{p}_{\zeta_k}$ for p_{ζ_k}, $k = 1, \cdots, V$. At this point, it is useful to note that the sampling mechanism should exhibit a cyclic behavior at steady state, as discussed in Sec. 3.1.4. After all, the sampling of the nodes should cease in the steady state, but not permanently, so as to enable to algorithm to track changes in the environment. Thus, when the node is sampled (i.e., $\alpha_k \geqslant 0$) α_k should decrease gradually until it becomes negative, at which point the sampling of node k ceases. On the other hand, when node k is not sampled ($\alpha_k < 0$), α_k should increase gradually until it becomes positive once again, thus resuming the sampling of that node. Taking this into consideration, we could obtain an upper bound for p_{ζ_k} by estimating the maximum expected "duty cycle" of the mechanism, i.e.,

$$p_{\zeta_k} \leqslant \widehat{p}_{\max_k} \triangleq \frac{\breve{\eta}_k}{\breve{\eta}_k + \overline{\eta}_k}, \tag{3.52}$$

where $\breve{\eta}_k$ denotes the maximum expected number of iterations per cycle in which node k is sampled and $\overline{\eta}_k$ is the minimum expected number of iterations in which it is not.

For ease of reading, in this section we omit the intermediate steps that have to be taken in order to estimate $\breve{\eta}_k$ and $\overline{\eta}_k$. However, the detailed derivation is provided in Appendix B. Assuming that $\sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{\varepsilon_i^2(n)\} = \mathrm{E}\{\widehat{\sigma}^2_{\mathcal{N}_k}(n)\} = \sigma^2_{\mathcal{N}_k}$, considering that $\phi'_k(n) \approx \phi'_0$ in steady state, and taking into account that the number of iterations during which the nodes are sampled or not are natural numbers greater than or equal to one, we can estimate $\breve{\eta}_k$ by

$$\breve{\eta}_k = \eta = \left\lceil \frac{1}{\gamma - 1} \right\rceil, \tag{3.53}$$

for $k = 1, \cdots, V$, where $\lceil\cdot\rceil$ denotes the ceiling function. We should notice that (3.53) is inherently greater than one for $\gamma > 1$. Analogously, for $\overline{\eta}_k$, we obtain

$$\overline{\eta}_k = \overline{\eta} = \max\{1, \lfloor \gamma - 1 \rfloor\}, \tag{3.54}$$

where $\lfloor\cdot\rfloor$ denotes the floor function. It is worth noting that $\breve{\eta}_k$ and $\overline{\eta}_k$ only depend on the value of γ, which is assumed to be the same for every node k in the network. Thus, we conclude that $p_{\max_k} = p_{\max}$ for $k = 1, \cdots, V$.

Replacing (3.53) and (3.54) in (3.52) and using (3.51), we finally obtain the following upper bound:

$$\mathrm{E}\{V_s\} \leqslant V \cdot \frac{\left\lceil \dfrac{1}{\gamma - 1} \right\rceil}{\left\lceil \dfrac{1}{\gamma - 1} \right\rceil + \max\{1, \lfloor \gamma - 1 \rfloor\}}. \tag{3.55}$$

Analyzing (3.55), we observe that $\lim_{\gamma\to 1} \mathrm{E}\{V_s\} = V$ and $\lim_{\gamma\to\infty} \mathrm{E}\{V_s\} = 0$, which is in accordance with our expectations. Finally, the upper bound for the expected number of sampled nodes only depends on the total number of nodes V, which is known beforehand, and on the value of γ. This is interesting, because it means that the maximum number of nodes sampled per iteration on average does not depend on the filter length M, the noise variance $\sigma^2_{v_k}$ or the step sizes $\widetilde{\mu}_k$ and μ_{ζ_k} at any node k, and so on. We should notice that (3.55) attests to the simplicity of the selection of γ, since it suffices to choose $\gamma > 1$ in order to ensure a reduction in the number of nodes sampled per iteration in steady state. Furthermore, it enables the filter designer to make a well-informed choice for this parameter, since the maximum number of nodes sampled per iteration on average at steady state is known beforehand. It should be mentioned that there is a compromise between tracking capability and computational cost reduction associated with the choice of $\mathrm{E}\{V_s\}$. For example, if $\mathrm{E}\{V_s\} \ll 1$, changes in the environment will not be detected until a node is sampled, which may only occur many iterations after the change has taken place. Despite this, sensible choices for γ typically lead to satisfactory results after incorporating the change-detection mechanism described in the next section.

3.2.3 Resetting the Sampling of the Nodes

The incorporation of the algorithm of [265] in the sampling mechanism in Sec. 3.2.1 provides a "reset tool" in the sampling mechanism through Criterion (3.48). Since we now have access to an estimate of $\sigma^2_{v_k}$ at every node k, we can now detect changes in the environment if we observe a significant rise in MSE for a long enough period of time. In this case, we could reset α_k to its original value by making $\alpha_k(n+1) = \alpha^+$ instead of running (3.9), in order to ensure the sampling of the nodes while the effects of the change are still observed by the algorithm. However, this criterion can generate many false positives throughout the operation of the algorithm, since it is very sensitive to variations in $e_k^2(n)$. While this does not harm the estimate $\widehat{\sigma}^2_{v_k}(n)$, it can lead to unnecessary resetting of the sampling mechanism.

To circumvent this problem, we introduce a second criterion, and only make $\alpha_k(n+1) = \alpha^+$ if

$$\widehat{\sigma}^2_{f_k}(n) > \chi \widehat{\sigma}^2_{v_k}(n) \tag{3.56}$$

holds for more than M consecutive iterations, where $\chi > 1$ is a sensitivity threshold that the filter designer must choose. After this criterion is met for the first time, $\alpha_k(n+1) = \alpha^+$ is applied until $\widehat{\sigma}^2_{f_k}(n) \leqslant \chi\widehat{\sigma}^2_{v_k}(n)$ is detected, in which case the iteration counter is reset to zero. For convenience, a summary of the proposed reset system for the sampling mechanism is presented as Algorithm 8. The pseudocode presented should be inserted in the combination step of DTAS-dNLMS, between the lines 14 and 15 of Algorithm 7. In order to differentiate between the versions of DTAS-dNLMS with and without the proposed reset system, we henceforth call the former Dynamic-Tuning-and-Resetting AS-dNLMS, or DTRAS-dNLMS for short. We remark that the activation of the change detection mechanism during the convergence of the DTRAS-dNLMS in terms of MSE is not a problem, since the algorithm should maintain the sampling of the nodes during this period.

Algorithm 8 Summary of the sampling reset mechanism of DTRAS-dNLMS.

```
1: % Initialization – for each node $k = 1, \cdots, V$, set counter$_k \leftarrow 0$
2: for $n = 1, 2, \cdots$ do
3:     for $k = 1, \cdots, V$ do
4:         if $\widehat{\sigma}^2_{f_k}(n) > \chi\widehat{\sigma}^2_{v_k}(n)$ then
5:             counter$_k \leftarrow$ counter$_k + 1$
6:         else
7:             counter$_k \leftarrow 0$
8:         end if
9:         if counter$_k > M$ then
10:            $\alpha_k(n+1) \leftarrow \alpha^+$
11:            Go to line 38 of Algorithm 7
12:        else
13:            Go to line 34 of Algorithm 7
14:        end if
15:    end for
16: end for
```

Ideally, χ must be chosen so that the reset mechanism activates when necessary, but registers as few "false positives" as possible. Since these goals are conflicting, there is an underlying compromise in the selection of χ. For this reason, we show next extensive simulation results that aid us in obtaining a practical rule for the choice of this threshold.

Firstly, we remark that (3.56) can be recast as

$$X \triangleq \frac{\widehat{\sigma}^2_{f_k}(n)}{\widehat{\sigma}^2_{v_k}(n)} > \chi, \tag{3.57}$$

where we introduced the auxiliary random variable X for compactness of notation. Thus, if we obtain a reasonable approximation for the probability density function (pdf) f_X of X, we can determine the values of χ for which the probability of Criterion (3.56) being met during the normal operation of the algorithm is sufficiently low.

To do so, we ran computer simulations considering different scenarios. In each case, we collected the values of X for a selected node at every iteration after DTAS-dNLMS achieved steady state, and plotted a histogram of X. We considered 100 realizations with $2 \cdot 10^5$ iterations in each simulation, which was enough for DTAS-dNLMS to converge in terms of NMSD. As a base scenario, we considered Scenario 1 in Sec. 3.2.6, and gradually implemented changes in order to analyze different conditions. The resulting histograms for some of scenarios tested are presented in Fig. 31. In Fig. 31(a), we consider Scenario 1 of Sec. 3.2.6 and show the results obtained for node 1. In Fig. 31(b), we also consider Scenario 1, except that the optimal system $\mathbf{w}^{\rm o}$ is comprised of $M = 10$ coefficients instead of $M = 50$. In Fig. 31(c), we also consider Scenario 1, but the step sizes $\widetilde{\mu}_k$ have been divided by ten in comparison with the original case. Finally, in order to test the validity of the results under different circumstances, in Fig. 31(d) we consider a scenario with a colored signal as input, i.e., $u_k(n) = r_k(n) - 0.8u_k(n-1)$, where $r_k(n)$ follows a Gaussian distribution with zero mean and unit variance for $k = 1, \cdots, V$. We also consider a network with $V = 20$ nodes, different from that of Scenario 1, and different profiles for the step sizes $\widetilde{\mu}_k$ and noise variance $\sigma^2_{v_k}$.

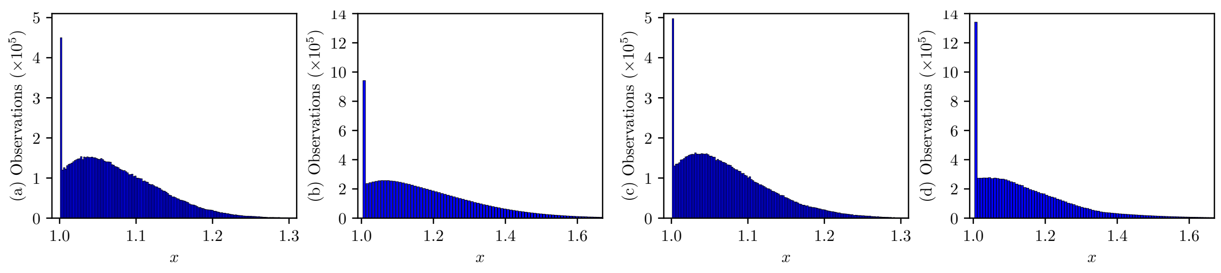


Figure 31: Histograms for X obtained from 100 realizations with $2 \cdot 10^5$ iterations each. Measurements taken in node 1 of the network depicted in Fig. 27. (a) Scenario 1 described in Sec. 3.2.6. (b) $M = 10$, noise variance $\sigma^2_{v_k}$ and step sizes $\widetilde{\mu}_k$ as depicted in Fig. 34. (c) $M = 50$, noise variance $\sigma^2_{v_k}$ and step sizes $\widetilde{\mu}_k$ as depicted in Fig. 34 but divided by 10. (d) $M = 10$, with a colored input signal and a different network, noise power, and step size profiles in comparison with Scenario 1.

Comparing Figs. 31(a), (b), (c), and (d) we observe that, although the exact distribution of X changes from one scenario to the other, its general shape does not vary significantly. Moreover, there are no observations for $x < 1$, and the histograms present a peak at $x = 1$. These observations stem from (3.47), which imposes that $\widehat{\sigma}^2_{f_k}(n) \geqslant \widehat{\sigma}^2_{v_k}(n)$, i.e., $X \geqslant 1$, and enables $\widehat{\sigma}^2_{f_k}(n) = \widehat{\sigma}^2_{v_k}(n)$ whenever $\widehat{\sigma}^2_{f_k}(n) \leqslant \widehat{\sigma}^2_{\iota_k}(n)$.

Approximating the curve for $x > 1$ by a scaled and truncated normal distribution with mean

a_1 and standard deviation a_2, we then estimate the pdf $f_X(x)$ as

$$f_X(x) \approx a_3\delta(x-1) + \frac{a_4}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(x-a_1)^2}{2a_2^2}\right] H(x-1), \tag{3.58}$$

where δ and H respectively denote the Dirac delta and Heaviside step functions, $a_3 = \Pr[X = 1]$ and a_4 is a scaling factor that is a function of a_1, a_2 and a_3.

We are interested in obtaining χ such that $\Pr[X > \chi] \leqslant p_\chi$, with $0 < p_\chi \ll 1$. Thus, we must have

$$1 - F_X(\chi) \leqslant p_\chi, \tag{3.59}$$

where $F_X(\chi) = \int_{-\infty}^{\chi} f_X(x)dx$ is the cumulative density function (cdf) of X.

For the sake of brevity, in this section we omit the step-by-step resolution of (3.59) and skip to the final solution of this inequality. However, a thorough demonstration for this result is provided in Appendix C. It can be shown that (3.59) is satisfied if we choose

$$\chi \geqslant a_1 + a_2\sqrt{2} \cdot \text{erf}^{-1}\left[\frac{p_\chi}{1-a_3} \cdot \text{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) + \frac{1-p_\chi-a_3}{1-a_3}\right], \tag{3.60}$$

where $\text{erf}(\cdot)$ and $\text{erf}^{-1}(\cdot)$ denote respectively the error function and the inverse error function.

Let us examine (3.60) for two special cases: $p_\chi = 0$ and $p_\chi = 1$. The former case corresponds to $\Pr[X > \chi] \leqslant 0$, i.e., we should choose a value for χ that X can never surpass. Making the replacement $p_\chi = 0$ in (3.60), we obtain $\chi \to \infty$, which is in accordance with our expectations. On the other hand, the case $p_\chi = 1$ corresponds to a situation where $\Pr[X > \chi] \leqslant 1$, which should hold for any finite value of χ, since the support of X ranges from one to infinity. Replacing $p_\chi = 1$ in (3.60), we obtain

$$\chi \geqslant a_1 + a_2\sqrt{2} \cdot \text{erf}^{-1}\left\{\frac{1}{1-a_3}\left[\text{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) - a_3\right]\right\}. \tag{3.61}$$

For the sake of simplicity, let us initially consider the special case $a_3 = 0$ in (3.61). This corresponds to the case where there is no Dirac delta in the expression for $f_X(x)$ in (3.58). In this case, (3.61) yields $\chi \geqslant 1$. This is reasonable, since $X \geqslant 1$ always holds. Moreover, for $0 < a_3 \leqslant 1$, (3.61) yields even lower values for χ, which further supports the validity of the obtained expression. Finally, simulation results suggest that, in stationary environments and in the absence of impulsive noise, $p_\chi = 5 \cdot 10^{-4}$ leads to good results.

In order to successfully apply (3.60), we must estimate the values of a_1, a_2 and a_3. Simulation results show that the values of these parameters do not vary significantly (e.g., more than 10%) with the step size $\widetilde{\mu}_k$, the network topology or the noise power profile. However, they do depend on the filter length M. In Fig. 32 we present estimates obtained for a_1, a_2 and a_3 considering different values for $10 \leqslant M \leqslant 100$ in Scenario 1. They were derived by fitting the histogram obtained for X to Model (3.58) for each value of M using the Nonlinear Least Squares method.

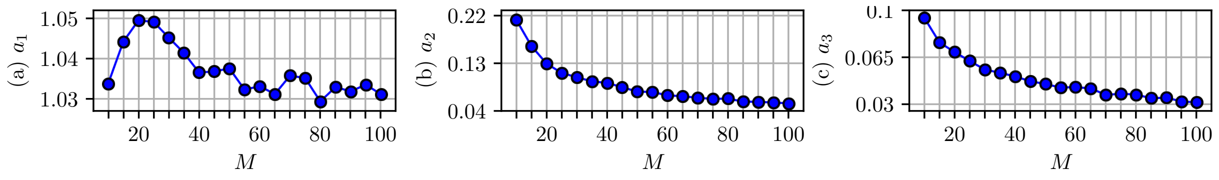


Figure 32: Values fit from the experimental data for (a) a_1, (b) a_2, and (c) a_3 for each filter length $10 \leqslant M \leqslant 100$ considering Model (3.58) and the Nonlinear Least Squares method.

Using the values depicted in Fig. 32 and considering (3.60) with an equality sign and $p_\chi = 5 \cdot 10^{-4}$, it is possible to plot χ as a function of M, as depicted in Fig. 33. Furthermore, for the sake of simplicity, one could seek to approximate $\chi(M)$ from the experimental data as an exponential function. Using once again the Nonlinear Least Squares method, the resulting approximation is given by

$$\chi(M) \approx 1.2326 + 0.8603 \exp(-0.0547 \cdot M), \tag{3.62}$$

which is also depicted in Fig. 33. As can be seen from the plot, (3.62) provides a reasonable approximation for $\chi(M)$, greatly facilitating the selection of this parameter after the filter length is set. Although we only present the results for $10 \leqslant M \leqslant 100$, (3.62) holds as an approximation for M outside of this range as well.

A few remarks should be made about the scenario considered in Fig. 31(d), with colored noise as input. If we fit the values of a_1, a_2 and a_3 using the Nonlinear Least Squares method to the experimental data, we get values quite different from those of Fig. 32, which were obtained considering Scenario 1 of Sec. 3.2.6 with different values for M and white noise as input. However, replacing these values for a_1, a_2 and a_3 in (3.59), we get $\chi = 1.622$. This represents an 8.4% error in comparison with the results depicted in Fig. 33, and a 6.7% error in comparison with the value yielded by (3.62). Thus, despite all the differences between the scenario of Fig. 31(d) and those of Figs. 31(a), (b) and (c), the final value obtained for χ by the method

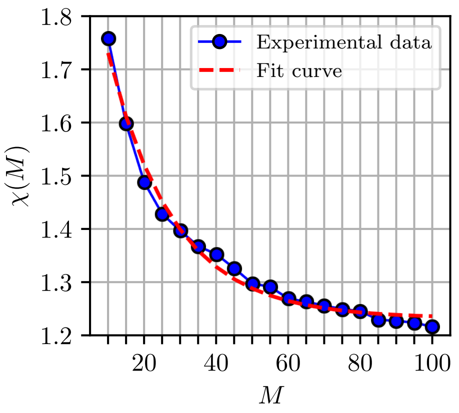


Figure 33: Comparison between the values obtained for χ using (3.60) with a_1, a_2 and a_3 as depicted in Fig 32 and those yielded by the Approximation (3.62).

described in this section is only slightly affected, which shows that it is robust to certain changes in the scenario. Overall, the vast number of scenarios considered and experiments conducted render this model reliable as well as wide-ranging.

In a nutshell, the DTRAS-dNLMS is given by the junction of Algorithms 6, 7, and 8. It enables each node to have its own local parameters β_k and μ_{ζ_k}, which are tuned at each iteration according to (3.40) and (3.50). This is made possible by the adaptive estimation of the noise power at each node and the communication between neighbors. By combining these algorithms, we are able to address all of the main limitations of AS-dNLMS.

3.2.4 Computational Complexity

In this section, we analyze the computational cost of DTRAS-dNLMS and compare it to those of dNLMS and AS-dNLMS. As can be seen from Algorithms 6 and 8, the number of operations required by DTRAS-dNLMS can vary from one iteration to another, since some operations are only carried out if certain conditions are met. Thus, we consider the worst-case scenario in our analysis and assume that the algorithms are used in conjunction with ACW.

We begin by comparing the costs of DTRAS-dNLMS and AS-dNLMS [81]. For this purpose, we examine the increase in cost that the modifications proposed in Secs. 3.2.1–3.2.3 produce. From Algorithms 6 and 7, we can see that the noise estimation algorithm of [265] is only run when node k is sampled, and its cost can be represented by $8\zeta_k(n)$ multiplications, $4\zeta_k(n)$ sums, and $3\zeta_k(n)$ comparisons per node at iteration n. Moreover, from lines 34 and 35 of Algorithm 7, we observe that the computation of $\beta_k(n)$ requires $\zeta_k(n) \cdot (|\mathcal{N}_k| + 1)$ multiplications and $\zeta_k(n) \cdot (|\mathcal{N}_k| - 1)$ sums per node per iteration, since we do not need to run these lines if node k is not sampled. From line 17 of the same table, we conclude that the computation of $\mu_{\zeta_k}(n)$ demands one extra sum and one extra division. Finally, line 5 of Algorithm 8 adds one multiplication and one comparison to the total computational cost of the algorithm. Line 5 from the same table contributes with yet another comparison, and if the condition of line 1 holds, there is one extra sum. Hence, in comparison with AS-dNLMS, DTRAS-dNLMS requires $\zeta_k(n) \cdot (|\mathcal{N}_k| + 9) + 1$ more multiplications, $\zeta_k(n) \cdot (|\mathcal{N}_k| + 3) + 2$ more sums, one extra division, and $2 + 3\zeta_k(n)$ more comparisons at each node k and time instant n in the worst-case scenario. These results are summarized in Tab. 5, in which we show the estimated number of operations required by each algorithm. For reference, we also show the computational cost of dNLMS with all nodes sampled. For both AS-dNLMS and DTRAS-dNLMS, we consider an implementation of $\phi'[\alpha_k(n)]$ through a look-up table, which is not taken into account in Tab. 5.

Table 5: Computational cost comparison between dNLMS, AS-dNLMS and DTRAS-dNLMS with ACW: number of operations per iteration for each node k.

Cost	dNLMS	AS-dNLMS	DTRAS-dNLMS (worst-case scenario)
Mult.	$M(3+2\|\mathcal{N}_k\|)+\|\mathcal{N}_k\|+1$	$\zeta_k(n)(3M+4)+2M\|\mathcal{N}_k\|+3\|\mathcal{N}_k\|+1$	$\zeta_k(n)(3M+13+\|\mathcal{N}_k\|)+2M\|\mathcal{N}_k\|+3\|\mathcal{N}_k\|$
Sums	$M(2+3\|\mathcal{N}_k\|)+2\|\mathcal{N}_k\|-1$	$\zeta_k(n)(5M-1)+(\|\mathcal{N}_k\|-1)(3M-1)$	$\zeta_k(n)(5M+2+\|\mathcal{N}_k\|)+(\|\mathcal{N}_k\|-1)(3M-1)+2$
Div.	$2\|\mathcal{N}_k\|+1$	$2\|\mathcal{N}_k\|+\zeta_k(n)$	$2\|\mathcal{N}_k\|+\zeta_k(n)+1$
Compar.	0	3	$5+3\zeta_k(n)$

From the analysis presented, we can see that the computational cost of DTRAS-dNLMS is always higher than that of AS-dNLMS if both algorithms are adjusted to sample the nodes at the same rate. In other words, the improvement in the tracking capability of the algorithm from the reset mechanism and the elimination of the need for *a priori* knowledge of the noise power come at the inevitable expense of an increase in computational complexity. However, this rise in the computational cost is mostly concentrated on the occasions in which node k is sampled, i.e., $\zeta_k(n) = 1$, especially if node k has many neighbors, in which case $|\mathcal{N}_k|$ is large. In contrast, if node k is not sampled, i.e. $\zeta_k(n) = 0$, the difference in complexity between both algorithms comes down to one extra multiplication, two sums, one division, and two comparisons at that node. Hence, the increase in computational cost generated by the proposed mechanisms tends to be much more noticeable in the transient than in steady state. Finally, the cost associated with the algorithm can be lower than depicted in Tab. 5 at several iterations. For example, if the reset mechanism is activated, i.e., the condition of line 9 of Algorithm 8 holds, $\alpha_k(n+1)$ is set to α^+ and lines 34–37 of Algorithm 7 do not have to be run, which saves some computation.

In comparison with dNLMS, we observe from Tab. 5 that DTRAS-dNLMS saves $3M-2|\mathcal{N}_k|+1$ multiplications and sums when node k is not sampled. Thus, the reduction in computational cost provided by DTRAS-dNLMS becomes more noticeable as M increases. On the other hand, if most nodes have large neighborhoods, the computational savings tend to be lower. However, the filter length M is usually larger than the average neighborhood size, especially for sparse and cluster topologies [42–44, 266, 267].

The expressions depicted in Tab. 5 refer to the instantaneous computational cost at each node k and time instant n. To analyze the expected computational cost of DTRAS-dNLMS in steady state, we should replace $\zeta_k(n)$ with its expected value $\mathrm{E}\{\zeta_k\} = p_{\zeta_k}$, for which we estimated an upper bound in Sec. 3.2.2. Denoting the savings difference in the number of multiplications required by dNLMS and DTRAS-dNLMS at each node k by $\Delta\otimes_k$, we conclude from Tab. 5 that, in steady state,

$$\mathrm{E}\{\Delta\otimes_k\} = 3M-2|\mathcal{N}_k|+1 - p_{\zeta_k}(3M+13+|\mathcal{N}_k|). \tag{3.63}$$

If $\mathrm{E}\{\Delta\otimes_k\} > 0$, DTRAS-dNLMS saves computation at node k in comparison with dNLMS. On the other hand, if $\mathrm{E}\{\Delta\otimes_k\} < 0$, DTRAS-dNLMS is the costlier algorithm. We can see from (3.63) that lower sampling probabilities p_{ζ_k} lead to greater $\mathrm{E}\{\Delta\otimes_k\}$. In other words, the less nodes are sampled on average, the greater the expected savings in computational resources, as we expected. On the other hand, if the sampling probabilities p_{ζ_k} are too high, $\mathrm{E}\{\Delta\otimes_k\}$ can become negative. Summing $\mathrm{E}\{\Delta\otimes_k\}$ for $k = 1, \cdots, V$, we obtain the difference in cost for the whole network, given by

$$\mathrm{E}\{\Delta\otimes_{\text{global}}\} = V(3M+1) - \sum_{k=1}^{V} \left[2|\mathcal{N}_k| + p_{\zeta_k}(3M+13+|\mathcal{N}_k|)\right]. \tag{3.64}$$

Since $p_{\zeta_k} \leqslant p_{\max_k} = p_{\max}$ for $k = 1, \cdots, V$, we conclude from (3.64) that, in order to ensure a reduction in the number of multiplications in steady state in comparison with dNLMS, i.e.,

$\mathrm{E}\{\Delta\otimes_{\text{global}}\} > 0$, we must have

$$p_{\max} = \frac{\left\lceil \frac{1}{\gamma - 1} \right\rceil}{\left\lceil \frac{1}{\gamma - 1} \right\rceil + \max\{1, \lfloor \gamma - 1 \rfloor\}} < \frac{V(3M+1) - 2\sum_{k=1}^{V} |\mathcal{N}_k|}{V(3M+13) + \sum_{k=1}^{V} |\mathcal{N}_k|}. \tag{3.65}$$

Hence, assuming that the network topology is known beforehand, which is a usual practice [1,4–6,142,186,268], we can use (3.65) to determine the minimum value of γ required to ensure that DTRAS-dNLMS has a lower computational cost than dNLMS in steady state for a certain filter length M. It is worth noting that if

$$M < \frac{1}{3}\left(2 \cdot \frac{1}{V}\sum_{k=1}^{V} |\mathcal{N}_k| - 1\right), \tag{3.66}$$

DTRAS-dNLMS cannot save computation in comparison with dNLMS in the worst-case scenario. If (3.66) holds, we would need $p_{\max} < 0$ to ensure a reduction in the computational cost in (3.65), which is impossible. Nonetheless, we should notice that the condition imposed by (3.66) is not very restrictive, since M is usually larger than the average neighborhood size.

Finally, we remark that an analogous procedure could be done to ensure a reduction in the number of sums, but we focused on the multiplications since they are usually more demanding from a computational perspective.

3.2.5 Overview of the Parameters of DTRAS-dNLMS

In this section, we provide a brief summary of the roles of the parameters of DTRAS-dNLMS and how to select them. Besides the forgetting factors ν_f, ν_m and ν_f of the algorithm of [265], which are respectively given by $\nu_f = 1 - \frac{1}{5M}$, $\nu_m = 1 - \frac{1}{15M}$ and $\nu_m = 1 - \frac{1}{45M}$, DTRAS-dNLMS has three other parameters: γ, Δn and χ. The role of γ is analogous to that of β in the AS-dNLMS algorithm. Both are used to control the number of nodes sampled in steady state. Furthermore, the parameter Δn was already present in AS-dNLMS. In both algorithms, its role is the same: adjusting the speed of the update of $\alpha_k(n)$, and thus controlling how fast the nodes cease to be sampled. The difference between both solutions resides in the fact that, in the AS-dNLMS algorithm, Δn is used to set the step size μ_ζ *a priori* according to (3.37), whereas in DTRAS-dNLMS it is used to adjust each local step size $\mu_{\zeta_k}(n)$ in an online manner, as can be seen in (3.50). Thus, the only additional parameter that DTRAS-dNLMS has in comparison with AS-dNLMS is χ, which is responsible for tuning the sensitivity of the reset mechanism proposed in Sec. 3.2.3. This comparison is summarized in Tab. 6.

Table 6: Comparison between the parameters of DTRAS-dNLMS and AS-dNLMS.

Role	AS-dNLMS	DTRAS-dNLMS
Controlling the number of nodes sampled in steady state	β	γ
Controlling the speed of the update of $\alpha_k(n)$	Δn	
Controlling the probability of activation of the reset mechanism	–	χ

The only condition on the parameter γ to ensure that the nodes cease to be sampled is $\gamma > 1$. Furthermore, (3.55) allows the filter designer to know how many nodes would be sampled per iteration in the worst-case scenario. If γ is close to one, the number of nodes sampled per iteration may be high, which undermines the benefits of the sampling mechanism. This can be attested from (3.65), which can be used to ensure a reduction in the computational cost of the algorithm. Due to (3.50) and to the reset mechanism, the influence of the parameter γ on the tracking capability is reduced. This is in stark contrast with the parameter β of AS-dNLMS. It was shown in [81] that the higher the β, the more noticeable the deterioration in the tracking capability of the algorithm, even for moderate $\beta/\sigma^2_{\max}$ ratios such as $\beta \leqslant 5\sigma^2_{\max}$. However, as mentioned in Sec. 3.2.2, selecting γ such that $\mathrm{E}\{V_s\} \ll 1$ can be problematic. In this case, changes in the environment may go unnoticed for long periods of time, since the network may not sample any node for a high number of iterations.

As for the parameter Δn, adopting a very low value for it can lead to high $\mu_{\zeta_k}(n)$, which may lead to the lack of sampling during transient. Simulations suggest that the convergence speed of the algorithm in terms of NMSD should be generally taken into account when selecting Δn. Nonetheless, it is important to mention that DTRAS-dNLMS is not very sensitive to moderate variations in its value, so this selection does not have to be very precise. Simulation results indicate that choosing

$$\Delta n \approx 2M^2 \cdot \frac{V}{\sum_{k=1}^{V} \widetilde{\mu}_k} \tag{3.67}$$

leads to good performances by the DTRAS-dNLMS algorithm if the average step size $\frac{1}{V}\sum_{k=1}^{V} \widetilde{\mu}_k$ is less than one. This is a heuristic result that can be interpreted as follows. The values for Δn should be greater when the convergence speed in terms of NMSD is slow, which occurs when M is high or the average step size is low.

Lastly, in stationary environments and in the absence of impulsive noise, χ can be set according to (3.62). The model proposed in Sec. 3.2.3 for the adjustment of this parameter produced satisfactory results in different environments. Therefore, DTRAS-dNLMS fully addresses the main weaknesses of AS-dNLMS and the adjustment of its parameters is simple given the analysis previously presented.

3.2.6 Simulation Results

In this section, we present simulation results to showcase the behavior of the DTAS-dNLMS and DTRAS-dNLMS algorithms. The results presented were obtained over an average of 100 independent realizations. For better visualization, we filtered the curves by a moving-average filter with 64 coefficients. In every case, we consider that the order of the filter is equal to that of the optimal system. The combination weights are updated using the ACW algorithm with $\nu_{k_{\text{ACW}}} = 0.2$ for $k = 1, \cdots, V$ [215]. Moreover, we use $\delta_r = 10^{-5}$ as a regularization factor in (2.53) and add $\delta_c = 10^{-8}$ to $\widehat{\sigma}^2_{ik}(n)$ before calculating its reciprocal in (2.23). Although the results obtained with DTAS-dNLMS in the simulations of Fig. 30 are poor, we still include it in the simulations of this section for the sake of comparison.

3.2.6.1 Scenario 1 – Base Scenario

We consider the network of Fig. 27 in the simulations. Furthermore, each node k is subject to a different noise variance $\sigma^2_{v_k}$, as shown in Fig. 34(a), and we consider $\widetilde{\mu}_k \in \{0.1, 1\}$ for each node k, as depicted in Fig. 34(b). For the optimal system $\mathbf{w}^{\text{o}}$, we consider a vector with $M = 50$ coefficients randomly generated following a Uniform distribution in the range $[-1, 1]$. The vector thus obtained was then normalized, so that the resulting $\mathbf{w}^{\text{o}}$ presents unit norm. To simulate an abrupt change in the environment, in the middle of each realization we multiply the vector $\mathbf{w}^{\text{o}}$ by 0.25. Finally, the input signal $u_k(n)$ is white Gaussian with unit variance for $k = 1, \cdots, V$.

Firstly, we resume the simulation of Fig. 28 in order to compare DTRAS-dNLMS with the algorithms previously considered. Its parameter χ was set to 1.298 using the results of Sec. 3.2.3 with $p_\chi = 5 \cdot 10^{-4}$. In order to obtain approximately the same number of nodes sampled per iteration in steady state in comparison with AS-dNLMS and DTAS-dNLMS, $\gamma = 11$ was adopted. Comparing to DTAS-dNLMS in which we adopted $\gamma = 9$, a higher value of γ is

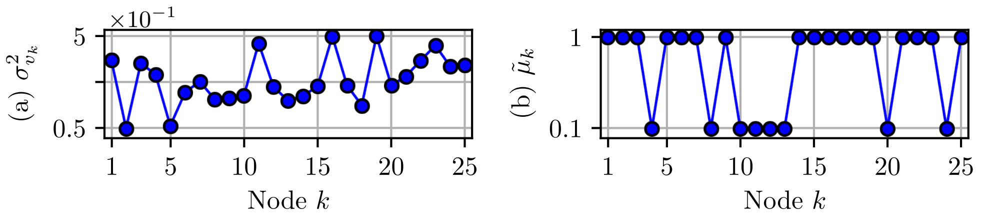


Figure 34: (a) Noise variance $\sigma^2_{v_k}$, and (b) normalized step size $\widetilde{\mu}_k$ for $k = 1, \cdots, V$ considered in the simulations.

required for DTRAS-dNLMS. This difference is due to spurious resets of the sampling system, which still occasionally occur in DTRAS-dNLMS even with $p_\chi = 5 \cdot 10^{-4}$. This will be illustrated in more detail in the sequel. Lastly, the value of μ_ζ for AS-dNLMS was chosen using (3.37) with an equality sign, whereas DTAS-dNLMS and DTRAS-dNLMS tune $\mu_{\zeta_k}(n)$ at each iteration using (3.50). In all cases, $\Delta n = 7000$ was adopted.

In Fig. 35(a) we present the NMSD curves, in Fig. 35(b) the number of nodes sampled, and in Fig. 35(c) the number of multiplications per iteration, respectively. As seen in Fig. 28, the more nodes are sampled during the transient, the faster the convergence rate. We also observe that, unlike AS-dNLMS and DTAS-dNLMS, DTRAS-dNLMS resumes the sampling of practically every node after the abrupt change in the environment. For this reason, its convergence rate is similar to that of dNLMS with $V_s = 25$ during both transients. Moreover, as observed in the simulations of Figs. 17, 28, and 29, the sampling of less nodes leads to a slight reduction in steady-state NMSD. The dNLMS algorithm with $V_s = 5$ sampled nodes per iteration presents a steady-state NMSD approximately 2 dB lower than the one achieved by the version with all nodes sampled, whereas DTRAS-dNLMS reaches a steady-state NMSD that is 1.3 dB lower than that of the algorithm with five sampled nodes, as well as a faster convergence rate. From Figs. 35(b) and 35(c) we observe that during the transients the computational cost of DTRAS-dNLMS is slightly higher than those of the dNLMS algorithm with all nodes sampled and AS-dNLMS. Furthermore, during the first iterations, its cost is slightly lower than that of DTAS-dNLMS, since in this period the activation of the reset mechanism prevents the update of α_k, which saves some computation. After a while, the cost of DTRAS-dNLMS slightly increases as the reset mechanism ceases to act, becoming close to that of DTAS-dNLMS. However, once steady state is achieved, the computational cost of DTRAS-dNLMS decreases drastically, becoming much lower than that of dNLMS and only marginally higher than that of AS-dNLMS. This is in line with the discussion presented in Sec. 3.2.4. In comparison with the worst-case scenario depicted in Tab. 5, DTRAS-dNLMS actually performed, on average, 18.6 less multiplications per iteration throughout the entire network. This discrepancy is mostly concentrated in the transients, rather than the steady state. For instance, between iterations $50 \cdot 10^3$ and $60 \cdot 10^3$, the average difference comes down to 1.4 multiplication per iteration.

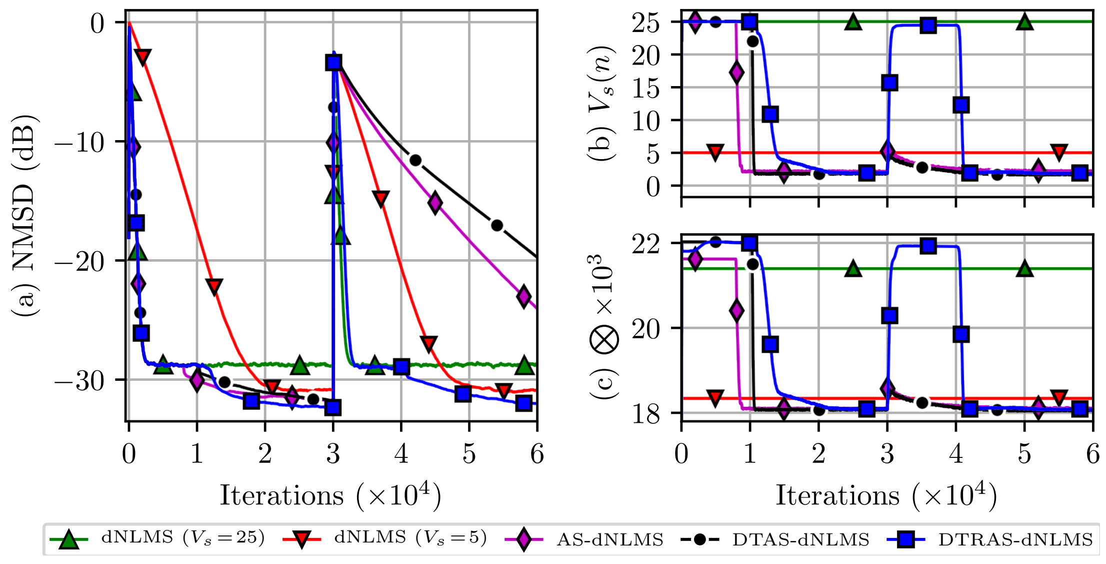


Figure 35: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 1.9$, $\mu_\zeta = 0.0045$), DTAS-dNLMS ($\gamma = 9$), and DTRAS-dNLMS ($\gamma = 11$, $\chi = 1.298$). For DTAS-dNLMS and DTRAS-dNLMS, $\mu_{\zeta_k}(n)$ was set using (3.50) with $\Delta n = 7000$. (a) NMSD curves, (b) number of nodes sampled and (c) multiplications per iteration.

In order to verify (3.55), we also tested the DTAS-dNLMS and DTRAS-dNLMS algorithms in a stationary environment with different values of $\gamma \geqslant 1$. For $\gamma = 1$, a fixed step size μ_{ζ_k} was adopted for $k = 1, \cdots, V$ in order to avoid division by zero in (3.50). Furthermore, $\chi = 1.298$ was adopted for DTRAS-dNLMS. In each experiment, we considered $4 \cdot 10^4$ iterations and calculated the average number of nodes sampled during the last $4 \cdot 10^3$, which guaranteed that the algorithm had achieved steady state in terms of NMSD and number of nodes sampled. The results are shown in Fig. 36. Along with the experimental data, we also present the result yielded by (3.55) for each γ. Firstly, we observe that the greater the γ, the less nodes are sampled, as expected. Furthermore, it can be seen that the simulation results obtained with DTAS-dNLMS lie below the theoretical upper bound for every value of $\gamma > 1$, while they coincide for $\gamma = 1$, which validates (3.55). It is interesting to notice that, as γ increases, the simulation results obtained with both algorithms approach the theoretical upper bound, whereas they remain far below it for small $\gamma > 1$. In the case of DTRAS-dNLMS, the number of nodes sampled per iteration lies below the theoretical upper bound for $\gamma < 30$. For $\gamma \geqslant 30$, the results yielded by (3.55) slightly surpass the theoretical upper bound due to spurious resets of the sampling system. Nonetheless, this difference between the simulation results and the values yielded by (3.55) are so slim in these cases that it can be neglected. Moreover, we observe that DTRAS-dNLMS in general samples slightly more nodes per iteration than DTAS-dNLMS for the same reason. However, we remark that the difference between both algorithms is less than 1 node per iteration for all values of γ. From Fig. 36, we can see that (3.55) enables a well-informed selection of γ, since it allows the filter designer to know how many nodes would be sampled per iteration in a worst-case scenario.

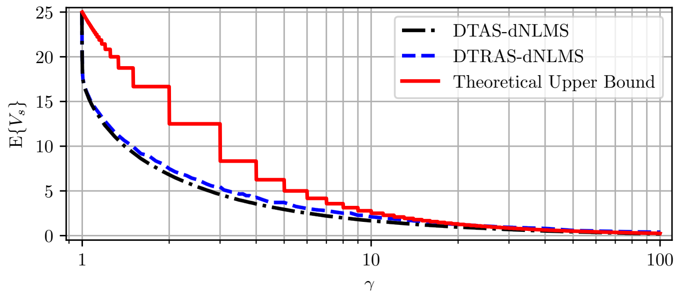


Figure 36: Theoretical results yielded by (3.55) and average number of nodes sampled by DTAS-dNLMS and DTRAS-dNLMS ($\chi = 1.298$) as a function of $\gamma \geqslant 1$.

3.2.6.2 Scenario 2 – Network with a Noisy Node

We now analyze a scenario in which one of the nodes is much noisier than the remainder of the network. Thus, starting from the base scenario, we increase the noise power $\sigma^2_{\max}$ of the noisiest node from Fig. 34(a) by tenfold.

Hence, in Fig. 37, we resume the simulations of Figs. 29 and 30, with the addition of the DTRAS-dNLMS algorithm. In Fig. 37(a) we present the NMSD curves, in Fig. 37(b) the number of nodes sampled, and in Fig. 37(c) the number of multiplications per iteration. The parameters of the AS-dNLMS and DTAS-dNLMS algorithms are the same as those used in Figs. 29 and 30, whereas for DTRAS-dNLMS we consider $\gamma = 11$, $\Delta n = 7 \cdot 10^4$, and $\chi = 1.298$. Moreover, we also show results obtained by the dNLMS algorithm with $V_s = 25$ and $V_s = 5$ nodes sampled per iteration.

We observe from Fig. 37(a) that, unlike AS-dNLMS and DTAS-dNLMS, the DTRAS-dNLMS algorithm was able to roughly maintain the convergence rate of dNLMS with $V_s = 25$ nodes sampled per iteration even after the abrupt change, while sampling the same number of nodes per iteration as AS-dNLMS with $\beta = 0.7\sigma^2_{\max}$ and DTAS-dNLMS in steady state, as we can see from Fig. 37(b). Hence, we observe from this simulation that DTRAS-dNLMS addresses the main limitations of AS-dNLMS and DTAS-dNLMS when an abrupt change occurs in the optimal system. Moreover, much like DTAS-dNLMS, its online estimation of the measurement noise power eliminates the need for *a priori* knowledge of this information.

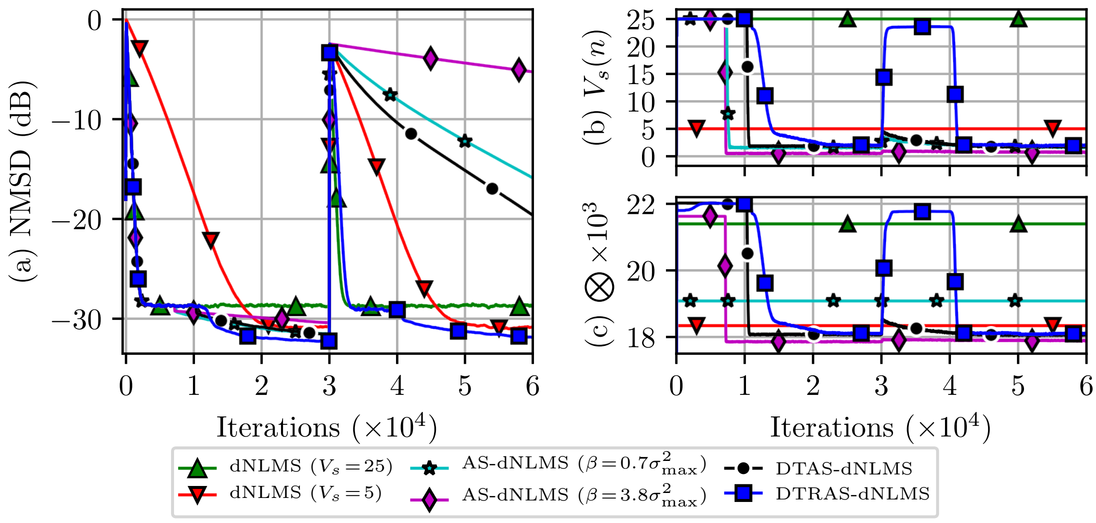


Figure 37: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 19 = 3.8\sigma^2_{\max}$, $\mu_\zeta = 0.0045$, and $\beta = 3.5 = 0.7\sigma^2_{\max}$, $\mu_\zeta = 0.0025$), DTAS-dNLMS ($\gamma = 9$, $\Delta n = 7000$), and DTRAS-dNLMS ($\gamma = 11$, $\chi = 1.298$, $\Delta n = 7000$) in a scenario where $\sigma^2_{\max}$ is increased by tenfold in comparison with Fig. 34(b). (a) NMSD curves, and (b) number of nodes sampled per iteration.

3.2.6.3 Scenario 3 – Random-Walk Tracking

In this section, we investigate the behavior of the proposed algorithms in nonstationary environments following a random-walk model, similarly to what was done in Sec. 3.1.6.4. Starting from Scenario 1, we consider that the optimal solution $\mathbf{w}^{\mathrm{o}}(n)$ varies according to (3.39). As in Sec. 3.1.6.4, we consider a Gaussian distribution for $\mathbf{q}(n)$ with $\mathbf{Q} = \sigma_q^2\mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix.

In Fig. 38, we present the results obtained with the AS-dNLMS, DTAS-dNLMS, and DTRAS-dNLMS algorithms for different values of $\mathrm{Tr}[\mathbf{Q}]$. For comparison, we also show results obtained by dNLMS with $V_s = 25$ and $V_s = 2$. Moreover, for DTRAS-dNLMS, we considered two values of χ: 1.3 and 1.2. The other parameters of the aforementioned algorithms were maintained from the simulations of Fig. 35. In Fig. 38(a), we present the steady-state levels of NMSD, and in Fig. 38(b) the average number of sampled nodes per iteration. The results presented were obtained by averaging the data over the last $70 \cdot 10^3$ iterations of each realization, after all the algorithms achieved steady state.

We observe from Fig. 38(a) that higher values for $\mathrm{Tr}[\mathbf{Q}]$ lead to worse steady-state performances by all algorithms, which was expected. However, the rate at which the steady-state NMSD deteriorates as we increase Tr$[\mathbf{Q}]$ varies from one solution to another. The dNLMS algorithm with $V_s = 25$ presents a slightly higher steady-state NMSD for Tr$[\mathbf{Q}] = 10^{-8}$, but a better performance for Tr$[\mathbf{Q}] \geqslant 10^{-6}$ in comparison with the other solutions. A possible interpretation for this is that, in the former case, the scenario is similar to that of Fig. 35. However, as Tr$[\mathbf{Q}]$ increases, it becomes more important to sample the nodes because it allows the algorithm to keep better track of the changes in the environment. Comparing Figs. 38(a) and 38(b), we observe that the more nodes are sampled, the better the performances of the algorithms for Tr$[\mathbf{Q}] \geqslant 10^{-6}$. Furthermore, dNLMS with $V_s = 2$ and DTAS-dNLMS present similar results and the highest NMSD for higher values of Tr$[\mathbf{Q}]$. As for AS-dNLMS and DTRAS-dNLMS with $\chi = 1.3$, we observe that their performances are similar to those of DTAS-dNLMS for Tr$[\mathbf{Q}] \leqslant 10^{-6}$, but are superior for Tr$[\mathbf{Q}] = 10^{-5}$ and Tr$[\mathbf{Q}] = 10^{-4}$. For Tr$[\mathbf{Q}] = 10^{-4}$, DTRAS-dNLMS with $\chi = 1.3$ performs noticeably better than AS-dNLMS. By changing the value of χ, we can control the behavior of the proposed algorithm, since for Tr$[\mathbf{Q}] \geqslant 10^{-7}$ DTAS-dNLMS with $\chi = 1.2$ performs better than AS-dNLMS and DTAS-dNLMS, as well as dNLMS with $V_s = 2$ and DTRAS-dNLMS. From Fig. 38(b) we observe that the DTRAS-dNLMS with $\chi = 1.2$ samples more nodes per iterations than all other solutions, except for dNLMS with $V_s = 25$. Thus, it is able to keep better track of the changes in the optimal system, which explains the improvement in the performance.

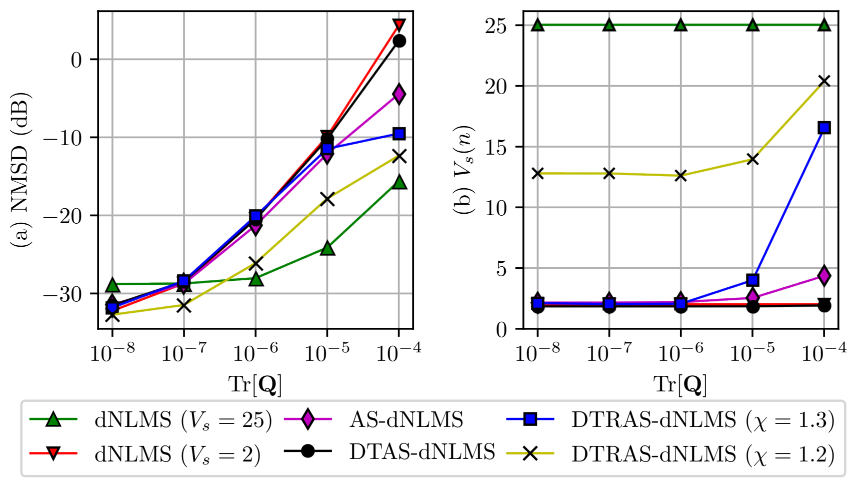


Figure 38: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 1.9$, $\mu_\zeta = 0.0045$), DTAS-dNLMS ($\gamma = 9$, $\Delta n = 7000$), and DTRAS-dNLMS ($\gamma = 11$, $\Delta n = 7000$, and different values for χ) in a nonstationary environment following Model (3.39). (a) steady-state NMSD, and (b) average number of nodes sampled per iteration.

The sampling mechanism of DTRAS-dNLMS algorithm presents a cyclic behavior in the case of $\text{Tr}[\mathbf{Q}] = 10^{-4}$. In Figs. 39(a) and 39(b), we respectively present the NMSD and number of nodes sampled per iteration under these circumstances. The number of nodes sampled per iteration by the DTRAS-dNLMS algorithm with $\chi = 1.3$ oscillates intensely during the first $2 \cdot 10^5$ iterations. Consequently, the NMSD also fluctuates greatly after the initial convergence. As time goes by, both of these oscillations decrease in amplitude, but never cease completely. An interpretation for this phenomenon lies in the reset system of the sampling mechanism. Since the variations in the optimal system are swift when $\text{Tr}[\mathbf{Q}] = 10^{-4}$, the lack of sampling heavily impacts the performance and, consequently, the error magnitude in each node. Thus, the reset mechanism is activated, which resumes the sampling of the nodes. This, in its turn, improves the tracking capability of the algorithm and decreases the magnitude of the error. However, such decrease leads to a reduction in the number of nodes sampled once again due to (3.9). Hence, the oscillations in the number of sampled nodes arise. Over time, they stabilize, but do not die out. From Figs. 39(a) and 39(b), we can see that the DTRAS-dNLMS algorithm with $\chi = 1.2$ also presents fluctuations, but these are much slighter in comparison. The adoption of a lower value for χ makes the reset system activate much more easily, which reduces the impact of oscillations in the error magnitude on the number of nodes sampled.

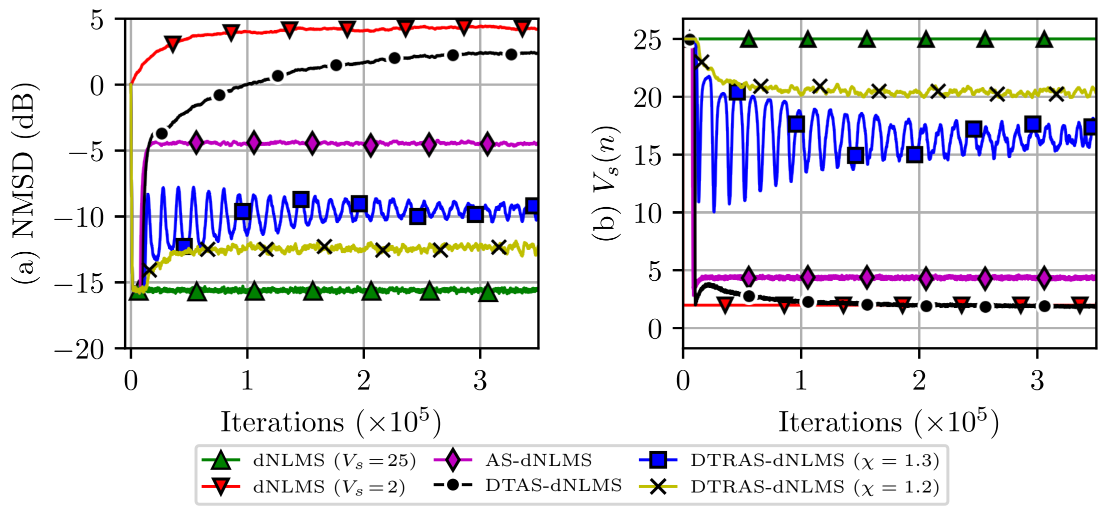


Figure 39: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 1.9$, $\mu_\zeta = 0.0045$), DTAS-dNLMS ($\gamma = 9$, $\Delta n = 7000$), and DTRAS-dNLMS ($\gamma = 11$, $\chi = 1.298$, $\Delta n = 7000$) in a scenario with random-walk tracking as in (3.39) with $\text{Tr}[\mathbf{Q}] = 10^{-4}$. (a) NMSD curves, and (b) number of nodes sampled per iteration.

Finally, in the simulations of Fig. 40, we repeat the experiments of Fig. 38 considering only the DTRAS algorithm with $1.15 \leqslant \chi \leqslant 1.45$. The lower the value of χ, the more nodes are sampled for all values of $\text{Tr}[\mathbf{Q}]$, as expected. By selecting $\chi = 1.15$, the reset mechanism maintained the sampling of all nodes. Moreover, for $\chi \geqslant 1.3$, the differences in performance and number of nodes sampled are slight for $\mathrm{Tr}[\mathbf{Q}] \leqslant 10^{-6}$. As $\mathrm{Tr}[\mathbf{Q}]$ increases, these disparities become more noticeable. When the changes in the optimal system are slow, choosing $\chi \geqslant 1.3$ prevents the reset mechanism from resetting spuriously. This is beneficial in the cases of stationary environments and of $\mathrm{Tr}[\mathbf{Q}] = 10^{-8}$, but deteriorates the performance as the variations in the optimal system become faster, e.g., $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-7}$. As these changes become even swifter, e.g. $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-5}$, the reset mechanism begins to act more noticeably. Thus, the selection of different values for χ leads to prominent discrepancies in the sensitivity of the reset system, which impacts the number of nodes sampled per iteration and the performance. In contrast, by selecting $1.2 \leqslant \chi \leqslant 1.25$, the reset mechanism activates for all values of $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-8}$. This is especially noticeable for $\chi = 1.2$, and, similarly to what was observed in Fig. 38, leads to more nodes sampled per iteration and a lower steady-state NMSD for $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-8}$.

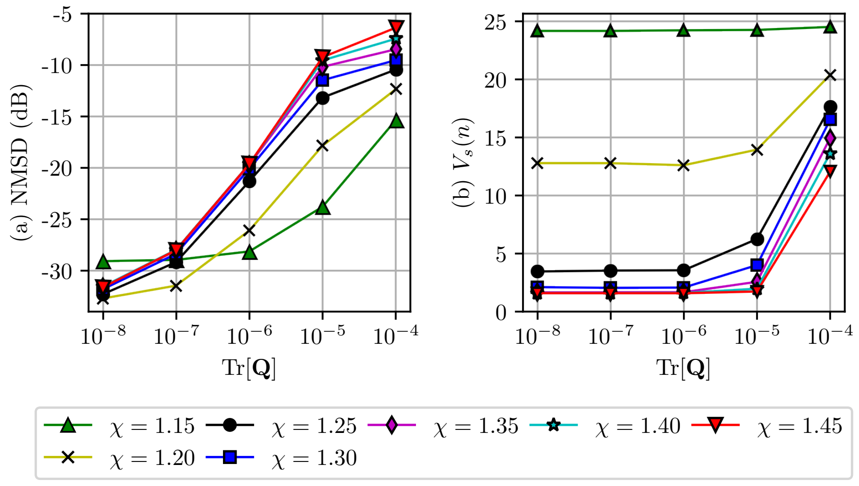


Figure 40: Simulation results obtained in a nonstationary environment following Model (3.39) with DTRAS-dNLMS ($\gamma = 11$, $\Delta n = 7000$ and different values for χ). (a) Steady-state NMSD, and (b) Number of nodes sampled per iteration.

In the presence of impulsive noise, higher values for χ lead to lower computational costs as well as improved performance, whereas in nonstationary environments lower values are required to maintain the performance. Moreover, in this case, there is a trade-off between computational cost and NMSD. Nonetheless, it should be noted that the parameter χ grants the DTRAS-dNLMS algorithm a great degree of flexibility, which makes it suitable for different scenarios and applications.

3.2.6.4 Scenario 4 – Colored Input and diffusion Affine Projection Algorithm

Unlike what was done in the experiments of Secs. 3.2.6.1, 3.2.6.2, and 3.2.6.3, in the simulations of this section we consider a colored input signal $u_k(n)$ at each node k, given by

$$u_k(n) = r_k(n) - 0.8u_k(n-1), \tag{3.68}$$

where $r_k(n)$ is white Gaussian with zero mean and unit variance for $k = 1, \cdots, V$. Furthermore, we consider a filter length of $M = 150$. As in Sec. 3.2.6.1, the coefficients of the optimal system $\mathbf{w}^{\mathrm{o}}$ were randomly generated following a Uniform distribution in the range $[-1, 1]$, and then normalized to ensure that $\mathbf{w}^{\mathrm{o}}$ has unit norm. In the middle of each realization, we multiply $\mathbf{w}^{\mathrm{o}}$ by 0.25 to simulate an abrupt change in the environment. The noise variances and step sizes are the same as in Fig. 34.

In order to illustrate how the proposed sampling mechanism can be used in conjunction with other types of diffusion algorithms aside from dNLMS, we apply it to dAPA [185, 186]. Its adaptation step is given by

$$\boldsymbol{\psi}_k(n+1) = \mathbf{w}_k(n) + \widetilde{\mu}_k \mathbf{U}_k(n)[\delta_r \mathbf{I} + \mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)]^{-1}\mathbf{e}_k(n), \tag{3.69}$$

where $\mathbf{U}_k(n) = [\mathbf{u}_k(n)\ \mathbf{u}_k(n-1)\ \cdots\ \mathbf{u}_k(n-L+1)]$, $\mathbf{e}_k(n) = \mathbf{d}_k(n) - \mathbf{U}_k^{\mathrm{T}}(n)\mathbf{w}_k(n)$, and $\mathbf{d}_k(n) = [d_k(n)\ d_k(n-1)\ \cdots\ d_k(n-L+1)]^{\mathrm{T}}$, with $L \leqslant M$ being a parameter that the filter designer must choose [125, 186]. The adoption of greater values for L usually increases the convergence speed, but also deteriorates its steady-state performance [186]. If $L = 1$ is chosen, the algorithm coincides with dNLMS. Finally, the combination step of dAPA is also given by (2.50b) [186].

To incorporate the sampling mechanisms into dAPA, we introduce the binary sampling variable $\zeta_k(n)$ in the correction term of (3.69) analogously to (2.50a). Hence, if $\zeta_k(n) = 0$, $\mathbf{U}_k^{\mathrm{T}}(n)\mathbf{w}_k(n)$, $\mathbf{e}_k(n)$, $\mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)$, $[\delta_r \mathbf{I} + \mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)]^{-1}$, and $\widetilde{\mu}_k \mathbf{U}_k(n)[\delta_r \mathbf{I} + \mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)]^{-1}\mathbf{e}_k(n)$ do not have to be calculated. If $\zeta_k(n) = 1$, (3.69) is computed as usual. Unlike in the dNLMS adaptation, we consider that the desired signal $d_k(n)$ is sampled even if $\zeta_k(n) = 0$, since its value may be necessary to form the vector $\mathbf{d}_k$ at future iterations. Furthermore, we continue to use the instantaneous error $e_k(n)$ given by (2.7) for the adaptation of the sampling mechanisms, rather than the error vector $\mathbf{e}_k(n)$.

In Fig. 41 we present a comparison between DTRAS-dAPA, DTAS-dAPA, AS-dAPA and dAPA with $V_s = 3$ nodes sampled randomly, as well as dAPA with all $V_s = 25$ nodes sampled. We adopted $L = 4$ for all of the aforementioned algorithms. In Fig. 41 (a) we show the NMSD curves, in Fig. 41(b) the number of nodes sampled, and in Fig. 41(c) the number of multiplications per iteration, respectively. The parameters of the sampling mechanisms of each algorithm were selected so as to obtain roughly the same steady-state NMSD for every solution. In the case of dAPA with every node sampled, we can see from Fig. 41 (a) that the steady-state NMSD is about 10 dB higher in comparison with the other algorithms, even though the step sizes are the same for every solution. Thus, we observe that the difference in steady-state performance entailed by the sampling of less nodes is more pronounced in comparison with the simulations using the dNLMS algorithm, such as in Fig. 35. We adopted $\beta = 4\sigma^2_{\max} = 2$ and $\mu_\zeta = 0.0098$ for AS-dAPA, $\gamma = 9.5$ and $\Delta n = 10^3$ for DTAS-dAPA, and $\gamma = 9.5$, $\Delta n = 500$, and $\chi = 1.2328$ for DTAS-dAPA. The value for χ was obtained by replacing $M = 150$ in (3.62). In this case, DTRAS-dNLMS sampled on average 2.1 nodes per iteration, whereas AS-dAPA and dAPA respectively sampled 2.2 and 2.6 nodes per iteration.

Before the abrupt change in the environment, we can see from Fig. 41 (a) that DTRAS-dAPA converges to a steady-state NMSD of -30 dB faster than dAPA with $V_s = 3$, albeit slower than AS-dAPA and DTAS-dAPA. This occurs since it maintains the sampling of all the nodes for a longer period of time, during which it behaves similarly to dAPA with every node sampled, as can be attested from Figs. 41 (a) and (b). On the other hand, it resumes to an NMSD level of -30 dB faster than any other solution after the abrupt change, which shows that it has a better tracking capability than the other sampling algorithms. In terms of the computational cost, we can see from Fig. 41 (c) that DTRAS-dAPA, AS-dAPA, DTAS-dAPA and dAPA with with $V_s = 3$ demanded a similar number of multiplications per iteration in steady state. Although the cost of DTRAS-dAPA is higher than that of dAPA with all nodes sampled during the transient, the difference in this case is negligible in comparison with the overall cost of both algorithms. This contrasts with Fig. 35 (c), in which the difference in cost between DTRAS-dNLMS and dNLMS with all nodes sampled during the transient is more noticeable, since the dNLMS algorithm is less costly overall than dAPA with $L > 1$.

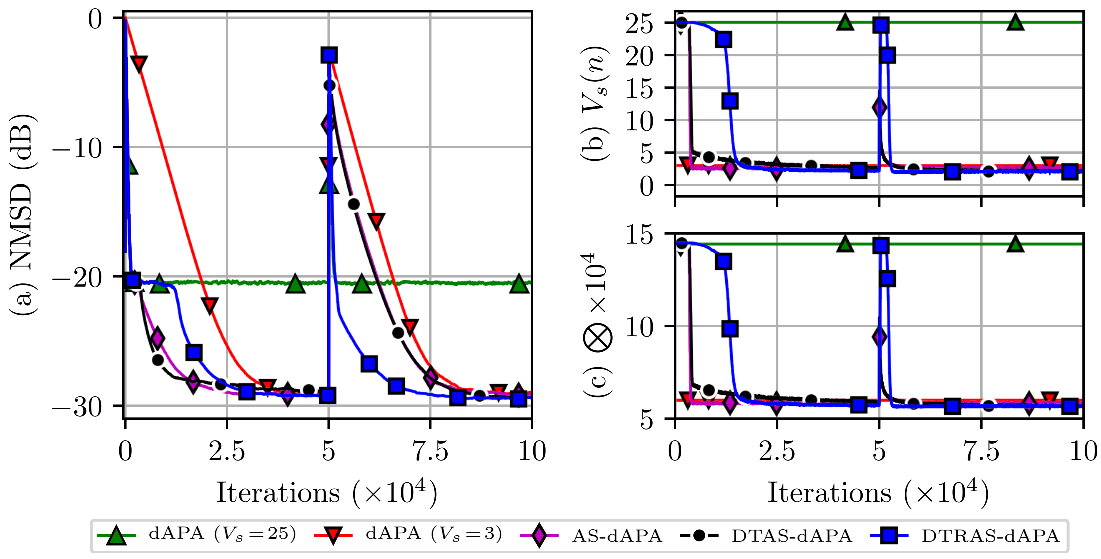


Figure 41: Comparison between dAPA with V_s nodes randomly sampled per iteration, AS-dAPA ($\beta = 2$, $\mu_\zeta = 0.0098$), DTAS-dAPA ($\gamma = 9.5$), and DTRAS-dAPA ($\gamma = 9.5$, $\chi = 1.2328$). For DTAS-dAPA and DTRAS-dAPA, $\mu_{\zeta_k}(n)$ was set using (3.50) with $\Delta n = 1000$ and $\Delta n = 500$, respectively. (a) NMSD curves, (b) number of nodes sampled and (c) multiplications per iteration.

3.3 Adaptive Sampling and Censoring for Kernel-Based Diffusion Networks

In this section, we extend the algorithm of Sec. 3.2 to RFF kernel-based adaptive diffusion networks. For simplicity, we restrict our attention to the DTAS technique of Algorithm 7, without the reset mechanism for the sampling proposed in Sec. 3.2.3. The analysis on the choice of the parameter p_χ carried out in that section would have to be redone for the RFF kernel algorithm in order to enable the usage of the proposed reset mechanism in this context. This poses an interesting challenge, which we consider a promising topic for future research.

Moreover, we also incorporate the concept of censoring into the DTAS algorithm of Sec. 3.2, similarly to what was done to the AS technique in Sec. 3.1.1. We would like to remark that that the concepts of sampling and censoring may be especially relevant for kernel-based diffusion networks than for the more “conventional” adaptive diffusion networks of Sec. 2.2. This is because kernel methods tend to present greater computational costs in comparison with linear solutions, which makes their application more challenging in practice. Hence, we believe that a technique for reducing the computational and energy costs associated with the learning task may be even more helpful in this context.

We begin by modifying (2.45a), similarly to what was done to (2.12a) in Sec. 3.1.1 with

Eq. (3.10). Thus, we obtain

$$\boldsymbol{\psi}_k(n+1) = [1-\zeta_k(n)]\boldsymbol{\psi}_k(n) + \zeta_k(n)\big[\mathbf{w}_k(n)+\mu_k(n)\breve{e}_k(n)\mathbf{z}_k(n)\big], \tag{3.70}$$

where we have introduced

$$\breve{e}_k(n) = d_k(n) - \mathbf{z}_k^{\mathrm{T}}(n)\mathbf{w}_k(n) \tag{3.71}$$

for compactness. Moreover, we remark that in (3.70) we are considering a normalized step size $\mu_k(n)$, similarly to the one adopted in (2.53). Then, replacing $e_i(n)$ with $\breve{e}_i(n)$ in Eq. (3.3), we arrive at the following cost function:

$$J_{\zeta,k}^{\mathrm{RFF}}(n) = \big[\bar{\zeta}_k(n)\big]\beta\zeta_k(n)+\big[1-\bar{\zeta}_k(n)\big]\sum_{i\in\mathcal{N}_k}c_{ik}(n)\breve{e}_i^2(n), \tag{3.72}$$

where $\bar{\zeta}_k(n)$ is defined as in (3.2). Following a rationale similar to the one employed in Sec. 3.1, we introduce the auxiliary variable α_k, and derive (3.72) with respect to it. Thus, we arrive at the following stochastic gradient descent rule:

$$\alpha_k(n+1)=\alpha_k(n)+\mu_\zeta\phi_k'(n)\left[\sum_{j\in\mathcal{N}_k}c_{jk}\breve{\varepsilon}_j^2(n)-\beta\zeta_k(n)\right], \tag{3.73}$$

where $\mu_{\zeta_k} > 0$ is a step size as before, $\phi_k'(n)$ is defined as in (3.6), and

$$\breve{\varepsilon}_k(n) = [1-\zeta_k(n)]\breve{\varepsilon}(n-1) + \zeta_k(n)e_k(n) \tag{3.74}$$

denotes the last measurement of $\breve{e}_k$ that we have access to, due to the sampling of node k.

Much like the AS-dNLMS algorithm of Sec. 3.1, the choice of the parameter β in (3.73) depends on the measurement noise power profile throughout the network. To circumvent this, we replace β by $\beta_k(n)$ given by Eq. (3.40) and μ_ζ with $\mu_{\zeta_k}(n)$ by (3.50) in (3.73). One thus obtains

$$\alpha_k(n+1)=\alpha_k(n)+\mu_{\zeta_k}(n)\phi_k'(n)\left[\sum_{j\in\mathcal{N}_k}c_{jk}\breve{\varepsilon}_j^2(n)-\gamma\hat{\sigma}_{\mathcal{N}_k}^2(n)\zeta_k(n)\right]. \tag{3.75}$$

The resulting algorithm is named as Dynamic-Tuning Adaptive-Samping-and-Censoring (DTASC) RFF-dKNLMS.

3.3.1 Simulation results

Next, we compare the performance of our DTASC-RFF-dKNLMS algorithm to that of other state-of-the-art solutions considering two simulation scenarios (Secs. 3.3.1.1 and 3.3.1.2), and using the network with $V = 20$ nodes depicted in Fig. 42(a). The noise power $\sigma_{v_k}^2$ is shown in Fig. 42(b) for $k = 1, \cdots, V$, and the input signal $u_k(n)$ is a white Gaussian noise with zero mean and unit variance. The results were obtained over an average of 500 realizations.

Besides the proposed DTASC-RFF-dKNLMS scheme, we also consider in the simulations the RFF-dKNLMS algorithm with two different censoring approaches: (i) the censoring mechanism of [83], and (ii) a random censoring policy, in which V_s nodes are selected randomly at each iteration to remain uncensored. To enable the comparison between DTASC-RFF-dKNLMS and the censoring strategy of [83], their parameters γ and τ_n were selected to obtain roughly one node uncensored per iteration in steady state in each case. In the random censoring technique, censored nodes are prevented from updating their local estimates, in a similar fashion to DTASC-RFF-dKNLMS. In addition, and for the sake of comparison, we also show results obtained with three other approaches: the RFF-dKNLMS algorithm with all nodes uncensored, the RFF-dKNLMS scheme with a noncooperative communication policy, and the linear dNLMS with all nodes uncensored. In all cases, we adopt $\widetilde{\mu}_k = 1$ for every node k and $\delta_r = 10^{-5}$.

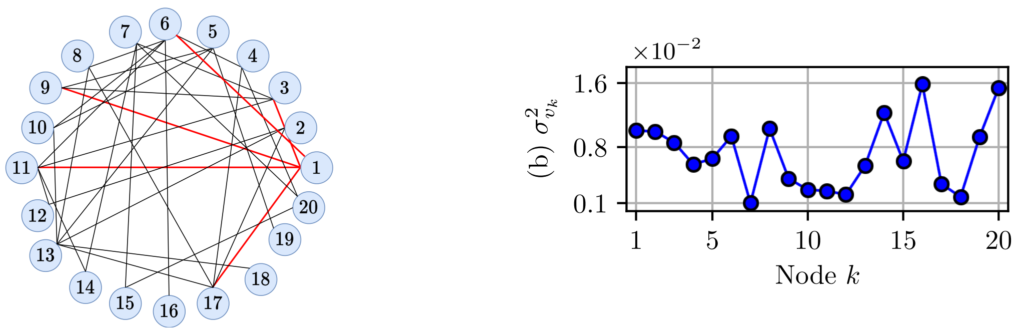


Figure 42: (a) Network topology, with the connections between node #1 and its neighbors highlighted, and (b) $\sigma_{v_k}^2$ used in the experiments.

3.3.1.1 Toy Example

Firstly, we consider a toy example in which $d_k(n) = \mathbf{z}_k^{\mathrm{T}}(n)\mathbf{w}^{\mathrm{o}} + v_k(n)$, where $\mathbf{w}^{\mathrm{o}}$ is an unknown system [1]. In our simulations, we randomly generated the $D = 50$ coefficients of $\mathbf{w}^{\mathrm{o}}$ following a Uniform distribution in $[-1,1]$. At each iteration, $\mathbf{z}_k(n)$ is generated using the past $M = 2$ samples of $u_k(n)$ and RFFs drawn from a multivariate Gaussian distribution with $\sigma^2 = 10^{-2}$.

We assume that the RFFs utilized to generate $\mathbf{z}_k(n)$ are the same employed by the RFF-dKNLMS algorithm. Although not realistic, this assumption allows us to adopt the network mean-square deviation (NMSD) as a performance indicator, which is given by $\text{NMSD}(n) = \frac{1}{V}\sum_{k=1}^{V} \text{E}\{\|\mathbf{w}^{\text{o}} - \mathbf{w}_k(n)\|^2\}$. Specifically for the dNLMS algorithm, we adopt $M = D = 50$ to enable the comparison between $\mathbf{w}^{\text{o}}$ and the coefficients of the adaptive filter. In addition, we set $\gamma = 10$ for our DTASC-RFF-dKNLMS scheme and, to preserve its convergence rate, $\Delta n = 5000$. For the censoring mechanism of [83], named COKE by the authors, we adopted $\tau_n = 0.05$. This parameter was set so as to achieve roughly the same number of uncensored nodes per iteration as the DTASC-RFF-dKNLMS algorithm.

In Fig. 43, we show (a) the NMSD curves, (b) the number of uncensored nodes, and (c) the number of multiplications per iteration. We can see that DTASC-RFF-dKNLMS keeps the nodes uncensored during the transient phase [Fig. 43(b)] and thus converges as fast as RFF-dKNLMS with $V_s = 20$ nodes uncensored, which can be considered as a lower bound regarding the NMSD [Fig. 43(a)]. In contrast, the RFF-dKNLMS with $V_s = 2$ nodes randomly uncensored clearly presents a slower convergence rate. The algorithm of [83] outperforms the random censoring technique while censoring more nodes, but converges slightly slower than DTASC-RFF-dKNLMS and the RFF-dKNLMS algorithm with $V_s = 20$ nodes uncensored. In terms of the computational cost, the DTASC-RFF-dKNLMS algorithm demands less multiplications per iteration in steady state than any other solution. On average, DTASC-RFF-dKNLMS reduces the burden by roughly 95% in comparison with the RFF-dKNLMS algorithm with $V_s = 20$ nodes uncensored and 90% in comparison with the solution of [83] in steady state. This clearly compensates the slight increment in computational burden during the transient phase.

3.3.1.2 Nonlinear Channel Identification

In this subsection, we consider that $d_k(n) = y_k(n) + 0.2y_k^2(n) - 0.1y_k^3(n) + v_k(n)$, where $y_k(n)$ is the output of a linear channel described by $H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ with input $u_k(n)$. In this case, we consider $M = 3$ and $D = 500$ features generated with $\sigma^2 = 1$. Moreover, we adopt the network MSE (NMSE) as a performance indicator, which is given by $\text{NMSE}(n) = \frac{1}{V}\sum_{k=1}^{V}\text{E}\{\breve{e}_k^2(n)\}$, and shown in Fig. 3(a) for all the algorithms. We selected

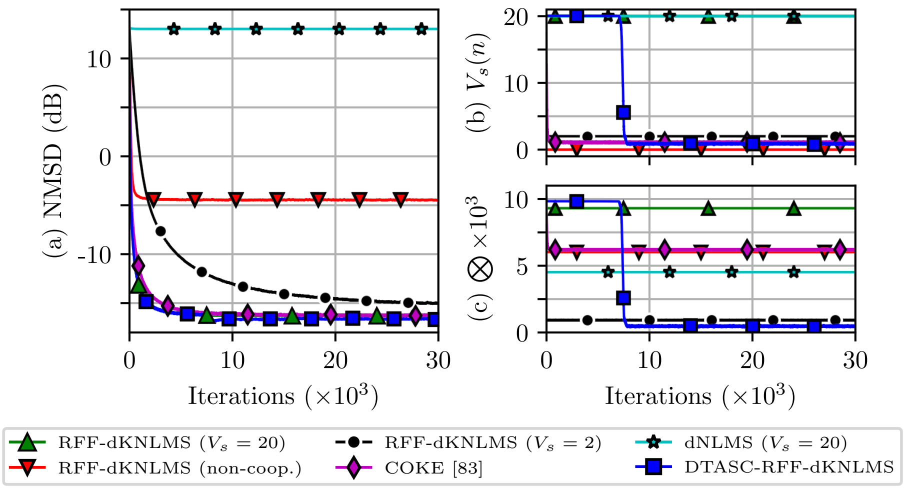


Figure 43: Simulation results obtained in the toy example scenario. (a) NMSD curves, (b) number of uncensored nodes, and (c) multiplications per iteration.

$\gamma = 10$ for the DTASC-RFF-dKNLMS algorithm and set $\Delta n = 10^4$ in order to preserve its convergence rate. For the solution of [83], we considered $\tau_n = 10^{-3}$. As in the previous simulation, this parameter was selected so as to obtain roughly the same number of uncensored nodes per iteration as DTASC-RFF-dKNLMS at steady state.

We observe from Fig. 44 that, similarly to what was seen in Fig. 43, DTASC-RFF-dKNLMS keeps the nodes uncensored during the transient phase and thus preserves the convergence rate of RFF-dKNLMS with $V_s = 20$ nodes uncensored. In terms of the computational cost, the DTASC-RFF-dKNLMS algorithm carries out less multiplications per iteration in steady state in comparison with any other solution, except for the linear dNLMS algorithm, whose cost was much lower in this scenario due to the adoption of $M = 3$. Once again, we can see that DTASC-RFF-dKNLMS outperforms the censoring algorithm of [83] in terms of the convergence rate and computational cost.

3.4 Adaptive Sampling for Multitask Diffusion Networks

In this section, we extend the AS-dNLMS algorithm of Sec. 3.1 to the multitask networks of Sec. 2.2.4. Similarly to the single-task case, the idea is that, if node k is not sampled, the gradient-related correction term should not be added to the combined estimate $\mathbf{w}_k(n)$ in the adaptation step, as was done in Eq. (3.1). Applying this idea to Eq. (2.29a), one obtains

$$\boldsymbol{\psi}_k(n+1) = \mathbf{w}_k(n) + \mu_k \zeta_k(n) \left\{ \sum_{i \in \mathcal{N}_k \cap C(k)} \frac{b_{ik} e_{ik}(n) \mathbf{u}_i(n)}{\delta_r + \|\mathbf{u}_i(n)\|^2} + \eta \sum_{i \in \mathcal{N}_k \setminus C(k)} \bar{\rho}_{ki} [\mathbf{w}_i(n) - \mathbf{w}_k(n)] \right\}, \quad (3.76)$$

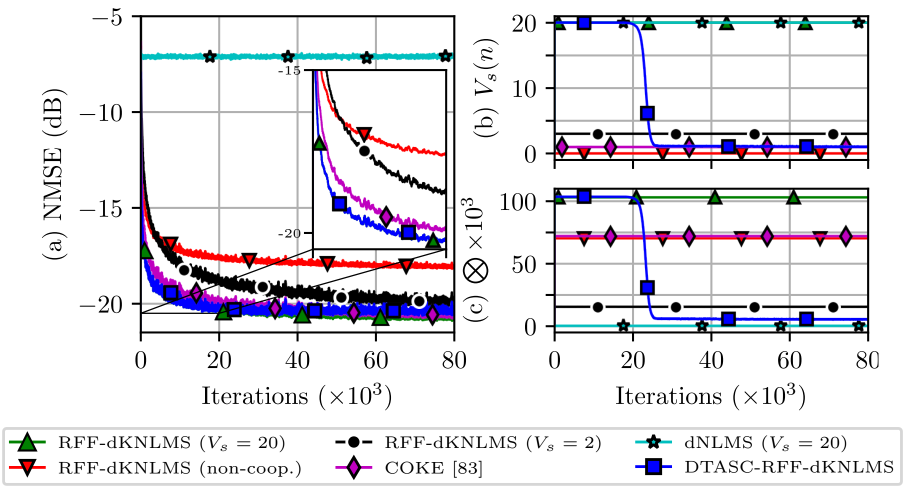


Figure 44: Simulation results obtained for nonlinear channel identification. (a) NMSE curves, (b) number of uncensored nodes, and (c) multiplications per iteration. For better visualization, we applied a moving-average filter with 64 coefficients to the curves.

where we normalized the first summation by the norm of the regressor vector $\mathbf{u}_i(n)$. This makes it easier to choose the adaptation steps μ_k, since the selection of this parameter depends on the norm of the regressor vectors for the non-normalized versions of the algorithm [1, 125]. We recall that, as mentioned in Sec. 2.2.4, the b_{ik} are convex weights satisfying (2.10), as mentioned in Secs. 2.2 and 2.2.4.

As before, the question that arises is how to adjust the sampling. In this regard, we remark that it would be convenient if each node k were influenced only by neighbors belonging to the same cluster, as these are the ones that share the same optimal system. This change in relation to the single-task algorithm may allow, for example, clusters whose nodes have already reached estimates sufficiently close to their optimal systems to no longer be sampled. On the other hand, in clusters where the learning task is still far from steady-state performance, it would be possible to maintain the adaptation of the nodes. In both scenarios, adapting the sampling mechanism to nodes in one cluster does not affect those belonging to other clusters.

Furthermore, comparing Eqs. (2.12a) and (2.29a), we notice that a difference between the multitask dLMS and the single-task one consists in the fact that, in the first case, each node k uses information from all neighbors belonging to the same cluster already in the adaptation stage. This could also be incorporated into the sampling mechanism in its multitasking version. Thus, we change the cost function of Eq. (3.3) to

$$J_{\zeta,k}^{\text{multi.}}(n) = \left[\bar{\zeta}_k(n)\right]\beta\zeta_k(n) + \left[1-\bar{\zeta}_k(n)\right] \sum_{i\in\mathcal{N}_k\cap C(k)} c_{ik}\bar{e}_i(n), \tag{3.77}$$

where

$$\bar{e}_i(n) = \sum_{j \in \mathcal{N}_i \cap C(i)} b_{ji} e_{ji}^2(n) \tag{3.78}$$

uses the data from $d_j(n)$ and $\mathbf{u}_j(n)$ from every neighboring node j belonging to the same cluster as node i.

3.4.1 Simulation Results

In this section, simulation results are presented to verify the performance of the multitask AS-dNLMS algorithm. These results were obtained from an average of 100 independent realizations. To facilitate visualization of the results, a moving average filter with 64 coefficients was applied to the curves presented.

For the simulations, a network was randomly generated with 28 nodes grouped into four different clusters, each with seven nodes, numbered from C_1 to C_4 . The resulting network is shown in Fig. 45(a). Furthermore, each node k is subject to a different noise variance $\sigma_{v_k}^2$ and uses a distinct adaptation step μ_k, as shown in Fig. 45(b).

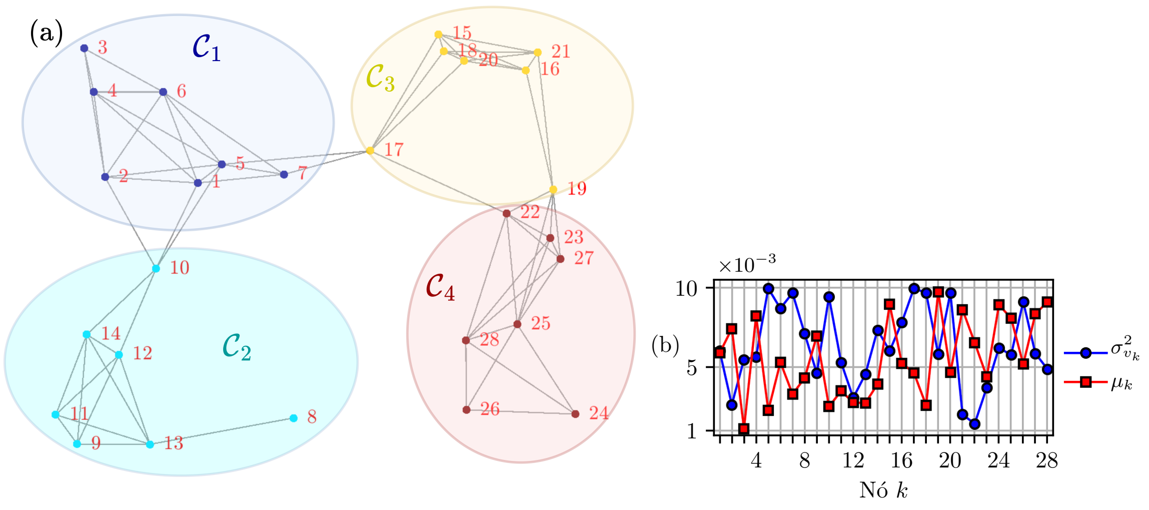


Figure 45: (a) The network considered in the simulations, in which the nodes are grouped into four distinct clusters, numbered as C_1 to C_4. (b) Noise power $\sigma_{v_k}^2$ and adaptation step μ_k for $k = 1, \cdots, V$.

The input signal $u_k(n)$ is Gaussian with zero mean and unit variance for all nodes k, $k = 1, \cdots, V$. The optimal system $\mathbf{w}_{C_i}^{\text{o}}$ associated with cluster C_i was generated as

$$\mathbf{w}_{C_i}^{\text{o}} = \mathbf{w}_{\text{common}}^{\text{o}} + \mathbf{w}_{\text{local}_i}^{\text{o}},$$

where $\mathbf{w}_{\text{common}}^{\text{o}}$ is a portion common to all clusters and $\mathbf{w}_{\text{local}_i}^{\text{o}}$ is specific to cluster C_i. The coefficients of the vector $\mathbf{w}^{\mathrm{o}}_{\mathrm{common}}$ were generated according to a Uniform distribution in the interval $[-1,1]$. This vector was then normalized to present unit norm. In contrast, the coefficients of the vectors $\mathbf{w}^{\mathrm{o}}_{\mathrm{local}_i}$ were generated according to a Uniform distribution in the interval $[-2\cdot 10^{-2},\, 2\cdot 10^{-2}]$. The length of the optimal systems is $M = 10$. Furthermore, to simulate an abrupt change in the environment, halfway through each realization the order of the vector coefficients $\mathbf{w}^{\mathrm{o}}_{\mathcal{C}_i}$ is reversed.

For the selection of the combination weights c_{jk} and b_{jk} in (2.29a), (2.29b), and (3.76), we consider the normalized version of ACW, as in Secs. 3.1 and 3.2. Moreover, we adopted $\delta_r = \delta_c = 10^{-5}$ as regularization factors. To simplify the choice of the parameters, we adopted $c_{ik} = b_{ik}$ for every i and k. For the weights $\bar{\rho}_{ik}$, we adopted the Uniform rule. For the selection of η, we ran preliminary simulations considering the scenario described with different values for this parameter. Since $\eta = 10^{-3}$ led to the best steady-state performance in our tests, we adopted this value in the simulations. As a performance indicator, we adopt the NMSD.

First, a comparison is made between the multitask AS-dNLMS and dNLMS algorithms with V_s nodes randomly sampled at each iteration, with $V_s \in \{7, 14, 21, 28\}$. It is worth noting that $V_s = 28$ corresponds to the case in which all nodes are sampled. For reference, the result obtained with the single-task dNLMS algorithm for the same simulation scenario with all nodes sampled is also presented. For each algorithm, the NMSD curves, and the average number of sampled nodes and multiplications per iteration are shown in Figs. 46(a), 46(b), and 46(c), respectively. For the proposed algorithm, $\beta = 0.09$ and $\mu_\zeta = 0.055$ were adopted. These values were selected because they led to the same number of multiplications per iteration in steady state as that of the single-task dNLMS algorithm, while maintaining a good performance. Analyzing Fig. 46(a), we notice that the single-task dNLMS algorithm is outperformed in steady state performance by the multitask solutions, which illustrates the importance of considering regional variations in the optimal system. In regards to the multitask algorithms, we notice that the random sampling significantly affects the convergence rate, as was observed in Secs. 3.1.6 and 3.2.6 for the single-task scenario. On the other hand, the proposed algorithm presents approximately the same convergence rate as that of multitask dNLMS with all sampled nodes. Observing Fig. 46(b), it is possible to see that the AS-dNLMS algorithm keeps all nodes sampled in the transient. Furthermore, the proposed algorithm was able to detect the change in the environment and resume the sampling of the nodes, which allows it to again achieve a convergence rate similar to that of multitask dNLMS after the abrupt change as well. Analyzing Fig. 46, we notice that the proposed algorithm presents a higher computational cost than the dNLMS algorithm with all nodes sampled during the transient. However, after the multitask AS-dNLMS algorithm reaches steady state, the computational cost is significantly reduced with the decrease in the number of nodes sampled per iteration. On average, the AS-dNLMS algorithm performs approximately 4211 multiplications per iteration during steady state, compared to 6436 for the multitask dNLMS algorithm with all nodes sampled and 4268 for the single-task algorithm.

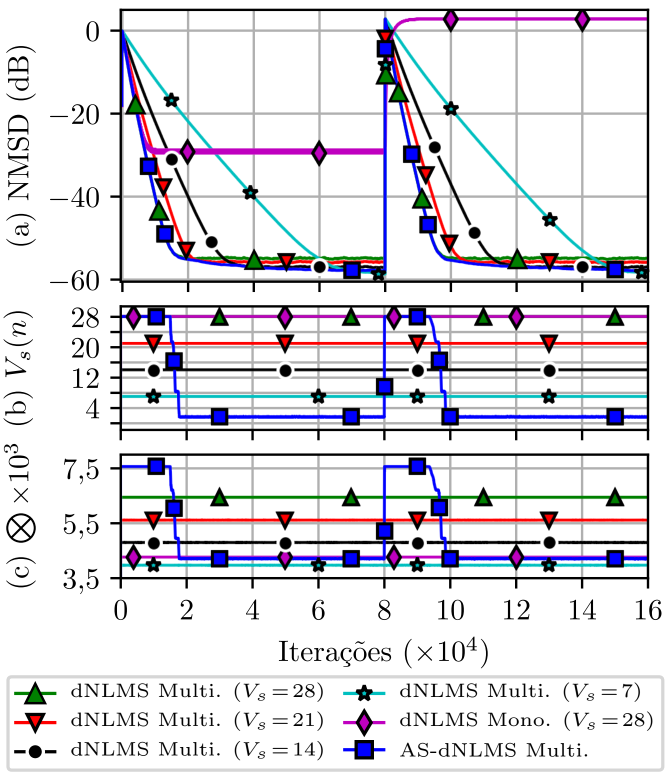


Figure 46: Comparison between the multitask dNLMS algorithm with V_s nodes sampled randomly per iteration ($V_s \in \{7, 14, 21, 28\}$), the multitask AS-dNLMS algorithm (β = 0,09 and μ_ζ = 0,055), and single-task dNLMS algorithm with all nodes sampled. (a) NMSD curves, (b) number of nodes sampled, and (c) number of multiplications per iteration.

In order to verify the effect of the sampling mechanism on different clusters, the previous simulation was repeated with the multitask AS-dNLMS algorithm, but considering a scenario in which the noise variance in the nodes belonging to *cluster* C_4 of Fig. 16(a) was increased by tenfold. The results are shown in Fig. 47. In Fig. 47(a), in addition to the NMSD curve of the network as a whole, we also present curves with the average NMSD of the nodes in each cluster. Furthermore, we show the total number of nodes sampled per iteration in the network as whole in Fig. 47(b), and in Fig. 47 (c) we present the number of nodes sampled in each cluster. We notice that the steady-state NMSD level in the cluster C_4 is greater than that found in the other clusters, as expected. Analyzing Fig. 47(c), we can also see that that, due to the greater noise variance in this cluster, the sampling of its nodes is maintained for more iterations in the transient, in comparison with the other clusters. Furthermore, in steady state, nodes sampled in cluster C_4 than in the rest of the network. For clusters C_1 to C_3, we can see a similar behavior to that observed for the network as a whole in the simulations of Fig. 46. This shows that the sampling mechanism works in an isolated manner in each cluster, which prevents adverse phenomena in a given region from affecting the sampling in the network as a whole.

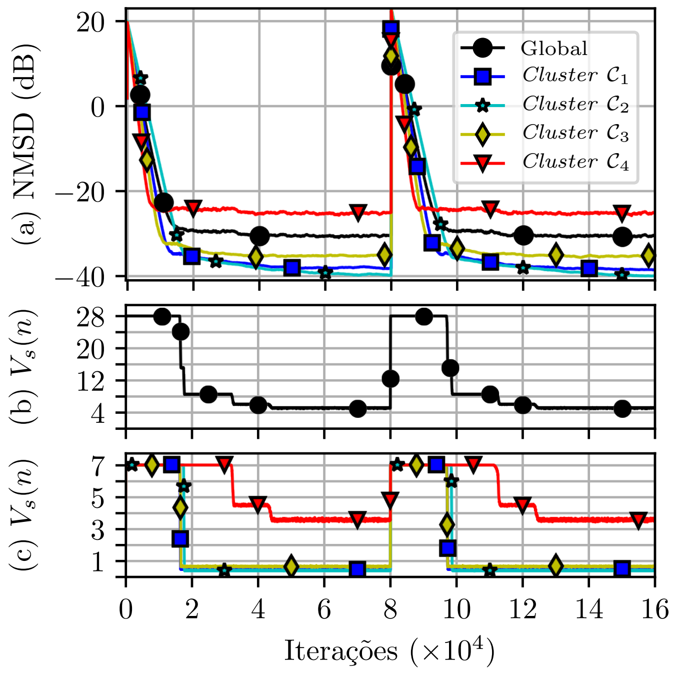


Figure 47: Results obtained with the multitask AS-dNLMS algorithm ($\beta = 0{,}09$ e $\mu_\zeta = 0{,}055$) considering the network of Fig. 16(a) as a whole, and each cluster individually. (a) NMSD curves, (b) Number of nodes sampled per iteration in the network as a whole, and (c) in each cluster.

3.5 Conclusions

In this chapter, several adaptive sampling and censoring mechanisms were derived for diffusion networks. In Sec. 3.1, we presented the AS-dNLMS algorithm, which served as the basis for the solutions proposed in the other sections. As shown in Sec. 3.1, this algorithm maintains the sampling of the nodes when the error is high in magnitude, and ceases to sample them otherwise. As a result, it can perform well under a wide range of scenarios. In particular, we were able to attest its satisfactory performance in a simulation with real-world temperature data. Nevertheless, AS-dNLMS does present two main weaknesses: its tracking capability, as observed in Sec. 3.1.6.4, and the fact that the proper choice of its parameter β depends on prior knowledge of the measurement noise power throughout the network. These weaknesses were addressed in Sec. 3.2, in which two modified versions of AS-dNLMS were presented. By estimating the noise power in an online manner, the DTAS-dNLMS algorithm eliminates the need for prior knowledge of the noise variance, whereas the DTRAS-dNLMS algorithm further enhances the proposed solution by incorporating a mechanism that resets the sampling of the nodes when changes in the environment are detected. Thus, the tracking capability of the DTRAS-dNLMS algorithm is significantly improved in comparison with that of the DTAS-dNLMS and AS-dNLMS algorithms. Then, in Secs. 3.3 and 3.4, we extended the algorithms presented in the previous sections to the kernel-based and multitask networks of Secs. 2.2.5 and 2.2.4, respectively. Simulation results exemplify the good behavior of the proposed sampling and censoring techniques in these scenarios as well.

It should be noted that algorithms have been proposed for the sampling of graph signals in GSP that utilize slightly different frameworks, such a graph spectral-domain concepts [44–46]. These strategies seek to take advantage of the fact that, when dealing with real-world data, graph signals often admit sparse representations in the frequency domain. The goal in these cases is to determine a subset of nodes to be sampled, such that the signal over the whole graph can be recovered from noisy measurements of the signal collected only at that subset. Due to the different nature of these applications in comparison with the ones studied in this chapter, a direct comparison between these techniques and the proposed algorithms is not possible in a straightforward manner. However, it may be interesting to study in future works if the proposed solutions can be modified to work in the spectral domain in GSP problems.

Lastly, it is worth noting that we can establish an analogy between the sampling of the nodes in diffusion networks and the adoption of VSS algorithms in the adaptive filtering literature [269–275]. These types of solutions seek to implement a step size that varies over time according to the estimation error. The idea is that, by employing a greater step size when the error is high, and a smaller one otherwise, we can speed the convergence up while retaining a good steady-state performance [269–275]. Building on this analogy, we have proposed a VSS adaptive filter based on the algorithms from Secs. 3.1 and 3.2 in [276]. However, since this algorithm is not directly related to adaptive diffusion networks apart from its inspiration, we believe that it is out of the scope of this work to present it and explain it in detail.

4 PERFORMANCE ANALYSIS: THE IMPACTS OF SAMPLING

In the simulations of Figs. 17, 28, 29, 35, 37, and 41, we have noticed that, in stationary environments, the sampling of less nodes leads to a slight reduction in the steady-state NMSD. The goal of this chapter is to investigate *why* this occurs through a theoretical analysis on the performance of adaptive diffusion networks with sampling mechanisms. This may be deemed a matter of practical interest, as the aforementioned simulations seem to suggest that, by controlling the sampling of the nodes appropriately, one may simultaneously reduce the computational cost and improve the performance in steady state in stationary environments, while preserving the convergence rate of the original dNLMS algorithm.

To the best of our knowledge, this phenomenon has not been pointed out in the literature. For example, in [277–279], a scenario was considered in which some of the nodes are not capable of performing the adaptation step. These are referred to as "uninformed" nodes, in contrast with the "informed" ones, which can perform it. In those works, it was shown that, in comparison with a network in which every node is informed, the steady-state NMSD can decrease, increase, or remain unchanged, as we turn some of the nodes into uninformed ones. However, differently from the scenario considered in the simulations of Figs. 17, 28, 29, 35, 37, and 41, informed nodes carry out the adaptation step at every time instant, whereas uninformed nodes never do so. Moreover, in this case, the enhancement or the deterioration in the steady-state NMSD depends on the noise power at the informed or uninformed nodes. In other words, prior knowledge of the noise variance at each node is required. This differs from the behavior observed in the simulations of the aforementioned figures, in which the nodes are sampled randomly, and this still leads to a decrease in the steady-state NMSD. Furthermore, in [146–148], the authors study networks in which the adaptation and combination steps are not necessarily carried out simultaneously by all the nodes at every iteration. These networks are referred to as "asynchronous," in contrast with the "synchronous" ones that appear, e.g., in [1–5]. From this perspective, the networks with sampling mechanisms can be deemed as a type of asynchronous network. However, in those papers, the reduction in the steady-state NMSD as a result of sampling was not observed. For this reason, in this section we shall refer to the networks in which the nodes are permanently sampled as "synchronous" ones, whereas the networks in which the sampling of the nodes may cease shall be referred to as "asynchronous."

To simplify the analysis, in this chapter, we shall consider the dLMS algorithm of (2.12), rather than its normalized version considered in Chapter 3. Therefore, we focus our attention on the linear and single-task type of scenario examined in Sec. 2.2. Moreover, for simplicity, we initially analyze the case in which the nodes are sampled randomly. This type of solution is covered in Sec. 4.1. Based on the results derived therein, in Sec. 4.2, we obtain a simplified model for the NMSD of the dTRAS-dLMS algorithm.

4.1 The Effects of Random Sampling

We begin by noticing that the we can incorporate the concept of sampling into the dLMS algorithm of (2.12) in a similar manner to what was done in Sec. 3.1 with Eq. (3.1). Multiplying the correction term in (2.12a) by the binary variable $\zeta_k(n)$, we get

$$\boldsymbol{\psi}_k(n) = \mathbf{w}_k(n-1) + \zeta_k(n)\mu_k e_k(n)\mathbf{u}_k(n) \tag{4.1}$$

In our analysis, we are especially interested in the NMSD. For the clarity of the exposition, it is convenient to introduce the quantities

$$\xi_{ij}(n) \triangleq \mathrm{E}\{\widetilde{\mathbf{w}}_i^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_j(n)\} \tag{4.2}$$

for $i = 1, \cdots, V$ and $j = 1, \cdots, V$, where $\widetilde{\mathbf{w}}$ is the weight error vector given by (2.16). Otherwise, the notation could become overloaded. It is worth noting that

$$\xi_{kk}(n) = \mathrm{E}\{\|\widetilde{\mathbf{w}}_k(n)\|^2\} = \mathrm{MSD}_k(n). \tag{4.3}$$

We then introduce the matrix $\boldsymbol{\Xi}(n)$ such that $[\boldsymbol{\Xi}(n)]_{ij} = \xi_{ij}(n)$, i.e.,

$$\boldsymbol{\Xi}(n) = \begin{bmatrix} \xi_{11}(n) & \xi_{12}(n) & \cdots & \xi_{1V}(n) \\ \xi_{21}(n) & \xi_{22}(n) & \cdots & \xi_{2V}(n) \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{V1}(n) & \xi_{V2}(n) & \cdots & \xi_{VV}(n) \end{bmatrix}, \tag{4.4}$$

which allows us to recast the NMSD as

$$\mathrm{NMSD}(n) = \frac{1}{V}\mathrm{Tr}\{\boldsymbol{\Xi}(n)\}. \tag{4.5}$$

Furthermore, defining the $V^2 \times 1$ vector

$$\boldsymbol{\xi}(n) \triangleq \text{vec}\{\boldsymbol{\Xi}(n)\} = [\xi_{11}(n)\ \xi_{21}(n)\ \cdots\ \xi_{VV}(n)]^{\text{T}}, \tag{4.6}$$

and recalling that

$$\text{Tr}\{\mathbf{M}_1\mathbf{M}_2\} = \text{vec}\{\mathbf{M}_1\}^{\text{T}}\text{vec}\{\mathbf{M}_2\} \tag{4.7}$$

for any arbitrary matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ of compatible dimensions, (4.5) can be written as

$$\text{NMSD}(n) = \frac{1}{V}\mathbf{b}^{\text{T}}\boldsymbol{\xi}(n), \tag{4.8}$$

where we have defined $\mathbf{b} \triangleq \text{vec}\{\mathbf{I}_V\}$.

Resuming our analysis, by subtracting both sides of (4.1) from $\mathbf{w}^{\text{o}}$, and replacing (2.1) and (2.7) in the resulting equation, after some algebraic manipulations, we can write

$$\widetilde{\boldsymbol{\psi}}_k(n) = [\mathbf{I}_M - \mu_k\zeta_k(n)\mathbf{u}_k(n)\mathbf{u}_k^{\text{T}}(n)]\widetilde{\mathbf{w}}_k(n-1) - \mu\zeta_k(n)\mathbf{u}_k(n)v_k(n), \tag{4.9}$$

where we have introduced

$$\widetilde{\boldsymbol{\psi}}_k(n) \triangleq \mathbf{w}^{\text{o}} - \boldsymbol{\psi}_k(n). \tag{4.10}$$

On the other hand, from (2.12b), we observe that

$$\widetilde{\mathbf{w}}_k(n) = \sum_{i\in\mathcal{N}_k} c_{ik}\widetilde{\boldsymbol{\psi}}_i(n). \tag{4.11}$$

We remark that, for the sake of simplicity, we are considering static combination weights. If we multiply both sides of (4.11) by $\widetilde{\mathbf{w}}_k^{\text{T}}(n)$ from the left, and use (2.12b) again, we obtain after some algebra

$$\|\widetilde{\mathbf{w}}_k(n)\|^2 = \sum_{i\in\mathcal{N}_k}\sum_{j\in\mathcal{N}_k} c_{ik}c_{jk}\widetilde{\boldsymbol{\psi}}_j^{\text{T}}(n)\widetilde{\boldsymbol{\psi}}_i(n). \tag{4.12}$$

Replacing (4.9) in (4.12), we obtain

$$\begin{aligned}\|\widetilde{\mathbf{w}}_k(n)\|^2 = \sum_{i\in\mathcal{N}_k}\sum_{j\in\mathcal{N}_k} c_{ik}c_{jk}&\Big\{[\mathbf{I}_M - \mu_j\zeta_j(n)\mathbf{u}_j(n)\mathbf{u}_j^{\text{T}}(n)]\widetilde{\mathbf{w}}_j(n-1) - \mu_j\zeta_j(n)\mathbf{u}_j(n)v_j(n)\Big\}^{\text{T}} \\ &\cdot\Big\{[\mathbf{I}_M - \mu_i\zeta_i(n)\mathbf{u}_i(n)\mathbf{u}_i^{\text{T}}(n)]\widetilde{\mathbf{w}}_i(n-1) - \mu_i\zeta_i(n)\mathbf{u}_i(n)v_i(n)\Big\}.\end{aligned} \tag{4.13}$$

To examine the MSD of node k, we need to take the expectations from both sides of (4.13).

At this point, we make a few assumptions to make the analysis more tractable:

A1. The weight error vectors $\tilde{\mathbf{w}}_i(n-1)$ are statistically independent of $\mathbf{u}_j(n)$ for any pair i and j. This is a multi-agent version of the independence theory, a common assumption in the adaptive filtering literature [125, 126];

A2. The measurement noise $v_k(n)$ is zero-mean with variance $\sigma^2_{v_k}$, i.i.d. across n, and independent from any other variable for $k = 1, \cdots, V$;

A3. The input signal at each node k is zero-mean and white Gaussian with variance $\sigma^2_{u_k}$. Hence, the autocorrelation matrix of $u_k(n)$ is given by $\mathbf{R}_{u_k} \triangleq \mathrm{E}\{\mathbf{u}_k(n)\mathbf{u}_k^{\mathrm{T}}(n)\} = \sigma^2_{u_k}\mathbf{I}$;

A4. For every node k, $\zeta_k(n)$ is independent from any other variable, and drawn from a Bernoulli distribution, such that $\zeta_k(n) = 1$ with probability $0 \leqslant p_{\zeta_k} \leqslant 1$ and $\zeta_k(n) = 0$ with probability $1-p_{\zeta_k}$. Furthermore, for any pair of distinct nodes, $\zeta_i(n)$ and $\zeta_j(n)$, $i \neq j$, are statistically independent from each other;

A5. At any time instant n, $u_i(n)$ is statistically independent from $u_j(n)$ for any pair of nodes i and j, $i \neq j$.

With these assumptions at hand, we can continue with our analysis. For the sake of brevity, we shall omit here the intermediate steps and focus on the main results obtained from (4.13). These results are justified in detail in Appendix D. For the scenario described, using (2.10), we can obtain

$$\xi_{kk}(n) = \sum_{i=1}^{V} c_{ik}^2 \tau_{ii}\xi_{ii}(n-1) + \sum_{j=1}^{V}\sum_{\substack{\ell=1\\ \ell\neq j}}^{V} c_{jk}c_{\ell k}\tau_{j\ell}\xi_{j\ell}(n-1) + M\sum_{q=1}^{V} c_{qk}^2\mu_q^2 p_{\zeta_q}\sigma^2_{u_q}\sigma^2_{v_q}, \tag{4.14}$$

where for the sake of compactness we have introduced

$$\tau_{j\ell} \triangleq \begin{cases} 1 - \mu_j p_{\zeta_j}\sigma^2_{u_j} - \mu_\ell p_{\zeta_\ell}\sigma^2_{u_\ell} - \mu_j\mu_\ell p_{\zeta_j}p_{\zeta_\ell}\sigma^2_{u_j}\sigma^2_{u_\ell} & \text{if } j \neq \ell \\ 1 - 2\mu_j p_{\zeta_j}\sigma^2_{u_j} + \mu_j^2 p_{\zeta_j}\sigma^4_{u_j}(M+2) & \text{otherwise.} \end{cases} \tag{4.15}$$

Hence, we can see that $\mathrm{MSD}_k(n)$ depends on the MSD of its neighbors at the previous iteration, as well as on the trace of the cross correlation matrices between $\tilde{\mathbf{w}}_j(n-1)$ and $\tilde{\mathbf{w}}_\ell(n-1)$, i.e., $\xi_{j\ell}(n-1) = \mathrm{E}\{\tilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\tilde{\mathbf{w}}_\ell(n-1)\}$, for every pair of nodes j and ℓ in the neighborhood of node k. The impact of each of these terms on the behavior of node k depends on the network topology, on the step sizes, on the sampling probabilities, and on the variance of the input signal in the nodes involved. Moreover, if $j = \ell$, the filter length M plays a role in the dynamic of the MSD as well. Lastly, the noise variance in the neighborhood of node k also impacts its MSD.

From (4.14), it becomes evident that we also need to study how the trace of the cross-correlation matrix of $\tilde{\mathbf{w}}_j(n)$ and $\tilde{\mathbf{w}}_\ell(n)$, with $j \neq \ell$, evolves over time. Again, we focus on the main result and leave the details for Appendix D. Using assumptions **A1** to **A5**, we can obtain the following recursion:

$$\xi_{j\ell}(n) = \sum_{t=1}^{V} c_{tj}c_{t\ell}\tau_{tt}\xi_{tt}(n-1) + \sum_{r=1}^{V}\sum_{\substack{s=1\\ s\neq r}}^{V} c_{rj}c_{s\ell}\tau_{rs}\xi_{rs}(n-1) + M\sum_{z=1}^{V} c_{zj}c_{z\ell}\mu_z^2 p_{\zeta_z}\sigma_{u_z}^2\sigma_{v_z}^2. \tag{4.16}$$

From (4.14) and (4.16), we observe that $\xi_{kk}(n)$ can be seen as a linear combination of the $\xi_{ii}(n-1)$ and $\xi_{j\ell}(n-1)$, plus a constant term that aggregates information from the step sizes, input signal power, and noise variance in the neighborhood of node k. An analogous procedure can be applied to each $\xi_{j\ell}(n)$ based on (4.16). Hence, we should be able to write

$$\boldsymbol{\xi}(n) = \boldsymbol{\Gamma}\boldsymbol{\xi}(n-1) + M\bar{\boldsymbol{\sigma}}, \tag{4.17}$$

where $\boldsymbol{\Gamma}$ is a matrix whose k-th row determines how exactly each $\xi_{ij}(n-1)$ influences the corresponding term in the current iteration, and $\bar{\boldsymbol{\sigma}}$ is a vector that aggregates the information from the network topology, step sizes, input signal variance, and noise power from the constant terms that appear in (4.14) and (4.16).

Let us now aggregate the combination weights into a $V \times V$ matrix $\mathbf{C}$, such that $[\mathbf{C}]_{ij} = c_{ij}$. Similarly, let us collect the information from the step sizes, sampling probabilities, input signal power, and noise variances from the last summations in the right-hand side (rhs) of (4.14) and (4.16) in a $V \times V$ diagonal matrix $\boldsymbol{\Pi}$, such that its k-th element is equal to $\mu_k^2 p_{\zeta_k}\sigma_{u_k}^2\sigma_{v_k}^2$, i.e.,

$$\boldsymbol{\Pi} = \begin{bmatrix} \mu_1^2 p_{\zeta_1}\sigma_{u_1}^2\sigma_{v_1}^2 & 0 & \cdots & 0 \\ 0 & \mu_2^2 p_{\zeta_2}\sigma_{u_2}^2\sigma_{v_2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \mu_V^2 p_{\zeta_V}\sigma_{u_V}^2\sigma_{v_V}^2 \end{bmatrix}. \tag{4.18}$$

In this case, we notice that $\mathbf{C\Pi C}^{\mathrm{T}}$ yields

$$\mathbf{C\Pi C}^{\mathrm{T}} = \begin{bmatrix} \sum_{k=1}^{V} c_{k1}^2 \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 & \sum_{k=1}^{V} c_{k1} c_{k2} \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 & \cdots & \sum_{k=1}^{V} c_{k1} c_{kV} \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 \\ \sum_{k=1}^{V} c_{k2} c_{k1} \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 & \sum_{k=1}^{V} c_{k2}^2 \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 & \cdots & \sum_{k=1}^{V} c_{k2} c_{kV} \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{V} c_{kV} c_{k1} \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 & \sum_{k=1}^{V} c_{kV} c_{k2} \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 & \cdots & \sum_{k=1}^{V} c_{kV}^2 \mu_k^2 p_{\zeta_k} \sigma_{u_k}^2 \sigma_{v_k}^2 \end{bmatrix}. \quad (4.19)$$

Consequently, we may write the $V^2 \times 1$ vector $\bar{\boldsymbol{\sigma}}$ in (4.17) as

$$\bar{\boldsymbol{\sigma}} = \mathrm{vec}\{\mathbf{C\Pi C}^{\mathrm{T}}\} = \begin{bmatrix} \sum_{i=1}^{V} c_{i1}^2 \mu_i^2 p_{\zeta_i} \sigma_{u_i}^2 \sigma_{v_i}^2 \\ \sum_{i=1}^{V} c_{i2} c_{i1} \mu_i^2 p_{\zeta_i} \sigma_{u_i}^2 \sigma_{v_i}^2 \\ \vdots \\ \sum_{i=1}^{V} c_{iV}^2 \mu_i^2 p_{\zeta_i} \sigma_{u_i}^2 \sigma_{v_i}^2 \end{bmatrix}. \quad (4.20)$$

As for the matrix $\boldsymbol{\Gamma}$, from (4.14) and (4.16) we obtain that

$$\boldsymbol{\Gamma} = \boldsymbol{\Omega} \odot \boldsymbol{\mathcal{C}}. \quad (4.21)$$

The matrix $\boldsymbol{\mathcal{C}}$ introduced in (4.21) is defined as

$$\boldsymbol{\mathcal{C}} \triangleq (\mathbf{C} \otimes \mathbf{C})^{\mathrm{T}}, \quad (4.22)$$

whereas $\boldsymbol{\Omega}$ is given by

$$\boldsymbol{\Omega} \triangleq \mathbf{1}_{V^2} \otimes \mathrm{vec}\{\mathbf{M}_\tau\}^{\mathrm{T}}, \quad (4.23)$$

where $\mathbf{M}_\tau$ is a $V \times V$ matrix such that $[\mathbf{M}_\tau]_{j\ell} = \tau_{j\ell}$. Hence, we can continue with the analysis of Eq. (4.17). Considering an initial condition $\boldsymbol{\xi}_0 = \boldsymbol{\xi}(0)$, the recursive application of (4.17) leads to

$$\boldsymbol{\xi}(n) = \boldsymbol{\Gamma}^n \boldsymbol{\xi}_0 + M \sum_{n_i=0}^{n-1} \boldsymbol{\Gamma}^{n_i} \bar{\boldsymbol{\sigma}}. \quad (4.24)$$

If the algorithm is initialized with $\mathbf{w}_k(0) = \mathbf{0}_M$ for every node k, we have that $\widetilde{\mathbf{w}}_k(0) = \mathbf{w}^{\mathrm{o}}$. Thus, for any i and j, we have that $\xi_{ij}(0) = \mathrm{E}\{\widetilde{\mathbf{w}}_i^{\mathrm{T}}(0)\widetilde{\mathbf{w}}_j(0)\} = \mathrm{E}\{\mathbf{w}^{\mathrm{o}^{\mathrm{T}}}\mathbf{w}^{\mathrm{o}}\} = \|\mathbf{w}^{\mathrm{o}}\|^2$, and, consequently,

$$\boldsymbol{\xi}_0 = \|\mathbf{w}^{\mathrm{o}}\|^2 \mathbf{1}_{V^2}. \quad (4.25)$$

Hence, replacing (4.25) in (4.24) and observing that $\sum_{n_i=0}^{n-1}\boldsymbol{\Gamma}^{n_i} = [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1}[\mathbf{I}_{V^2} - \boldsymbol{\Gamma}^n]$, we obtain

$$\boldsymbol{\xi}(n) = \|\mathbf{w}^{\mathrm{o}}\|^2\boldsymbol{\Gamma}^n\mathbf{1}_{V^2} + M[\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1}[\mathbf{I}_{V^2} - \boldsymbol{\Gamma}^n]\bar{\boldsymbol{\sigma}}. \tag{4.26}$$

Thus, considering (4.26) and (4.8), we can write

$$\boxed{\mathrm{NMSD}(n) = \frac{1}{V}\left\{\|\mathbf{w}^{\mathrm{o}}\|^2\mathbf{b}^{\mathrm{T}}\boldsymbol{\Gamma}^n\mathbf{1}_{V^2} + M\mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1}[\mathbf{I}_{V^2} - \boldsymbol{\Gamma}^n]\bar{\boldsymbol{\sigma}}\right\}.} \tag{4.27}$$

Assuming that

$$\rho(\boldsymbol{\Gamma}) < 1, \tag{4.28}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix, i.e., the maximum of the absolute values of its eigenvalues, we have that $\lim_{n\to\infty}\boldsymbol{\Gamma}^n = \mathbf{0}_{V^2\times V^2}$. In this case, we therefore observe from (4.27) that

$$\boxed{\mathrm{NMSD}(\infty) = \frac{M}{V}\mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1}\bar{\boldsymbol{\sigma}}.} \tag{4.29}$$

It is worth noting that although $\mathbf{w}^{\mathrm{o}}$ appears in (4.26) and (4.27), we do not need to know it beforehand. Instead, we only need to know its norm. This is usually not a problem, since the norm of the optimal system can be adjusted by using adaptive gain control. Furthermore, we remark that $\xi_{ij}(n) = \xi_{ji}(n)$, which means that the matrix $\boldsymbol{\Xi}$ defined in (4.4) is symmetric. Thus, the vector $\boldsymbol{\xi}(n)$ given by (4.2) has $V(V-1)/2$ duplicated entries. Although we could remove these elements from our model to make it more efficient from a computational perspective, and make the appropriate modifications where needed, we opted not to make this change for the clarity of the exposition. This is due to the fact that we are not primarily concerned with the computational complexity of the proposed model, and we believe that the formulation adopted is more convenient for the calculations. However, we would like to reinforce that this change is possible, and can reduce the amount of computations significantly, especially if V is large.

The models described by (4.27) and (4.29) can be applied to networks in which each node k has a distinct step size μ_k, input signal variance $\sigma_{u_k}^2$, and sampling probability p_ζ. However, it is very difficult to draw qualitative conclusions about the network performance from it. For this reason, in the remainder of this section we focus our attention on the particular case in which these parameters have the same value for every node in the network. As will become clear, this will give us some insights into the effects of sampling on the behavior of adaptive diffusion networks. Thus, we shall proceed with the analysis with an additional assumption:

A6. All the nodes in the network employ the same step size, i.e., $\mu_1 = \cdots = \mu_V = \mu > 0$ and are sampled with the same probability, i.e., $p_{\zeta_1} = p_{\zeta_2} = \cdots = p_{\zeta_V} = p_\zeta$. Moreover, the input signals have the same variance throughout the network, i.e., $\sigma^2_{u_1} = \sigma^2_{u_2} = \cdots = \sigma^2_{u_V} = \sigma^2_u$.

In this case, we notice from (4.15) that we may write

$$\tau_{kk} = \tau_a \triangleq 1 - 2\mu p_\zeta \sigma^2_u + \mu^2 p^2_\zeta \sigma^4_u \tag{4.30}$$

for $k = 1, \cdots, V$, and

$$\tau_{j\ell} = \tau_b \triangleq 1 - 2\mu p_\zeta \sigma^2_u + \mu^2 p_\zeta \sigma^4_u (M+2) \tag{4.31}$$

for $j \neq \ell$.

Moreover, in this case we notice from (4.18) that

$$\mathbf{\Pi} = \mu^2 p_\zeta \sigma^2_u \mathbf{R}_v, \tag{4.32}$$

where $\mathbf{R}_v$ is a diagonal matrix such that its k-th element is equal to $\sigma^2_{v_k}$, i.e.,

$$\mathbf{R}_v = \begin{bmatrix} \sigma^2_{v_1} & 0 & \cdots & 0 \\ 0 & \sigma^2_{v_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma^2_{v_V} \end{bmatrix}. \tag{4.33}$$

Consequently, we notice that $\bar{\boldsymbol{\sigma}}$ can be recast as

$$\bar{\boldsymbol{\sigma}} = \mu^2 p_\zeta \sigma^2_u \boldsymbol{\sigma}, \tag{4.34}$$

where we have introduced

$$\boldsymbol{\sigma} \triangleq \text{vec}\{\mathbf{C}\mathbf{R}_v\mathbf{C}^{\text{T}}\}. \tag{4.35}$$

Finally, we remark that in this case the matrix $\mathbf{\Omega}$ given by (4.23) can be written as

$$\mathbf{\Omega} = [\mathbf{\Omega}_1 \; \mathbf{\Omega}_2 \; \cdots \; \mathbf{\Omega}_V], \tag{4.36}$$

in which $\mathbf{\Omega}_i$ is a $V^2 \times V$ matrix whose elements in the i-th column are all equal to τ_b, and whose other elements are all equal to τ_a, i.e.

$$\boldsymbol{\Omega}_i = \underbrace{\begin{array}{c} \begin{array}{c} i\text{-th column} \\ \downarrow \end{array} \\ \begin{bmatrix} \tau_a & \cdots & \tau_a & \tau_b & \tau_a & \cdots & \tau_a \\ \tau_a & \cdots & \tau_a & \tau_b & \tau_a & \cdots & \tau_a \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \tau_a & \cdots & \tau_a & \tau_b & \tau_a & \cdots & \tau_a \end{bmatrix} \end{array}}_{V \text{ columns}}. \tag{4.37}$$

For illustration purposes, in Appendix E, we calculate the matrix $\boldsymbol{\Gamma}$ for a network with two nodes considering Assumption **A6**. Under these circumstances, (4.26) can be recast as

$$\boldsymbol{\xi}(n) = \|\mathbf{w}^{\mathrm{o}}\|^2 \boldsymbol{\Gamma}^n \mathbf{1}_{V^2} + \mu^2 p_\zeta M \sigma_u^2 [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1} [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}^n] \boldsymbol{\sigma}. \tag{4.38}$$

Thus, (4.27) and (4.29) can be rewritten as

$$\boxed{\mathrm{NMSD}(n) = \frac{1}{V}\left\{ \|\mathbf{w}^{\mathrm{o}}\|^2 \mathbf{b}^{\mathrm{T}} \boldsymbol{\Gamma}^n \mathbf{1}_{V^2} + \mu^2 p_\zeta M \sigma_u^2 \mathbf{b}^{\mathrm{T}} [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1} [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}^n] \boldsymbol{\sigma} \right\}} \tag{4.39}$$

and

$$\boxed{\mathrm{NMSD}(\infty) = \frac{\mu^2 p_\zeta M \sigma_u^2}{V} \mathbf{b}^{\mathrm{T}} [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1} \boldsymbol{\sigma},} \tag{4.40}$$

respectively.

Lastly, it is interesting to notice that we can obtain the theoretical MSD of the LMS algorithm [125, 126, 280] as a special case of (4.39). In this situation, we have that $V = 1$, and $\mathbf{b}$, $\boldsymbol{\sigma}$, $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\Omega}$ and $\boldsymbol{\Gamma}$ degenerate into $\mathbf{b} = 1$, $\boldsymbol{\sigma} = \sigma_v^2$, $\boldsymbol{\mathcal{C}} = 1$, $\boldsymbol{\Omega} = \tau_b$ and $\boldsymbol{\Gamma} = \tau_b$, respectively. Replacing these results in (4.39), we obtain

$$\mathrm{MSD}(n) = (\|\mathbf{w}^{\mathrm{o}}\|^2 - \chi_{\mathrm{LMS}}) \tau_b^n + \chi_{\mathrm{LMS}},$$

with χ_{LMS} given by

$$\chi_{\mathrm{LMS}} \triangleq \frac{\mu M \sigma_v^2}{2 - \mu \sigma_u^2 (M + 2)}. \tag{4.41}$$

In order to draw further insights on the behavior of adaptive diffusion networks and on the effects of sampling on their performance, we break our analysis down into two cases: one for the noncooperative strategy, which shall be analyzed in Sec. 4.1.1, and another one for the cooperative schemes. For the latter, we shall obtain an approximate model in Sec. 4.1.2, which will enable us to reach interesting qualitative conclusions.

4.1.1 The noncooperative Case

For the noncooperative case, we have $c_{kk} = 1$ for any $k = 1, \cdots, V$, and $c_{ik} = 0$ if $i \neq k$. Thus, we can replace $\boldsymbol{\mathcal{C}}$ by $\mathbf{I}_{V^2}$ in (4.21) and notice that, in this case, we obtain from (4.35) that $\boldsymbol{\sigma} = \text{vec}\{\mathbf{R}_v\}$. Alternatively, we may also see from (4.14) that in this case we can write

$$\xi_{kk}(n) = \tau_b \xi_{kk}(n-1) + \mu^2 p_\zeta M \sigma_u^2 \sigma_{v_k}^2. \tag{4.42}$$

Assuming $\xi_{kk}(0) = \|\mathbf{w}^{\text{o}}\|^2$, by recursively applying (4.42) we get

$$\xi_{kk}(n) = \tau_b^n \|\mathbf{w}^{\text{o}}\|^2 + \mu^2 p_\zeta M \sigma_u^2 \sigma_{v_k}^2 \sum_{n_i=0}^{n-1} \tau_b^{n_i}. \tag{4.43}$$

Assuming that $|\tau_b| < 1$, we have that τ_b^n fades to zero as $n \to \infty$. At this point, it is worth noting that

$$\tau_a = (1 - \mu p_\zeta \sigma_u^2)^2 \geqslant 0 \tag{4.44}$$

and that

$$\tau_b = \tau_a + \mu^2 p_\zeta \sigma_u^4 (M + 2 - p_\zeta) \geqslant \tau_a, \tag{4.45}$$

where the equality only occurs if $p_\zeta = 0$, in which case $\tau_b = \tau_a = 1$. Hence, assuming $p_\zeta > 0$, we notice that $\tau_b < 1$ if, and only if

$$0 < \mu < \frac{2}{(M+2)\sigma_u^2}, \tag{4.46}$$

where we have incorporated the fact that $\mu > 0$. In this case, we can write $\sum_{n_i=0}^{n-1} \tau_b^{n_i} = \frac{1-\tau_b^n}{1-\tau_b}$. Thus, considering (2.15) and (4.3), by applying some algebraic manipulations to (4.43), we can write the NMSD as

$$\boxed{\text{NMSD}_{\text{nc}}(n) = (\|\mathbf{w}^{\text{o}}\|^2 - \chi_{\text{nc}}) \tau_b^n + \chi_{\text{nc}},} \tag{4.47}$$

with

$$\chi_{\text{nc}} \triangleq \frac{\mu M}{2 - \mu \sigma_u^2 (M+2)} \cdot \frac{\sum_{k=1}^{V} \sigma_{v_k}^2}{V}. \tag{4.48}$$

Taking the limit of the NMSD for $n \to \infty$ in (4.47) yields

$$\boxed{\text{NMSD}_{\text{nc}}(\infty) = \chi_{\text{nc}}.} \tag{4.49}$$

Therefore, we can clearly see that χ_{nc} given by (4.48) represents the steady-state value of the NMSD for the noncooperative strategy. It is interesting to notice that p_ζ does not appear in (4.48). Thus, the sampling probability does not affect the steady-state NMSD of the algorithm in the noncooperative approach whatsoever, so long as $p_\zeta > 0$. If $p_\zeta = 0$ were chosen, we would obtain $\tau_b = 1$, and, from (4.42), we would get $\xi_k k(n) = \|\mathbf{w}^{\text{o}}\|^2$ for every iteration n. This is reasonable, since in this case the nodes would never acquire any information on the optimal system.

There are a couple of additional things to notice from the previous analysis. Firstly, we remark that (4.48) agrees with Eq. (2.18) for $p_\zeta = 1$, considering sufficiently small step sizes. Lastly, we remark that (4.46) is simply the condition for the stability of an LMS filter in the mean [125, 126]. Therefore, we can interpret (4.46) as follows: so long as $p_\zeta > 0$, if we pick a step size μ that leads to the individual stability of each filter in the network, the network as a whole will be stable, regardless of the value of p_ζ.

Finally, we notice that in (4.47) the term $(\|\mathbf{w}^{\text{o}}\|^2 - \chi_{\text{nc}})\tau_b^n$ decays exponentially along the iterations. Assuming that this term is dominant during the transient phase in comparison with χ_{nc}, we conclude that the closer τ_b is to unity, the slower the convergence rate. From (4.31), we can clearly see that

$$\lim_{p_\zeta \to 0^+} \tau_b = 1. \tag{4.50}$$

This indicates that the lower the sampling probability, the slower the convergence.

From the previous discussion, we can summarize the effects of sampling on the behavior of noncooperative networks as in the following result.

Result 1 (noncooperative networks): In the case of the noncooperative networks, the stability of dLMS is ensured if (4.46) holds and $0 < p_\zeta \leqslant 1$. Under these conditions, the lower the sampling probability p_ζ, the slower the convergence rate. Moreover, the steady-state NMSD is completely unaffected by p_ζ. This result follows as a direct consequence of Eqs. (4.47)–(4.49) and (4.50).

4.1.2 An Approximate Model for the Cooperative Strategies

Since the columns of the matrix $\mathbf{\Omega}$ are filled by either τ_a or τ_b, we could adopt $\mathbf{\Omega} \approx \tau_a \mathbf{1}_{V^2 \times V^2}$ or $\mathbf{\Omega} \approx \tau_b \mathbf{1}_{V^2 \times V^2}$ as an approximation. Making these replacements in (4.21) leads to $\mathbf{\Gamma} \approx \tau_a \mathcal{C}$ or $\mathbf{\Gamma} \approx \tau_b \mathcal{C}$, respectively. Due to (4.45), the second approximation tends to be more conservative. Replacing it in (4.28), and using the fact that for any real scalar α and matrix $\mathbf{M}$, $\rho(\alpha\mathbf{M}) = |\alpha|\rho(\mathbf{M})$, we conclude that

$$\tau_b \cdot \rho(\mathcal{C}) < 1, \tag{4.51}$$

where we used the fact that $\tau_b > 0$.

Due to (2.10), $\mathbf{C}$ is a left-stochastic matrix, i.e., a matrix whose entries are all non-negative, and whose columns add up to one. Consequently, $\mathbf{C} \otimes \mathbf{C}$ is also a left-stochastic matrix. By transposing it, we conclude that $\mathcal{C}$ is a right-stochastic matrix, i.e., all of its entries are non-negative, and its rows add up to one. One interesting property of such matrices is that their spectral radius is always equal to one [211, 281]. Thus, the condition established by (4.51) can be recast as simply $\tau_b < 1$. Replacing (4.31) in (4.51) and assuming that $p_\zeta > 0$, after some algebra we get (4.46). Hence, we can observe that our previous conclusion that if μ lies within a certain range, the sampling probability p_ζ does not affect the stability of the algorithm, so long as $p_\zeta > 0$, holds for the general case, and not just for the noncooperative approach. We remark that (4.46) corresponds to ensuring that each individual filter in the noncooperative scheme is stable. It is a well-known fact in the adaptive diffusion networks literature that, if every individual node is stable, the stability of the network as a whole in a cooperative scenario is also ensured [1–3]. However, we remark that (4.46) was obtained considering a worst-case scenario, in which $\mathbf{\Gamma} = \tau_b \mathcal{C}$. In practice, (4.46) is not strictly necessary to ensure the stability of the algorithm if a cooperative strategy is adopted. In these cases, greater step sizes may be employed without making the algorithm unstable. In this case, it will be shown in Sec. 4.1.4 that the sampling of the nodes may actually be beneficial to the stability. In other words, in the worst-case scenario, sampling does not hinder the stability of the algorithm, and, in general, it may improve it.

Furthermore, simulation results show that, when the nodes do cooperate, the approximation $\mathbf{\Gamma} \approx \tau_a \mathcal{C}$ leads to more accurate predictions than $\mathbf{\Gamma} \approx \tau_b \mathcal{C}$. Thus, adopting the first approximation for the cooperative strategies, we obtain from (4.38)

$$\mathrm{NMSD}(n) = \frac{1}{V}\left\{ \|\mathbf{w}^{\mathrm{o}}\|^2 \mathbf{b}^{\mathrm{T}} \tau_a^n \mathcal{C}^n \mathbf{1}_{V^2} + \mu^2 p_\zeta M \sigma_u^2 \mathbf{b}^{\mathrm{T}} [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1} [\mathbf{I}_{V^2} - \tau_a^n \mathcal{C}^n] \boldsymbol{\sigma} \right\}. \tag{4.52}$$

We remark that the result of the multiplication $\mathcal{C}^n \mathbf{1}_{V^2}$ is a column vector whose i-th element is the sum of the elements of the i-th row of the matrix $\mathcal{C}^n$. Since the product of right-stochastic matrices is also right-stochastic [211], $\mathcal{C}^n$ is right-stochastic, from which we conclude that $\mathcal{C}^n \mathbf{1}_{V^2} = \mathbf{1}_{V^2}$. Thus, (4.52) can be recast as

$$\mathrm{NMSD}(n) = \frac{1}{V}\left\{ \|\mathbf{w}^{\mathrm{o}}\|^2 \tau_a^n \mathbf{b}^{\mathrm{T}} \mathbf{1}_{V^2} + \mu^2 p_\zeta M \sigma_u^2 \mathbf{b}^{\mathrm{T}} [\mathbf{I}_{V^2} - \boldsymbol{\Gamma}]^{-1} [\mathbf{I}_{V^2} - \tau_a^n \mathcal{C}^n] \boldsymbol{\sigma} \right\}. \tag{4.53}$$

Using the fact that $\mathbf{b}^{\mathrm{T}} \mathbf{1}_{V^2} = V$, we thus conclude that for the cooperative strategies, the NMSD is well approximated by

$$\boxed{\mathrm{NMSD}_{\tau_a}(n) = \|\mathbf{w}^{\mathrm{o}}\|^2 \tau_a^n + \frac{\mu^2 p_\zeta M \sigma_u^2}{V} \cdot \mathbf{b}^{\mathrm{T}} [\mathbf{I}_{V^2} - \tau_a \mathcal{C}]^{-1} [\mathbf{I}_{V^2} - \tau_a^n \mathcal{C}^n] \boldsymbol{\sigma}.} \tag{4.54}$$

Analogously to what we observed in Sec. 4.1.1 about (4.47), the first term in (4.54) decays exponentially along the iterations with τ_a^n. Assuming once again that this term is dominant during the transient phase, we conclude that the closer that τ_a is to unity, the slower the convergence rate. From (4.31) and (4.30), we observe that

$$\lim_{p_\zeta \to 0^+} \tau_a = 1. \tag{4.55}$$

Hence, much like in the noncooperative case of Sec. 4.1.1, the lower the sampling probability, the slower the convergence for the cooperative strategies.

Finally, for the steady state, assuming that $\tau_a < 1$ and taking the limit when $n \to \infty$ in (4.54) yields

$$\boxed{\mathrm{NMSD}_{\tau_a}(\infty) = \frac{\mu^2 p_\zeta M \sigma_u^2}{V} \mathbf{b}^{\mathrm{T}} [\mathbf{I}_{V^2} - \tau_a \mathcal{C}]^{-1} \boldsymbol{\sigma}.} \tag{4.56}$$

Eq. (4.56) clearly depends on the matrix $\mathcal{C}$, and, therefore, on the network topology and combination rule adopted. It is not straightforward to extract conclusions from (4.56) and (4.49) for any arbitrary topology without calculating them explicitly. However, we can show that, if the matrix $\bar{\mathbf{C}}$ is symmetric, which certainly occurs if Metropolis weights are employed, for example, one may write

$$\mathrm{NMSD}_{\tau_a}(\infty) \leqslant \frac{\mu M}{2-\mu p_\zeta \sigma_u^2} \cdot \frac{\sum_{i=1}^{V^2} |\lambda_i(\bar{\boldsymbol{\Sigma}})|}{V}, \tag{4.57}$$

where $\lambda_i(\cdot)$ denotes the i-th greatest eigenvalue of a matrix and

$$\bar{\boldsymbol{\Sigma}} \triangleq \frac{1}{2}\left(\boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\mathrm{T}}\right), \tag{4.58}$$

with $\boldsymbol{\Sigma}$ defined as

$$\boldsymbol{\Sigma} \triangleq \boldsymbol{\sigma}\mathbf{b}^{\mathrm{T}}. \tag{4.59}$$

This result is derived in Appendix F. We notice from (4.57) that the upper bound for the steady-state NMSD decreases as we reduce p_ζ.

Although the steady-state NMSD for the cooperative strategies does depend on the network topology, it has been shown that its impact on the network performance is somewhat limited [63], so long as the graph that represents the network remains strongly connected, as defined in Sec. 2.2. With this in mind, let us calculate the steady-state NMSD for a particular network topology, which will facilitate the math and enable some qualitative conclusions that, as we shall see in Sec. 4.1.4, also hold for arbitrary networks. Thus, we next consider a network topology represented by a complete graph, i.e., one in which every pair of nodes is directly connected by an edge. A graph with this topology and V nodes is usually denoted by K_V in the literature. An example with $V = 8$ nodes is depicted in Fig. 48.

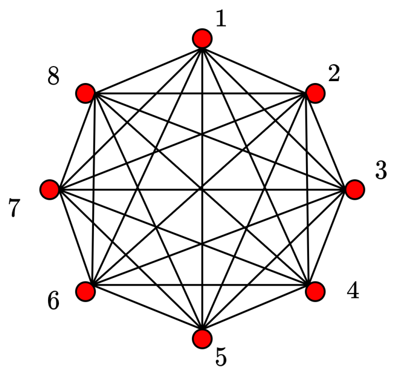


Figure 48: A network arranged according to the K_8 topology.

For the K_V topology, the Uniform and Metropolis weights coincide, and lead to $\mathbf{C}_{K_V} = \frac{1}{V}\mathbf{1}_{V\times V}$ and $\boldsymbol{\mathcal{C}}_{K_V} = \frac{1}{V^2}\mathbf{1}_{V^2\times V^2}$, where the index K_V is adopted to refer to this type of topology with

Uniform or Metropolis weights. For the aforementioned matrix $\mathcal{C}_{K_V}$, from (4.56), we get

$$\mathrm{NMSD}_{\tau_{K_V}}(\infty) = \chi_{K_V} \triangleq \frac{\mu M}{2 - \mu p_\zeta \sigma_u^2} \frac{\sum_{k=1}^{V} \sigma_{v_k}^2}{V^2}. \tag{4.60}$$

This result is derived in Appendix G. Comparing (4.48), (4.49), and (4.60), we observe that the sampling probability does not affect the steady-state performance of the noncooperative strategy, but does influence the steady-state NMSD of the cooperative schemes. From (4.60), we get

$$\frac{\mu M}{2} \frac{\sum_{k=1}^{V} \sigma_{v_k}^2}{V^2} < \chi_{K_V} \leqslant \frac{\mu M}{2 - \mu \sigma_u^2} \frac{\sum_{k=1}^{V} \sigma_{v_k}^2}{V^2}, \tag{4.61}$$

i.e., as we reduce the value of p_ζ, χ_{K_V} decreases as well. The first inequality is obtained by taking the limit of χ_{K_V} as $p_\zeta \to 0^+$. This observation is in accordance with the results from Figs. 17, 28, 29, 35, 37, and 41. This reduction in the steady-state NMSD should be more significant for relatively large values of μ and σ_u^2. If $\mu\sigma_u^2 \ll 2$, however, the impact of sampling becomes negligible. We remark that (4.48) and (4.60) agree with the analysis of [1–3] for the ATC dLMS algorithm with every node sampled and small step sizes, if we consider $p_\zeta = 1$ in the latter. Lastly, it is worth noting that (4.60) only holds for $p_\zeta > 0$. This is because, in order to obtain this result, we assumed in our calculations that $\tau_a < 1$, which can only be true if $p_\zeta > 0$, as can be seen from (4.30). If we consider $p_\zeta = 0$, we would get $\tau_a = 1$ and therefore obtain from (4.56) that $\mathrm{NMSD}_{\tau_a}(n) = \|\mathbf{w}^{\mathrm{o}}\|^2$ for every iteration n, which is in accordance with our expectations.

We can summarize the results of the analysis for the cooperative networks as in the following result.

Result 2 (cooperative networks): In the case of the cooperative networks, the stability of dLMS is ensured in the mean-squared sense if (4.28) holds, which can only occur if $p_\zeta > 0$. If the sampling probability is different from zero, (4.46) is a sufficient but not strictly necessary condition for stability of cooperative networks. Furthermore, if the algorithm is stable, then the lower the sampling probability p_ζ, the slower the convergence rate, much like in the noncooperative case. However, in contrast with the noncooperative approach, in cooperative schemes the steady-state NMSD decreases as we reduce p_ζ. This follows as a consequence of Eqs. (4.28) and (4.54)–(4.56), and is better visualized from the approximate model given by (4.60).

Comparing Results 1 and 2, therefore, we can see that in both the noncooperative and cooperative schemes, the convergence rate is deteriorated as we reduce p_ζ. However, it is only in the latter case that the steady-state NMSD decreases. One possible interpretation is as follows. The adaptation step is the process through which the algorithm acquires knowledge about its environment. For this reason, it is particularly relevant when there is little knowledge about the optimal system – e.g., during the transient phase. Thus, it makes sense that by not sampling the nodes in the transient phase, the convergence rate should deteriorate. However, the adaptation step also introduces noise into the algorithm, since it involves the acquisition of the desired signal, which is corrupted by it. In steady state, the algorithm does not gain enough information from the adaptation step to continue to improve its estimate of the optimal system, but is affected by the noise that it injects. The step size directly influences the impact of the measurement noise on the algorithms, since it multiplies $d_k(n)$, and, therefore, $v_k(n)$ in (4.1). Thus, the greater the μ, the more the noise will affect the behavior of the algorithm. Conversely, if μ is small, this effect is restricted. Following this line of reasoning, if we cease to sample some of the nodes, there is less noise entering the algorithm. In a noncooperative scheme, if a node is not sampled, its estimate remains fixed until its sampling is resumed. Hence, there is no reason why the steady-state performance should be affected by sampling less nodes, which is predicted by our theoretical model. However, in the cooperative schemes, this is changed by the existence of the combination step. Indeed, (4.60) shows that we should expect some decrease in the steady-state NMSD in this scenario, even if slightly. The theoretical model attributes this to the parameter τ_a, which is related to the cooperation between nodes in (4.14), since it determines how $\mathrm{E}\{\widetilde{\mathbf{w}}_\ell^{\mathrm{T}}(n-1)\widetilde{\mathbf{w}}_j(n-1)\}$, for ℓ and j in the neighborhood of node k, will affect $\mathrm{E}\{\|\widetilde{\mathbf{w}}_k(n)\|^2\}$. The model also shows that when μ is large, the effects of sampling less nodes on the steady-state NMSD should be more noticeable , which supports the idea of the step size as a factor that determines the impact of the measurement noise in the algorithm. Interestingly, the diagnosis that the adaptation step injects noise in the algorithm, and that the combination step tends to remove it, has been raised in, e.g., [81, 82, 282], but lacked formal theoretical support, until now.

Lastly, it may be interesting to compare the steady-state NMSD achieved by the algorithm with a certain step size μ and sampling probability p_ζ, and the one obtained by utilizing $\mu' = \mu p_\zeta$ and maintaining all nodes sampled. Denoting these quantities by $\mathrm{NMSD}_{\mathrm{async.}}(\infty)$ and $\mathrm{NMSD}_{\mathrm{sync.}}(\infty)$, respectively, we notice from (4.60) that, for the topology of Fig. 48, we get

$$\mathrm{NMSD}_{\mathrm{sync.}}(\infty) = p_\zeta \mathrm{NMSD}_{\mathrm{async.}}(\infty). \tag{4.62}$$

This is in accordance with [148], in which it was noticed that the steady-state NMSD of the synchronous networks should be less than or equal to that of the asynchronous one, if an adjusted step size is adopted taking p_ζ into consideration. In other words, if we adjust the step size, the network with all nodes sampled should outperform the one with random sampling, as it will present approximately the same convergence rate, with a lower steady-state NMSD. However, for a fixed step size, our conclusion that sampling less nodes improves the steady-state performance remains valid. This means that networks with random sampling are outperformed by synchronous ones. Thus, it is clear that the random sampling strategy is not, in itself, a recommendable approach. However, if one adapts the sampling of the nodes according to the estimation error, as the algorithms of Chapter 3 do, one may outperform the synchronous networks – as observed in Figs. 17, 28, 29, 35, 37, and 41 – while reducing the computational cost in steady state. As seen in Chapter 3, this comes at the expense of a slight increase in the computational burden in the transient phase.

4.1.3 Computational Cost Reduction

In this section, we seek to estimate the effects of random sampling on the expected computational cost of the dLMS algorithm. For brevity, we focus on the number of multiplications per iteration, but a similar analysis could be done for the additions.

We begin by noticing that each sampled node k needs to perform $M(2+|\mathcal{N}_k|)+1$ multiplications per iteration, assuming that static combination weights are employed. This is summarized in Table 7, in which the number of multiplications required at each node k per iteration is denoted by $\otimes_k$ and detailed for each calculation required. We can see that $2M + 1$ multiplications are related to the adaptation step. Assuming that none of the operations associated with this step have to be performed at node k if it is not sampled, we conclude that the total number of multiplications required at the iteration n and node k with random sampling can be estimated as

$$\otimes_k(n) = \zeta_k(n) \cdot (2M + 1) + M|\mathcal{N}_k|. \tag{4.63}$$

Taking the expectations from both sides in (4.63), we obtain

$$\mathrm{E}\{\otimes_k\} = p_\zeta(2M + 1) + M|\mathcal{N}_k|, \tag{4.64}$$

where we dropped the indication of the time instant n since the right-hand side of (4.64) remains constant along the iterations.

Table 7: Estimated number of multiplications per iteration at each sampled node k.

	Calculation	Step	$\otimes_k$
1	$y_k(n) = \mathbf{u}_k^{\mathrm{T}}\mathbf{w}_k(n-1)$	Adapt.	M
2	$e_k(n) = d_k(n) - y_k(n)$	Adapt.	0
3	$\mu \cdot e_k(n)$	Adapt.	1
4	$[\mu \cdot e_k(n)] \cdot \mathbf{u}_k(n)$	Adapt.	M
5	$\mathbf{w}_k(n-1)+\{[\mu \cdot e_k(n)] \cdot \mathbf{u}_k(n)\}$	Adapt.	0
6	$\sum_{i\in\mathcal{N}_k} c_{ik}\boldsymbol{\psi}_i(n)$	Comb.	$M\lvert\mathcal{N}_k\rvert$
Total			$M(2+\lvert\mathcal{N}_k\rvert)+1$

Summing (4.64) for $k = 1, \cdots, V$, we can estimate the expected computational cost for the whole network as

$$\mathrm{E}\{\otimes_{\text{total}}\} = Vp_\zeta(2M+1) + M\sum_{k=1}^{V}\lvert\mathcal{N}_k\rvert. \tag{4.65}$$

By replacing $p_\zeta = 1$ in (4.65), we obtain the number of multiplications required by the dLMS with every node sampled. Thus, denoting the number of multiplications saved per iteration due to the sampling by $\Delta\otimes_{\text{total}}$, we can thus estimate it as

$$\mathrm{E}\{\Delta\otimes_{\text{total}}\} = V(2M+1)(1-p_\zeta) \tag{4.66}$$

by subtracting (4.65) from the case with $p_\zeta = 1$.

From (4.66), we see that the smaller the p_ζ, the greater the savings in computation, as expected. Moreover, for a given p_ζ, the number of multiplications saved increases with V and M.

4.1.4 Simulation Results

In this section, we present simulation results to validate the theoretical analysis. They were obtained over an average of 1000 independent realizations, considering the scenarios summarized in Table 8. In every case, the coefficients of the optimal system $\mathbf{w}^{\mathrm{o}}$ are drawn from a Uniform distribution in the range $[-1,1]$, and later normalized so that $\mathbf{w}^{\mathrm{o}}$ has unit norm. Moreover, the length of the adaptive filter is always equal to that of $\mathbf{w}^{\mathrm{o}}$. We consider the network topology presented in Fig. 49(a), which was generated randomly. The input signal $u_k(n)$ and the measurement noise $v_k(n)$ follow Gaussian distributions with zero mean for each node k, with $\sigma_{u_k}^2 = \sigma_u^2 = 1$, whereas the noise variance $\sigma_{v_k}^2$ is drawn from a Uniform distribution in the range

$[0.001, 0.01]$ for $k = 1, \cdots, V$, as depicted in Fig. 49(b).

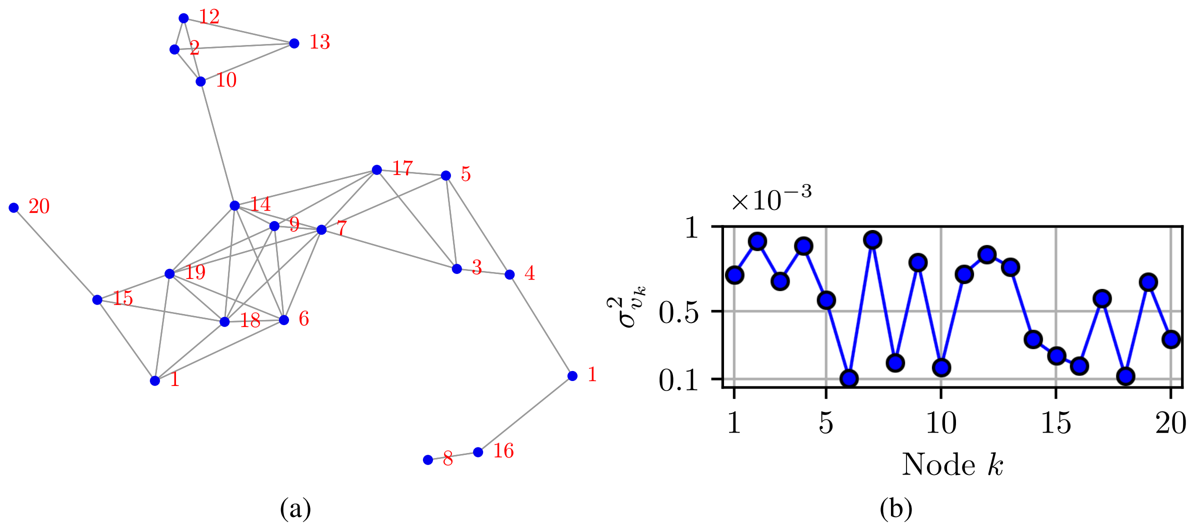


Figure 49: (a) Network topology, and (b) noise variance profile considered in the simulations.

Table 8: List of scenarios considered in the simulations.

Scenario	μ	M	Combination Rule
Scenario 1	0.1	10	Metropolis
Scenario 2	0.01	10	Metropolis
Scenario 3	0.02	100	Uniform
Scenario 4	0.01	10	noncooperative

This section is organized as follows. In Sec. 4.1.4.1, we study the transient performance of the algorithm, in Sec. 4.1.4.2, its stability, and in Sec. 4.1.4.3, its steady-state NMSD.

4.1.4.1 Transient Performance

In Fig. 50, we present the simulation results obtained in the Scenarios 1, 2, and 3 of Table 8, considering $p_\zeta \in \{1, 0.5, 0.1\}$, and compare them to the theoretical models of Sec. 3.1.2. The sub-figures in the top row present a comparison with the more precise model of Eq. (4.39), whereas those in the bottom row show the comparison with the approximate model of Eq. (4.54). Furthermore, each column of Fig. 50 refers to a scenario, with Figs. 50 (a) and (b) presenting the results obtained in Scenario 1, Figs. 50 (c) and (d) those from Scenario 2, and Figs. 50 (e) and (f) those from Scenario 3.

From Figs. 50(a), (c), and (e), we can see that the simulation results match the theoretical curves very well for all three scenarios, and that the model accurately predicts the improvement in steady-state NMSD caused by the sampling of less nodes. Furthermore, comparing the results of Figs. 50(c) and (e) with those of Fig. 50(a), we can observe that the difference in performance caused by the sampling is indeed more noticeable for larger step sizes, as expected. By comparing Figs. 50(a) and 50(b) one can see that, in Scenario 1, the approximate model is less accurate than the one described by Eq. (4.39). This was expected, to a certain extent. The same can be said about Scenario 3, by comparing Figs. 50(e) and (f). However, we observe from Figs. 50(c) and (d) that, in Scenario 2, both models practically coincide. Hence, we can conclude that the approximate model of Eq. (4.54) tends to be more accurate for relatively low step sizes μ and filter lengths M, and is more affected by them than the model of Eq. (4.39).

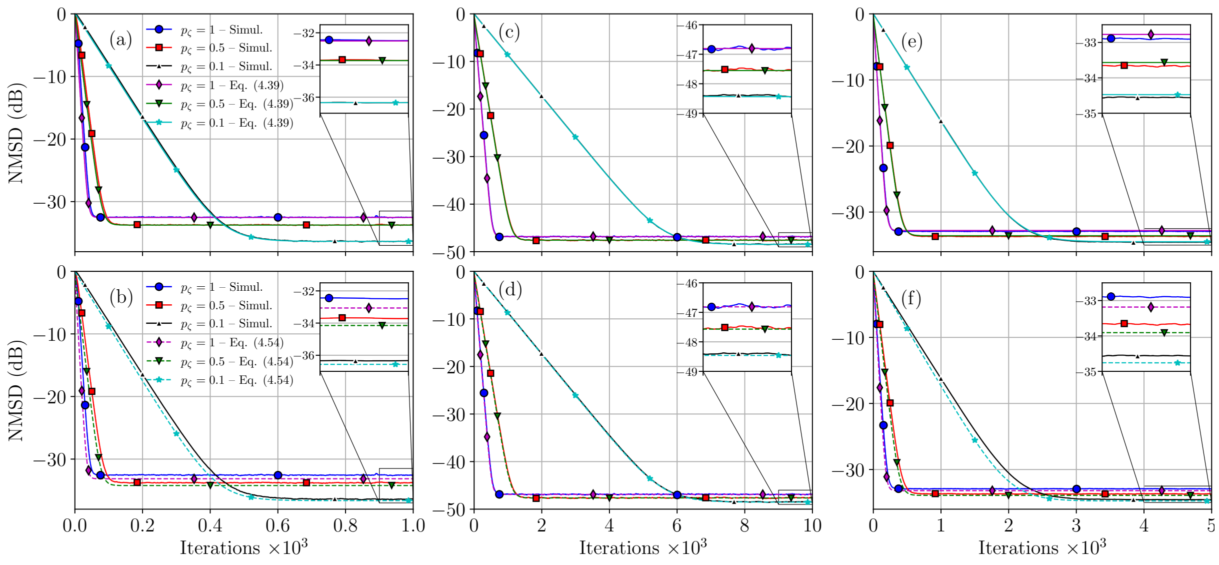


Figure 50: The sub-figures in the top row present a comparison between the simulation results and the model of Eq. (4.39), whereas the ones in the bottom row show a comparison with the model of Eq. (4.54). The simulation results were obtained considering the Scenario 1 of Table 8 in sub-figures (a) and (b), Scenario 2 in (c) and (d), and Scenario 3 in (e) and (f).

In order to examine the impact of sampling on the computational cost in this scenario, in Table 9 we present the average number of multiplications required per iteration in the whole network for each p_ζ considered in the simulations for $M = 10$, as in Scenarios 1 and 2, and for $M = 100$, as in Scenario 3. We also present the number of multiplications saved per iteration in comparison with the case in which every node is sampled, and compare them to the results given by (4.66). We can see that the average number of operations saved per iterations matches Eqs. (4.66), which can be attributed to the high number of realizations and iterations considered in the computations. Furthermore, it is straightforward to see that the smaller the p_ζ, the greater the savings in terms of computation, as expected. Moreover, for a fixed value of p_ζ, the computational cost and the savings increase with M, as expected. From these experiments, we can summarize the effects of sampling as follows: smaller sampling probabilities p_ζ lead to lower steady-state NMSD and computational costs, at the expense of a deteriorated convergence rate.

Table 9: Average number of multiplications per iteration in the network for $p_\zeta \in \{0.1, 0.5, 1\}$ with $M = 10$ and $M = 100$.

p_ζ	$\otimes_{\text{total}}$		$\Delta\otimes_{\text{total}}$		Eq. (4.66)	
	$M=10$	$M=100$	$M=10$	$M=100$	$M=10$	$M=100$
1	≈ 1460	14420	0	0	0	0
0.5	1250	12410	210	2010	210	2010
0.1	1082	10802	378	3618	378	3618

In the simulations of Fig. 51, we consider the Scenario 4 of Table 8, and compare the simulation results to the theoretical model given by Eq. (4.47) for the noncooperative scheme. Once again, the simulation results closely match the theoretical analysis. Furthermore, the simulations support the idea that the sampling probability does not affect the steady-state performance of the algorithm in the noncooperative scheme, unlike what was observed in Fig. 50 for the cooperative rules.

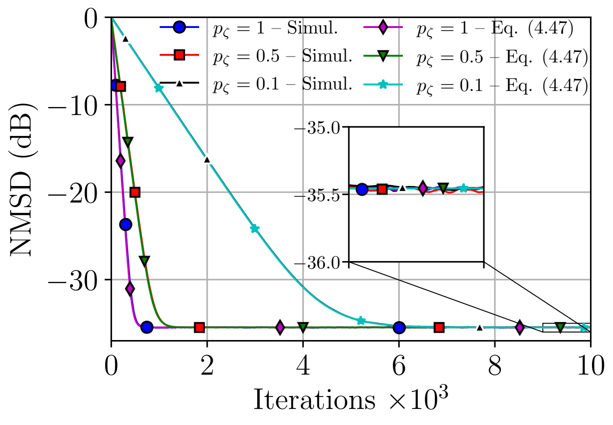


Figure 51: Comparison between the simulation results and the model of Eq. (4.47) for $p_\zeta \in \{1, 0.5, 0.1\}$ in Scenario 4.

4.1.4.2 Effects of Sampling on the Stability

From (4.46), we concluded that the sampling probability does not affect the stability of the algorithm so long as μ is sufficiently small and $p_\zeta > 0$. However, (4.46) is not strictly necessary to ensure the stability of the algorithm in the mean-squared sense. For instance, the value of μ considered in the Scenario 3 of Table 8 does not satisfy (4.46), but still leads to the stability of the dLMS algorithm for the network of Fig. 4 with Uniform weights and $M = 100$. Under these conditions, one obtains $\rho(\mathbf{\Gamma}) \approx 0.9628$ for $p_\zeta = 1$, which satisfies (4.28) and thus ensures the convergence in the mean-squared sense. For $p_\zeta = 0.5$ and $p_\zeta = 0.1$, we get $\rho(\mathbf{\Gamma}) < 1$ as well.

To verify if the sampling of the nodes influences the stability of the algorithm in the general case, we calculated the spectral radius of the matrix $\mathbf{\Gamma}$ considering $M = 100$ and the three combination policies for the network of Fig. 4 with $\mu = 0.1$ and several values of p_ζ. The results are shown in Fig. 52 (a), where we have highlighted with a red horizontal line the threshold $\rho(\mathbf{\Gamma}) = 1$. We can see that, for all combination policies, the adoption of $p_\zeta = 0$ leads to $\rho(\mathbf{\Gamma}) = 1$. This is expected, since in this case we get $\tau_b = \tau_a = 1$, and consequently $\mathbf{\Gamma} = \boldsymbol{\mathcal{C}}$, whose spectral radius is equal to one. Intuitively, this comes from the fact that the algorithm never acquires any knowledge on the optimal system if the nodes are never sampled. For the noncooperative strategy, $\rho(\mathbf{\Gamma})$ increases with p_ζ, indicating that the algorithm is unstable for any sampling probability. For the Uniform and Metropolis combination policies, however, $\rho(\mathbf{\Gamma})$ decreases up to a certain point with the increase of p_ζ, and then begins to rise. Interestingly, for both policies, Fig. 52 (a) tells us that, under the conditions considered, the algorithm is unstable with all nodes sampled, but we can stabilize it by sampling less nodes. For the Uniform rule, we conclude from Fig. 52 (a) that the dLMS algorithm is stable for $p_\zeta \in]0, 0.71]$ (approximately), whereas for the Metropolis rule the stability occurs for $p_\zeta \in]0, 0.39]$. In order to verify these results, we ran the dLMS algorithm under the same circumstances considered in Fig. 52 (a) with different sampling probabilities in the range $[0.01, 1]$ for $200 \cdot 10^3$ iterations, which is more than necessary for the algorithm to achieve the steady state with $p_\zeta = 0.01$ and cooperative strategies. Then, utilizing the `isnan` and `isinf` functions of MATLAB®, we calculated the percentage of realizations in which the dLMS algorithm diverged at some iteration. The results are depicted in Fig. 52 (b). We can see that, for the noncooperative strategy, the algorithm diverges in 100% of the realizations for all values of p_ζ considered. For the Uniform and Metropolis rules, the percentage of realizations in which the algorithm diverges is initially zero, and increases steeply as p_ζ approaches the limit values of $p_\zeta = 0.71$ and $p_\zeta = 0.39$, respectively. For the former combination policy, the algorithm starts to diverge for $p_\zeta > 0.68$, whereas for the latter the first divergences occur for $p_\zeta > 0.41$. In both cases, by increasing p_ζ slightly further, the algorithm begins to diverge at some point in 100% of the realizations. Therefore, the simulation results of Fig. 52 (b) support the theoretical findings of Fig. 52 (a). It is worth noting that, although the Uniform rule leads to the stability of the algorithm for a wider range of p_ζ than the Metropolis rule in this case, this does not necessarily occur in all scenarios. For example, for the topology in Fig. 48, the weights coincide for the two rules and therefore there is no difference between them in terms of the stability of the algorithm.

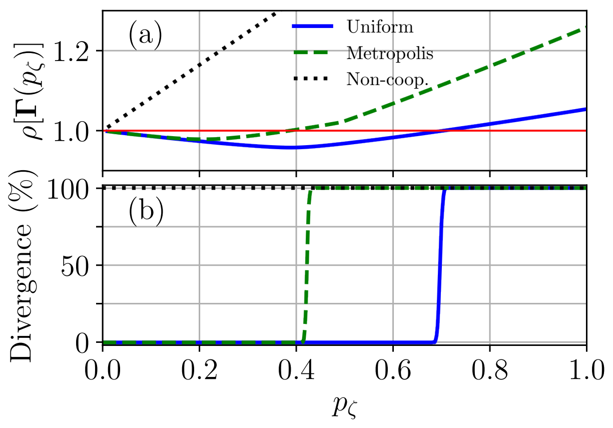


Figure 52: (a) $\rho(\mathbf{\Gamma})$ as a function of p_ζ, and (b) percentage of realizations in which the dLMS diverged with $\mu = 0.1$ and $M = 100$ for $p_\zeta \in [0.01, 1]$ with different combination policies.

4.1.4.3 Steady-State Performance

Lastly, in order to verify Eqs. (4.40) and (4.56) in detail, we ran the ATC dLMS for different values of $p_\zeta \in [0.1, 1]$, and calculated the average NMSD during the final 20% iterations of each realization. The results are shown in Figs. 53(a), (b), and (c) for Scenarios 1, 2, and 3, respectively. In each case, we set the total number of iterations N so that the algorithms achieved the steady state before the end of each realization, resulting in $10^3 \leqslant N \leqslant 10^5$. In all scenarios, we observe that the steady-state NMSD drops continuously as we decrease p_ζ. Moreover, the simulation results match (4.40) very closely in Scenarios 1 and 2. In Scenario 3, there is a discrepancy between the simulation results and the theoretical curve of 0.1 dB, on average. As for the model of Eq. (4.56), there is a difference of approximately 0.40 dB in comparison with the simulation results in Fig. 53(a), on average. In Fig. 53(c), this difference is of roughly 0.23 dB, whereas in Fig. 53(b) the curve practically overlaps with the simulation results.

4.2 A Simplified Model for the NMSD of DTRAS-dLMS

Based on the analysis presented in Sec. 4.1, in this section we derive a simplified model for the performance of the DTAS-dNLMS and DTRAS-dNLMS algorithm in a stationary environment. We assume that, in this case, the reset mechanism of DTRAS-dNLMS is never activated

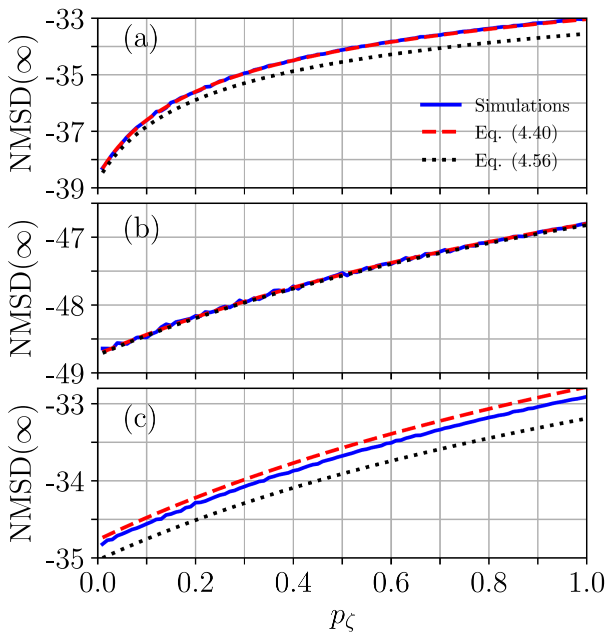


Figure 53: Steady-state NMSD for $p_\zeta \in [0.01, 1]$ in: (a) Scenario 1, (b) Scenario 2, and (c) Scenario 3.

due to the absence of changes in the environment, and thus neglect its effects on the sampling of the nodes. For simplicity, we maintain our Assumption **A6** in our analysis.

In this model, we consider that the sampling probability can vary over time, but is common to every node in the network, i.e., $p_{\zeta_1}(n) = p_{\zeta_2}(n) = \cdots = p_{\zeta_V}(n) = p_\zeta(n)$. Moreover, we consider that $p_\zeta(n)$ is equal to one in the transient phase, but switches to $\widehat{p}_{\max}$ given by (3.52) after a certain number of iterations, which we denote by n_{switch}. This is a conservative approach, but leads to a good match between the theoretical results thus obtained and the steady-state performance, as we shall see next. Hence, we have that

$$p_\zeta(n) = \begin{cases} 1, & \text{if } n < n_{\text{switch}}, \\ p_{\text{s.s.}_{\max}}, & \text{otherwise.} \end{cases} \tag{4.67}$$

In a situation in which p_ζ varies over time, the parameters τ_b and τ_a do as well. In this case, we can see from (4.30) and (4.31) that they are given by

$$\tau_a(n) = 1 - 2\mu p_\zeta(n)\sigma_u^2 + \mu^2 p_\zeta^2(n)\sigma_u^4 \tag{4.68}$$

and

$$\tau_b(n) = 1 - 2\mu p_\zeta(n)\sigma_u^2 + \mu^2 p_\zeta(n)\sigma_u^4(M+2). \tag{4.69}$$

Consequently, we have that $\mathbf{\Gamma}(n)$ varies over time according to

$$\mathbf{\Gamma}(n) = \mathbf{\Omega}(n) \odot \boldsymbol{\mathcal{C}}, \tag{4.70}$$

with

$$\mathbf{\Omega}(n) = [\mathbf{\Omega}_1(n)\ \mathbf{\Omega}_2(n)\ \cdots\ \mathbf{\Omega}_V(n)], \tag{4.71}$$

in which $\mathbf{\Omega}_i(n)$ is given by

$$\mathbf{\Omega}_i(n) = \underbrace{\begin{bmatrix} \tau_a(n) & \cdots & \tau_a(n) & \overset{\overset{i\text{-th column}}{\downarrow}}{\tau_b(n)} & \tau_a(n) & \cdots & \tau_a(n) \\ \tau_a(n) & \cdots & \tau_a(n) & \tau_b(n) & \tau_a(n) & \cdots & \tau_a(n) \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \tau_a(n) & \cdots & \tau_a(n) & \tau_b(n) & \tau_a(n) & \cdots & \tau_a(n) \end{bmatrix}}_{V \text{ columns}}. \tag{4.72}$$

Thus, (4.38) must be recast as

$$\boldsymbol{\xi}(n) = \mathbf{\Gamma}(n)\boldsymbol{\xi}(n-1) + \mu^2 p_\zeta(n) M\sigma_u^2 \boldsymbol{\sigma}. \tag{4.73}$$

By considering $p_\zeta(n)$ given by (4.67) in (4.31), (4.30), and (4.73), we can predict the NMSD performance of the DTRAS-dLMS algorithm. The only question left is at which iteration n_{switch} the sampling probability transitions from unity to its steady-state value. To answer this, we must investigate when the algorithm achieves the steady state in terms of the NMSD. From Assumptions **A1–A5**, we can write that $\text{MSE}_k(n) \approx \sigma_u^2 \text{MSD}_k(n-1) + \sigma_{v_k}^2$, and, consequently,

$$\text{NMSE}(n) \approx \sigma_u^2 \text{NMSD}(n-1) + \bar{\sigma}_v^2, \tag{4.74}$$

where we have introduced $\bar{\sigma}_v^2 \triangleq \frac{\sum_{k=1}^V \sigma_{v_k}^2}{V}$. During the transient phase, we have $p_\zeta(n) = 1$ and therefore, $\tau_a(n) = \tau_{a_0} \triangleq 1 - 2\mu\sigma_u^2 + \mu^2\sigma_u^4$. Moreover, if we adopt the approximation $\mathbf{\Omega}_i(n) \approx \tau_{a_0}\mathbf{1}_{V^2\times V}$ while the nodes are still sampled, we have that $\mathbf{\Gamma}(n) \approx \mathbf{\Gamma}_0 \triangleq \tau_{a_0}\boldsymbol{\mathcal{C}}$ during this period. Thus, using (4.8) and (4.73) with the previous approximations, and considering that

$\xi(0) = \|\mathbf{w}^{\mathrm{o}}\|^2$, we conclude that, while the nodes are still sampled, we may write

$$\mathrm{NMSD}(n) \approx \frac{\|\mathbf{w}^{\mathrm{o}}\|^2 \tau_{a_0}^n}{V} \cdot \mathbf{b}^{\mathrm{T}} \boldsymbol{\mathcal{C}}^n \mathbf{1}_{V^2} + \frac{\mu^2 M \sigma_u^2}{V} \mathbf{b}^{\mathrm{T}} [\mathbf{I}_{V^2} - \tau_{a_0} \boldsymbol{\mathcal{C}}]^{-1} [\mathbf{I}_{V^2} - \tau_{a_0}^n \boldsymbol{\mathcal{C}}^n] \boldsymbol{\sigma}. \tag{4.75}$$

From (4.75), we see that the NMSD depends on the network topology due to the matrix $\boldsymbol{\mathcal{C}}$. For simplicity, as an approximation, we consider instead that the network has a K_V topology such as that of Fig. 48. Using the results from Appendix G, in this case we obtain $\mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \tau_{a_0} \boldsymbol{\mathcal{C}}_C]^{-1} \boldsymbol{\sigma} = \frac{1}{1-\tau_{a_0}} \bar{\sigma}_v^2$. Thus, from (4.75) and (4.74) we can write

$$\mathrm{NMSE}(n) \approx \sigma_u^2 \|\mathbf{w}^{\mathrm{o}}\|^2 \tau_{a_0}^{n-1} + \left[\frac{\mu M \sigma_u^2}{(2 - \mu \sigma_u^2) V} - \frac{\mu^2 M \sigma_u^4 \tau_{a_0}^{n-1}}{V} + 1 \right] \bar{\sigma}_v^2. \tag{4.76}$$

From (4.76), we notice that, while the nodes are sampled, and assuming $\tau_{a_0} < 1$, the NMSE converges approximately to

$$\chi_{\mathrm{NMSE}} = \left[\frac{\mu M \sigma_u^2}{(2 - \mu \sigma_u^2) V} + 1 \right] \bar{\sigma}_v^2. \tag{4.77}$$

Next, we shall consider that the algorithm has achieved the steady state in terms of the NMSE at the time instant n if $\mathrm{NMSE}(n) \leqslant (1 + \delta_{\mathrm{threshold}}) \chi_{\mathrm{NEMSE}}$, where $0 < \delta_{\mathrm{threshold}} \ll 1$ is a constant. From (4.76) and (4.77), after some algebra, we conclude that this holds for

$$n \geqslant n_{\mathrm{s.s.}} = 1 + \left\lceil \frac{\ln(\delta_{\mathrm{threshold}}) + \ln(\chi_{\mathrm{NEMSE}}) - \ln\left[\sigma_u^2 \left(\|\mathbf{w}^{\mathrm{o}}\|^2 - \frac{\mu^2 M \sigma_u^2 \bar{\sigma}_v^2}{V} \right) \right]}{\ln(\tau_{a_0})} \right\rceil. \tag{4.78}$$

Finally, the iteration at which the sampling probability transitions in (4.67) can be approximated by

$$n_{\mathrm{switch}} \approx n_{\mathrm{s.s.}} + \Delta n. \tag{4.79}$$

4.2.1 Simulations

The results presented next were obtained from an ensemble average of 100 independent realizations. In each experiment, we consider the network topology presented in Fig. 54(a), which was generated randomly. The input signal $u_k(n)$ and the measurement noise $v_k(n)$ follow Gaussian distributions with zero mean for each node k, with $\sigma_{u_k}^2 = \sigma_u^2 = 1$, whereas the noise variance $\sigma_{v_k}^2$ is drawn from a Uniform distribution in the range [0.001,0.01] for $k = 1, \cdots, V$, as shown in Fig. 54(b). We consider $M = 10$ for both the optimal system and the diffusion algorithm, and adopt $\mu = 0.1$. The coefficients of the optimal system $\mathbf{w}^{\text{o}}$ are drawn from a Uniform distribution in the range $[-1, 1]$, and are later normalized so as to obtain $\|\mathbf{w}^{\text{o}}\|^2 = 1$. We adopt Metropolis combination weights. For the theoretical model, we consider $\delta_{\text{threshold}} = 0.01$ in Eq. (4.78).

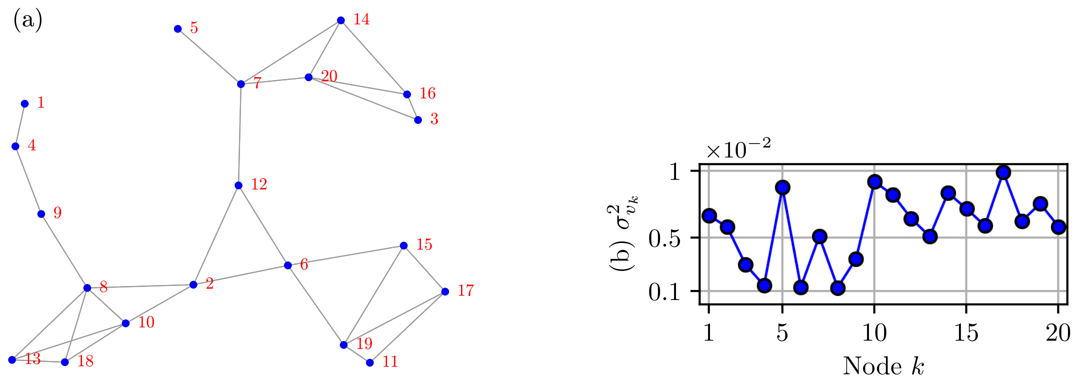


Figure 54: (a) Network topology, and (b) noise variance profile considered in the simulations.

In Fig. 55, we show the theoretical curves as well as the simulation results obtained with the DTRAS-dLMS algorithm with $\gamma = 25$ and $\Delta n = 200$. These parameters were chosen so as to obtain a significant reduction in the sampling probability in steady state and a good transient performance. It is worth noting that in this case (3.65) yields $p_{\text{s.s.}_{\max}} = 0.04$. For reference, we also show the results for the dLMS algorithm with fixed sampling probabilities $p_\zeta = 1$ and $p_\zeta = p_{\text{s.s.}_{\max}}$. In Fig. 55(a) we show the NMSD curves, and in Fig. 55(b) the sampling probability along the iterations. We notice that, initially, the DTRAS-dLMS algorithm maintains every node sampled, and consequently presents the same convergence rate as the dLMS algorithm with $p_\zeta = 1$, which is captured by our theoretical model. Then, after $n_{\text{switch}} = 146$ iterations, the sampling probability of the DTRAS-dLMS suddenly decreases. We observe that the approximation given by Eq. (4.67) is reasonable in this case, and we notice that the NMSD of DTRAS-dLMS begins to decrease until it stabilizes at a steady-state level approximately 6 dB lower than that of the dLMS algorithm with every node sampled. This is also predicted by our theoretical model. Overall, we observe that the theoretical curves match the simulation results well. Lastly, we observe that the dLMS algorithm with a fixed sampling probability of $p_\zeta = 0.04$ converges approximately to the same level of steady-state NMSD as the DTRAS-dLMS algorithm, but at a much slower convergence rate. Simulation results obtained with other values of M and μ led to similar conclusions, but are omitted here due to space restrictions.

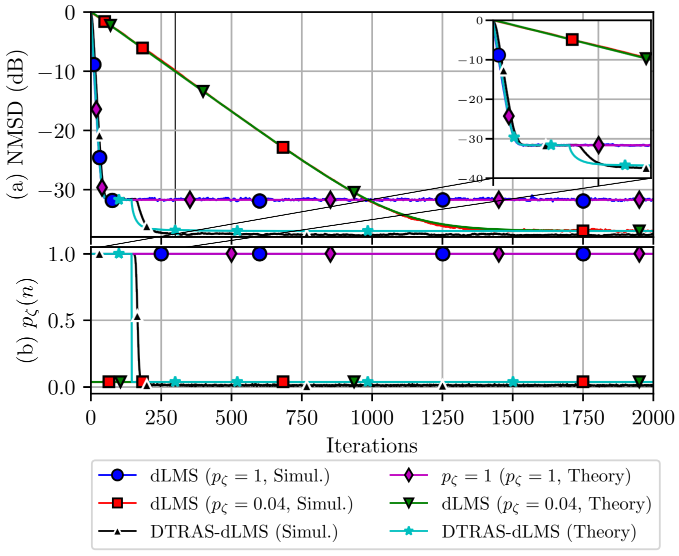


Figure 55: Comparison between the theoretical models and the simulation results with $\gamma = 25$ and $\Delta n = 100$. (a) NMSD curves, and (b) sampling probability along the iterations.

Lastly, in Fig. 56, we repeat the previous experiment, but considering $\Delta n = 1000$ for the DTRAS-dLMS algorithm. In this case, the theoretical model underestimates n_{switch}, and, as a result, there is a noticeable mismatch between the simulation results and the theoretical NMSD curve between $n \approx 2000$ and $n \approx 2800$. Overall, the simulations suggest that the theoretical model for n_{switch} is fairly accurate for relatively small values of Δn, but can lead to poor estimates if Δn is very large. Nonetheless, we remark that, typically, it is not desirable to choose excessively large values for Δn, since in this case it takes more iterations for the algorithm to cease the sampling of the nodes. Thus, the situation depicted in Fig. 56 is not of practical interest. Nevertheless, in future works we intend to improve our estimate of n_{switch}. Despite this, it is interesting to note that the theoretical model still predicts the steady-state NMSD accurately in this scenario.

4.3 Conclusions

In this chapter, we analyzed the effects of sampling on the network performance. In Sec. 4.1, we focused our attention on the case in which the nodes are sampled at random, whereas in Sec. 4.2 we derived a simplified model for the NMSD of the DTRAS-dLMS algorithm. In the former case, the analysis shows that, as we reduce the sampling probability, the convergence rate is severely affected, but the steady-state NMSD slightly decreases, which is in accordance with the phenomena observed in the simulations of Chapter 3. Interestingly, we also conclude that the sampling of less nodes may render the algorithm stable in situations in which it would

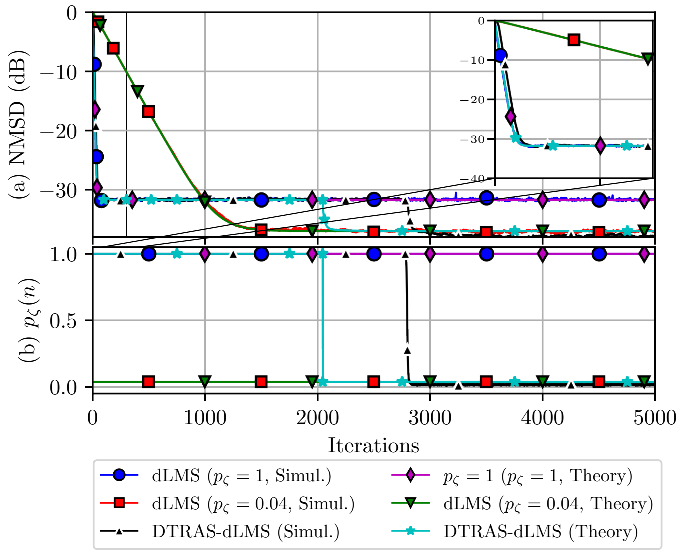


Figure 56: Comparison between the theoretical models and the simulation results with $\gamma = 25$ and $\Delta n = 1000$. (a) NMSD curves, and (b) sampling probability along the iterations.

be unstable if every node remained sampled. However, when the nodes are sampled randomly, the network performance is degraded in comparison with the asynchronous network if the step sizes are adjusted, which indicates that the random sampling technique is not the most adequate approach, as expected. Nevertheless, the model derived in Sec. 4.2 shows that, by keeping the nodes sampled in the transient phase, and ceasing to sample them otherwise, the DTAS-dLMS algorithm can achieve a slightly improved steady-state performance in comparison with the case in which every node is permanently sampled, while preserving its convergence rate. Thus, the analysis shows that, by managing the sampling of the nodes in an intelligent manner, it is possible for adaptive diffusion networks to perform better with less, rather than more, data, as we can attest from the simulations of Chapter 3.

5 CONCLUSIONS

Due to their advantages in comparison with centralized strategies and other distributed approaches, adaptive diffusion networks have consolidated themselves in the literature as an interesting tool for distributed signal processing [1–6]. Moreover, with the recent advances in areas such as IoT and 5G networks, we may expect to see an increased applicability of adaptive diffusion networks in the near future [99, 100].

Due to their success, several branches of research have emerged from adaptive diffusion networks. Examples include, e.g., kernel-based [32–36] and multitask adaptive diffusion networks [20–29], as well as adaptive diffusion networks for GSP [39, 41, 42, 47, 48].

However, it is oftentimes desirable to restrict the amount of data measured, processed, and transmitted in these networks. Such restrictions may affect the network performance, but are typically important for their feasibility in practice. One way to limit the number of measurements and the computational cost in adaptive diffusion networks is through the adoption of sampling, whereas several techniques have been proposed to reduce the amount of data transmitted. Some of these solutions are known as censoring techniques in the literature, since they seek to prevent transmissions between a node and all of its neighbors [72–82].

In Chapter 2, we carried out an extensive literature review on the various forms of adaptive diffusion networks, highlighting the types of situations that each approach seeks to address. Moreover, we showed simulation results considering synthetic and real-world data, illustrating the potential, as well as the challenges, associated with the usage of each type of network.

In Chapter 3, we presented several algorithms for the adaptive sampling and censoring of diffusion networks. In Sec. 3.1, we presented the AS-dNLMS algorithm. Its goal is to keep the nodes censored when the error is high in magnitude, and cease to censor them otherwise. Thus, the sampling of the nodes is maintained, e.g., during the transient phase, which preserves the convergence rate of the original dNLMS algorithm with all nodes sampled. Moreover, with only slight modifications it can also be used as a censoring technique, resulting in the ASC-dNLMS algorithm. Simulation results showed the good behavior of the AS-dNLMS and ASC-dNLMS solutions in a wide range of situations, including simulations with real-world data in a GSP setting.

However, the AS-dNLMS does have weaknesses. One important issue resides in the fact that the proper selection of its parameter depends on prior knowledge of the noise power. At the very least, the filter designer must know the maximum noise variance in the network, but this may lead to problematic situations if one of the nodes is much noisier than the others, as was shown in Sec. 3.2. For this reason, the DTAS-dNLMS algorithm was introduced in Sec. 3.2. By allowing each node k to select its own time-varying parameter $\beta_k(n)$ and by incorporating an online estimation of $\sigma^2_{v_k}$, the DTAS-dNLMS directly addresses the first weakness of the AS-dNLMS algorithm. However, its other main weakness is not solved by the DTAS-dNLMS version: its comparatively poor tracking capability. For this reason, we also proposed in Sec. 3.2 the DTRAS-dNLMS algorithm, which incorporates a reset mechanism for the sampling of the nodes. Thus, if changes in the environment are detected, the algorithm resumes the sampling of the nodes, which significantly improves its tracking capability in comparison with the previous versions of the algorithm.

Moreover, in Sec. 3.3, we extended the DTAS-dNLMS algorithm of Sec. 3.2 to RFF kernel-based networks, and also employed it as a censoring technique. Finally, in Sec. 3.4, we also extended the AS-dNLMS algorithm to multitask networks. Since the AS-dNLMS, DTAS-dNLMS and DTRAS-dNLMS algorithms can be straightforwardly employed in GSP settings, as was done in Sec. 3.1.6.5, we remark that we have proposed a sampling and/or censoring solution for every type of network covered in Chapter 2.

Interestingly, we noticed in many of the simulations of Chapter 3 that, under many circumstances, the sampling of less nodes led to a slightly lower steady-state NMSD in comparison with the case in which every node is permanently sampled. This motivated us to analyze the effects of sampling on network performance in Chapter 4. We began by investigating the effects of random node sampling on these networks in Sec. 4.1. Our analysis shows that, indeed, a reduction in the steady-state NMSD is expected as we decrease the probability p_ζ of the nodes being sampled. On the other hand, the convergence rate deteriorates as we reduce p_ζ. If adjusted step sizes are adopted, a network with random node sampled is outperformed by its counterpart with every node permanently sampled. Thus, we observe that the random node sampling is not an efficient technique, as expected. However, this also shows that, if one manages to maintain the sampling of the nodes in the transient phase, and cease to sample them in the steady state, the convergence rate of the original should be preserved, and the sampling of less nodes should lead to a slightly better steady-state performance. This is exactly what was observed, e.g., in

Figs. 17, 28, 29, 35, 37, and 41 with the algorithms proposed in Chapter 3. Based on this, we derived in Sec. 4.2 a simple model for the NMSD of the DTAS-dNLMS and DTRAS-dNLMS algorithm. The theoretical curves thus obtained matched the simulation results well in the scenario of Fig. 55. In contrast, in the simulations of Fig. 56, there is a period during which there is a mismatch between the theoretical curve and the simulation results. Despite this, in both cases we observed a good match between the predicted steady-state NMSD and the simulations.

In light of these observations, we provide in Table 10 a succinct list of the contributions of each chapter.

Table 10: Summary of the main contributions of each chapter.

Chapter	Main Contributions
2	Literature Review
3	Adaptive Sampling and Censoring algorithms: AS-dNLMS, ASC-dNLMS, DTAS-dNLMS, DTRAS-dNLMS, DTASC-RFF-dKNLMS, and Multitask AS-dNLMS
4	Theoretical analysis of the effects of random sampling on network performance and theoretical model for the NMSD of DTAS-dNLMS and DTRAS-dNLMS

Moreover, based on the conclusions presented, we believe that the following items constitute a list of interesting topics for future work:

1. **Apply the sampling mechanism to the distributed training of neural networks**. As mentioned in Sec. 2.1.4, the distributed training of neural networks has been proposed in recent years [168–171]. Given the potentially high computational cost associated with neural networks, we believe that it would be interesting to thoroughly investigate whether the sampling mechanisms proposed in this work can be extended to this type of solution. Some preliminary steps have been taken in this direction in the works [NC-3] and [NC-4] mentioned in Sec. 1.4, but further research is necessary in this regard.

2. **Investigate the existence of optimal sampling probabilities**. Eq. (4.29) yields the steady-state NMSD for arbitrary network topologies, step sizes, input signal variances, and sampling probabilities. One interesting question is whether it is possible to select the probabilities $p_{\zeta_1}, p_{\zeta_2}, \cdots, p_{\zeta_V}$ so as to minimize the steady-state NMSD, and therefore

optimize the network performance in steady state. Considering that $0 < p_{\zeta_k} \leqslant 1$ for $k = 1, \cdots, V$, we can see that this leads to a constrained optimization problem. Additionally, we could also enforce a minimum sampling probability to ensure that the network does not cease to sample all of the nodes permanently, which could have negative effects on the tracking capability, for example. From an intuitive perspective, we expect that assigning smaller sampling probabilities to the noisier nodes should lead to a better performance, but a thorough mathematical analysis is necessary to examine this issue. We believe that this topic deserves to be investigated in future works.

3. **Extend the reset mechanism of DTRAS-dNLMS to the algorithms of Secs. 3.3 and 3.4**. In Secs. 3.3 and 3.4, we extended the AS-dNLMS and DTAS-dNLMS algorithms to kernel-based and multitask networks. However, the reset mechanism of the DTRAS-dNLMS algorithm could also be adjusted to these scenarios. For this, the analysis of Sec. 3.2.3 would need to be redone. This may pose an interesting challenge for future works as well.

4. **Improve the theoretical model for the NMSD of the DTAS-dNLMS and DTRAS-dNLMS algorithms**. Currently, the model presented in Sec. 4.2 provides an accurate prediction for the steady-state NMSD. However, the simulations show that the main weakness of the model lies in the estimation of the iteration n_{switch} at which the algorithms switch their sampling probability from one to its steady-state value. For relatively low values of Δn, this prediction tends to be fairly accurate, but, as Δn increases, a noticeable mismatch arises between the theoretical model and the simulation results. Therefore, deriving a more refined version of Eqs. (4.78) and (4.79) seems to be an interesting topic for future research.

As we wrap up this work, it seems fitting to return to the title question: after all, **can adaptive networks do better with less data?** After all, when the nodes are not sampled, they do not sense their desired signals, which effectively reduces the amount of data used in the learning task. However, from the previous chapters, an appropriate answer seems to be: **not if the selection of the data is done randomly, as in the cases with the random node sampling**. However, if we control this process, so that we employ more data when the error is high, and less data otherwise, the theory and the simulations show that we can maintain the convergence rate of the synchronous network, and, in a single-task and linear scenario with a stationary environment, achieve a slightly better, rather than worse, steady-state performance. Thus, it seems reasonable to claim that, **with a proper sampling mechanism, adaptive diffusion networks may indeed, under these circumstances, do better with less data.**

REFERENCES

1 SAYED, A. H. **Adaptation, Learning, and Optimization over Networks**, volume 7. Foundations and Trends in Machine Learning, now Publishers Inc., Hanover, MA, 2014.

2 SAYED, A. H. Diffusion adaptation over networks. In CHELLAPA, R. and THEODORIDIS, S. (Ed.). **Academic Press Library in Signal Processing: array and statistical signal processing**, volume 3, chapter 9, p. 323–456. Academic Press, 2014.

3 SAYED, A. H. Adaptive networks. **Proceedings of the IEEE**, v. 102, n. 4, p. 460–497, 2014.

4 LOPES, C. G. and SAYED, A. H. Diffusion least-mean squares over adaptive networks: Formulation and performance analysis. **IEEE Transactions on Signal Processing**, v. 56, n. 7, p. 3122–3136, 2008.

5 CATTIVELLI, F. S. and SAYED, A. H. Diffusion LMS strategies for distributed estimation. **IEEE Transactions on Signal Processing**, v. 58, n. 3, p. 1035–1048, 2009.

6 CATTIVELLI, F. S.; LOPES, C. G.; and SAYED, A. H. Diffusion recursive least-squares for distributed estimation over adaptive networks. **IEEE Transactions on Signal Processing**, v. 56, n. 5, p. 1865–1877, 2008.

7 SAYED, A. H. and LOPES, C. G. Adaptive processing over distributed networks In **Transactions on Fundamentals of Electronics, Communications, and Computer Sciences**, v. E90-A, p. 1504–1510, 2007.

8 RABBAT, M. G. and NOWAK, R. D. Quantized incremental algorithms for distributed optimization. **IEEE Journal on Selected Areas in Communications**, v. 23, n. 4, p. 798–808, 2005.

9 LOPES, C. G. and SAYED, A. H. Distributed adaptive incremental strategies: Formulation and performance analysis. In: PROC. 2006 IEEE Int. Conf. Acoustics, Speech, and Signal Process. (ICASSP). 2006, v. III, P. 584–587.

10 LOPES, C. G. and SAYED, A. H. Incremental adaptive strategies over distributed networks. **IEEE Transactions on Signal Processing**, v. 55, n. 8, p. 4064 –4077, 2007.

11 OLFATI-SABER, R. and MURRAY, R. M. Consensus problems in networks of agents with switching topology and time-delays. **IEEE Transactions on automatic control**, v. 49, n. 9, p. 1520–1533, 2004.

12 XIAO, L. and BOYD, S. Fast linear iterations for distributed averaging. **Systems & Control Letters**, v. 53, n. 1, p. 65–78, 2004.

13 OLFATI-SABER, R. and SHAMMA, J. S. Consensus filters for sensor networks and distributed sensor fusion. In: PROC. 44th IEEE Conference on Decision and Control. 2005. P. 6698–6703.

14 XIAO, L.; BOYD, S.; and LALL, S. A scheme for robust distributed sensor fusion based on average consensus. In: PROC. Fourth International Symposium on Information Processing in Sensor Networks (IPSN). 2005. P. 63–70.

15 SPANOS , D. P.; OLFATI-SABER, R.; and MURRAY, R. M. Dynamic consensus on mobile networks. In: IFAC World Congress. 2005. P. 1–6.

16 BOYD, S.; XIAO, L. and LALL, S. A space-time diffusion scheme for peer-to-peer least-squares estimation. In: PROC. 5th International Conference on Information Processing in Sensor Networks. 2006. P. 168–176.

17 BARBAROSSA, S. and SCUTARI, G. Bio-inspired sensor network design. **IEEE Signal Processing Magazine**, v; 24, n. 3, p. 26–35, 2007.

18 DI LORENZO, P.; BARBAROSSA, S.; and SAYED, A. H. Bio-inspired decentralized radio access based on swarming mechanisms over adaptive networks. **IEEE Transactions on Signal Processing**, v. 61, n. 12, p. 3183–3197, 2013.

19 AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; and CAYIRCI, E. A survey on sensor networks. **IEEE Communications Magazine**, v. 40, n. 8, p. 102–114, 2002.

20 CHEN, J.; RICHARD, C.; and SAYED, A. H. Multitask diffusion adaptation over networks. **IEEE Transactions on Signal Processing**, v. 62, n. 16, p. 4129–4144, 2014.

21 CHEN, J.; RICHARD, C.; Alfred O Hero, and SAYED, A. H. Diffusion LMS for multitask problems with overlapping hypothesis subspaces. In: PROC. 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP). 2014. P. 1–6.

22 CHEN, J.; RICHARD, C.; and SAYED, A. H. Diffusion LMS over multitask networks. **IEEE Transactions on Signal Processing**, v. 63, n. 11, p. 2733–2748, 2015.

23 PLATA-CHAVES, J.; BOGDANOVIĆ, N., and BERBERIDIS, K. Distributed diffusion-based LMS for node-specific adaptive parameter estimation. **IEEE Transactions on Signal Processing**, v. 63, n. 13, p. 3448–3460, 2015.

24 NASSIF, R.; RICHARD, C.; FERRARI, A.; and SAYED, A. H. Proximal multitask learning over networks with sparsity-inducing coregularization. **IEEE Transactions on Signal Processing**, v. 64, n. 23, p. 6329–6344, 2016.

25 CHEN, J.; RICHARD, C.; and SAYED, A. H. Multitask diffusion adaptation over networks with common latent representations. **IEEE Journal of Selected Topics in Signal Processing**, v. 11, n. 3, p. 563–579, 2017.

26 GOGINENI, V. C. and CHAKRABORTY, M. Partial diffusion affine projection algorithm over clustered multitask networks. In: PROC. 2019 IEEE International Symposium on Circuits and Systems (ISCAS). 2019. P. 1–5.

27 JIN, D.; CHEN, J.; RICHARD, C.; and CHEN, J. Online proximal learning over jointly sparse multitask networks with $\ell_{\infty,1}$ regularization. **IEEE Transactions on Signal Processing**, v. 68, p. 6319–6335, 2020.

28 NASSIF, R.; VLASKI, S.; RICHARD, C.; and SAYED, A. H. Learning over multitask graphs – part I: Stability analysis. **IEEE Open Journal of Signal Processing**, v. 1, p. 28–45, 2020.

29 NASSIF, R.; VLASKI, S.; RICHARD, C.; and SAYED, A. H. Learning over multitask graphs – part II: Performance analysis. **IEEE Open Journal of Signal Processing**, v. 1, p. 46–63, 2020.

30 GOGINENI, V. C.; TALEBI, S. P.; and WERNER, S. Performance of clustered multitask diffusion LMS suffering from inter-node communication delays. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 68, n. 7, p. 2695–2699, 2021.

31 MARANO, S. and SAYED, A. H. Decision learning and adaptation over multi-task networks. **IEEE Transactions on Signal Processing**, v. 69, p. 2873–2887, 2021.

32 GAO, W.; CHEN, J.; RICHARD, C.; and HUANG, J. Diffusion adaptation over networks with kernel least-mean-square. In: PROC. 2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP). 2015. P. 217–220.

33 SHIN, B.-S.; PAUL, H.; and DEKORSY, A. Distributed kernel least squares for nonlinear regression applied to sensor networks. In: PROC. 2016 European Signal Processing Conference (EUSIPCO). 2016. P. 1588–1592.

34 CHOUVARDAS, S. and DRAIEF, M. A diffusion kernel LMS algorithm for nonlinear adaptive networks. In: PROC. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2016. P. 4164–4168.

35 SHIN, B.-S.; YUKAWA, M.; CAVALCANTE, R. L. G.; and DEKORSY, A. Distributed adaptive learning with multiple kernels in diffusion networks. **IEEE Transactions on Signal Processing**, v. 66, n. 21, p. 5505–5519, 2018.

36 GAO, W.; CHEN, J.; and ZHANG, L. Diffusion approximated kernel least mean p-power algorithm. In: PROC. 2019 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). 2019. P. 1–6.

37 BOUBOULIS, P.; CHOUVARDAS, S.; and THEODORIDIS, S. Online distributed learning over networks in RKH spaces using random Fourier features. **IEEE Transactions on Signal Processing**, v. 66, n.7, p. 1920–1932, 2017.

38 BOUBOULIS, P.; THEODORIDIS, S.; and CHOUVARDAS, S. A random Fourier features perspective of KAFs with application to distributed learning over networks. In COMINIELLO, D. and PRÍNCIPE, J. C. (Ed.). **Adaptive Learning Methods for Nonlinear System Modeling**, chapter 7, p. 149–172. Elsevier, 2018.

39 ELIAS, V. R. M.; GOGINENI, V. C.; MARTINS, W. A.; and WERNER, S. Adaptive graph filters in reproducing kernel hilbert spaces: Design and performance analysis. **IEEE Transactions on Signal and Information Processing over Networks**, v. 7, p. 62–74, 2020.

40 MITRA, R. and KADDOUM, G. Random Fourier feature-based deep learning for wireless communications. **IEEE Transactions on Cognitive Communications and Networking**, v. 8, n. 2, p. 468–479, 2022.

41 DI LORENZO, P.; ISUFI, E.; BANELLI, P.; BARBAROSSA, S.; and LEUS, G. Distributed recursive least squares strategies for adaptive reconstruction of graph signals. In: PROC. 2017 European Signal Processing Conference (EUSIPCO). 2017. P. 2289–2293.

42 DI LORENZO, P.; BANELLI, P.; BARBAROSSA, S.; and SARDELLITTI, S. Distributed adaptive learning of graph signals. **IEEE Transactions on Signal Processing**, v. 65, n. 16, p. 4193–4208, 2017.

43 DI LORENZO, P.; BANELLI, P.; ISUFI, E.; BARBAROSSA, S.; and LEUS, G. Adaptive graph signal processing: Algorithms and optimal sampling strategies. **IEEE Transactions on Signal Processing**, v. 66, n. 13, p. 3584–3598, 2018.

44 ANIS, A.; GADDE, A.; and ORTEGA, A. Efficient sampling set selection for bandlimited graph signals using graph spectral proxies. **IEEE Transactions on Signal Processing**, v. 64, n. 14, p. 3775–3789, 2016.

45 MARQUES, A. G.; SEGARRA, S.; LEUS, G.; and RIBEIRO, A. Sampling of graph signals with successive local aggregations. **arXiv preprint arXiv:1504.04687**, 2015.

46 CHAMON, L. F. O.; and RIBEIRO, A. Greedy Sampling of Graph Signals. **arXiv preprint arXiv: arXiv:1704.01223**, 2017.

47 NASSIF, R.; RICHARD, C.; CHEN, J.; and SAYED, A. H. Distributed diffusion adaptation over graph signals. In: PROC. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018. P. 4129–4133.

48 HUA, F.; NASSIF, R.; RICHARD, C.; WANG, H.; and SAYED, A. H. A preconditioned graph diffusion LMS for adaptive graph signal processing. In: PROC. 2018 European Signal Processing Conference (EUSIPCO). 2018. P. 111–115.

49 YUAN, K.; YING, B.; ZHAO, X.; and SAYED, A. H. Exact diffusion for distributed optimization and learning – part I: Algorithm development. **IEEE Transactions on Signal Processing**, v. 67, n. 3, p. 708–723, 2018.

50 YUAN, K.; YING, B.; ZHAO, X.; and SAYED, A. H. Exact diffusion for distributed optimization and learning – part II: Convergence analysis. **IEEE Transactions on Signal Processing**, v. 67, n. 3, p. 724–739, 2018.

51 BORDIGNON, V.; MATTA, V.; and SAYED, A. H. Adaptive social learning. **IEEE Transactions on Information Theory**, v. 67, n. 9, p. 6053–6081, 2021.

52 INAN, Y.; KAYAALP, M.; TELATAR, E.; and SAYED, A. H. Social learning under randomized collaborations. In: PROC. 2022 IEEE International Symposium on Information Theory (ISIT). 2022. P. 115–120.

53 HU, P.; BORDIGNON, V.; VLASKI, S.; and Ali H Saye. Optimal combination policies for adaptive social learning. In: PROC. ICASSP 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2022. P. 5842–5846.

54 ARABLOUEI, R.; WERNER, S.; HUANG, Y.-F.; and DOĞANÇAY, K. Distributed least mean-square estimation with partial diffusion. **IEEE Transactions on Signal Processing**, v. 62, n. 2, p. 472–484, 2013.

55 XU, S.; DE LAMARE, R. C.; and POOR, H. V. Dynamic topology adaptation for distributed estimation in smart grids. In: PROC. 2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP). 2013. P. 420–423.

56 CHOUVARDAS, S.; SLAVAKIS, K.; and THEODORIDIS, S. Trading off complexity with communication costs in distributed adaptive learning via Krylov subspaces for dimensionality reduction. **IEEE Journal of Selected Topics in Signal Processing**, v. 7, n. 2, p. 257–273, 2013.

57 SAYIN, M. O. and KOZAT, S. S. Compressive diffusion strategies over distributed networks for reduced communication load. **IEEE Transactions on Signal Processing**, v. 62, n. 20, p. 5308–5323, 2014.

58 XU, S.; DE LAMARE, R. C.; and POOR, H. V. Distributed compressed estimation based on compressive sensing. **IEEE Signal Processing Letters**, v. 22, n. 9, p. 1311–1315, 2015.

59 GUPTA, S.; SAHOO, A. S.; and SAHOO, U. K. Partial diffusion over distributed networks to reduce inter-node communication. In: PROC. 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS). 2017. P. 1–6.

60 HARRANE, I. E. K.; FLAMARY, R.; and RICHARD, C. On reducing the communication cost of the diffusion LMS algorithm. **IEEE Transactions on Signal and Information Processing over Networks**, v. 5, n. 1, p. 100–112, 2018.

61 CARPENTIERO, M.; MATTA, V.; and SAYED, A. H. Adaptive diffusion with compressed communication. In: PROC. 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2022. P. 5672–5676.

62 CARPENTIERO, M.; MATTA, V.; and SAYED, A. H. Compressed distributed regression over adaptive networks. In: PROC. 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2023. P. 1–5.

63 LOPES, C. G. and SAYED, A. H. Diffusion adaptive networks with changing topologies. In: PROC. 2008 IEEE International Conference on Acoustics, Speech and Signal Processing. 2008. P. 3285–3288

64 WERNER, S.; HUANG, Y.-F.; DE CAMPOS, M. L. R.; and KOIVUNEN, V. Distributed parameter estimation with selective cooperation. In: PROC. 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. 2009. P. 2849–2852.

65 TAKAHASHI, N. and YAMADA, I. Link probability control for probabilistic diffusion least-mean squares over resource-constrained networks. In: PROC. 2010 IEEE International Conference on Acoustics, Speech and Signal Processing. 2010. P. 3518–3521.

66 RØRTVEIT, Ø. L.; HUSØY, J. H.; and SAYED, A. H. Diffusion LMS with communication constraints. In: PROC. 2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers. 2010. P. 1645–1649.

67 ZHAO, X. and SAYED, A. H. Single-link diffusion strategies over adaptive networks. In: PROC. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2012. p. 3749–3752.

68 XU, S.; DE LAMARE, R. C.; and POOR, H. V. Adaptive link selection algorithms for distributed estimation. **EURASIP Journal on Advances in Signal Processing**, v. 2015, n. 1, p. 86, 2015.

69 ARABLOUEI, R.; WERNER, S.; DOĞANÇAY, K.; and HUANG, Y.-F. Analysis of a reduced-communication diffusion LMS algorithm. **Signal Processing**, v. 117, p. 355–361, 2015.

70 CHEN, F. and SHAO, X. Broken-motifs diffusion LMS algorithm for reducing communication load. **Signal Processing**, v. 133, p. 213–218, 2017.

71 RASTEGARNIA, A. Reduced-communication diffusion RLS for distributed estimation over multi-agent networks. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 67, n. 1, p. 177–181, 2019.

72 ARROYO-VALLES, R.; MALEKI, S.; and LEUS, G. A censoring strategy for decentralized estimation in energy-constrained adaptive diffusion networks. In: PROC. 2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC). 2013. P. 155–159.

73 GHAREHSHIRAN, O. N.; KRISHNAMURTHY, V.; and YIN, G. Distributed energy-aware diffusion least mean squares: Game-theoretic learning. **IEEE Journal of Selected Topics in Signal Processing**, v. 7, n. 5, p. 821–836, 2013.

74 FERNANDEZ-BES, J.; ARROYO-VALLES, R.; ARENAS-GARCÍA, J., and CID-SUEIRO, J. Censoring diffusion for harvesting WSNs. In: PROC. 2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP). 2015. P. 237–240.

75 BERBERIDIS, D. K.; KEKATOS, V.; WANG, G.; and GIANNAKIS, G. B. Adaptive censoring for large-scale regressions. In: PROC. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2015. P. 5475–5479.

76 YU, C.-K.; VAN DER SCHAAR, M.; and SAYED, A. H. Information-sharing over adaptive networks with self-interested agents. **IEEE Transactions on Signal and Information Processing over Networks**, v. 1, n.1, p. 2–19, 2015.

77 WANG, Z.; YU, Z.; LING, Q.; BERBERIDIS, D. K.; and GIANNAKIS, G. B. Distributed recursive least-squares with data-adaptive censoring. In: PROC. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2017. P. 5860–5864.

78 WANG, Z.; YU, Z.; LING, Q.; BERBERIDIS, D. K.; and GIANNAKIS, G. B. Decentralized RLS with data-adaptive censoring for regressions over large-scale networks. **IEEE Transactions on Signal Processing**, v. 66, n. 6, p. 1634–1648, 2018.

79 YANG, L.; ZHU, H.; KANG, K.; LUO, X.; QIAN, H.; and YANG, Y. Distributed censoring with energy constraint in wireless sensor networks. In: PROC. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018. P. 6428–6432.

80 YANG, L.; ZHU, H.; WANG, H.; KANG, K.; and QIAN, H. Data censoring with network lifetime constraint in wireless sensor networks. **Digital Signal Processing**, v. 92, p. 73–81, 2019.

81 TIGLEA, D. G.; CANDIDO, R.; and SILVA, M. T. M. A low-cost algorithm for adaptive sampling and censoring in diffusion networks. **IEEE Transactions on Signal Processing**, v. 69, 58–72, 2020.

82 TIGLEA, D. G.; CANDIDO, R.; and SILVA, M. T. M. An adaptive algorithm for sampling over diffusion networks with dynamic parameter tuning and change detection mechanisms. **Digital Signal Processing**, v. 127 p. 103587, 2022.

83 XU, P.; WANG, Y.; CHEN, X.; and TIAN, Z. Coke: Communication-censored decentralized kernel learning. **Journal of Machine Learning Research**, , v. 22, n. 196, p. 1–35, 2021.

84 TIGLEA, D. G.; CANDIDO, R.; AZPICUETA-RUIZ, L. A.; and SILVA, M. T. M. Reducing the communication and computational cost of random Fourier features kernel LMS in diffusion networks. In: PROC. 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2023. P. 1–5.

85 OSSEIRAN, A.; MONSERRAT, J. F.; and MARSCH, P. **5G mobile and wireless communications technology**. Cambridge University Press, 2016.

86 MIRAZ, M. H.; ALI, M.; EXCELL, P. S.; and PICKING, R. A review on internet of things (IoT), internet of everything (IoE) and internet of nano things (IoNT). In: PROC. 2015 Internet Technologies and Applications (ITA). 2015. P. 219–224.

87 KUMAR, S. and RAZA, Z. Internet of things: possibilities and challenges. **Fog Computing: Breakthroughs in Research and Practice**, p. 1–24, 2018.

88 TOMKOS, I.; KLONIDIS, D.; PIKASIS, E.; and THEODORIDIS, S. Toward the 6G network era: opportunities and challenges. **IT Professional**, v. 22, n. 1, p. 34–38, 2020.

89 GIORDANI, M.; POLESE, M.; MEZZAVILLA, M.; RANGAN, S.; and ZORZI, M. Toward 6G networks: use cases and technologies. **IEEE Communications Magazine**, v. 58, n. 3, p. 55–61, 2020.

90 BANAFAA, M; SHAYEA, I.; DIN, J.; AZMI, M. H.; ALASHBI, A.; DARADKEH, Y. I.; and ALHAMMADI, A. 6G Mobile communication technology: requirements, targets, applications, challenges, advantages, and opportunities. **Alexandria Engineering Journal**, v. 64, p. 245–274, 2023.

91 CHAFII, M.; BARIAH, L.; MUHAIDAT, S.; and DEBBAH, M. Twelve scientific challenges for 6G: rethinking the foundations of communications theory. **IEEE Communications Surveys & Tutorials**, v. 25, n. 2, p. 868–904, 2023.

92 ZHOU, D.; SHENG, M.; LI, J.; and HAN, Z. Aerospace integrated networks innovation for empowering 6G: a survey and future challenges. **IEEE Communications Surveys & Tutorials**, v. 25, n. 2, p. 975–1019, 2023.

93 LATRÉ, B.; BRAEM, B.; MOERMAN, I.; BLONDIA, C.; and DEMEESTER, P. A survey on wireless body area networks. **Wireless networks**, v. 17, n. 1, p. 1–18, 2011.

94 NEGRA, R.; JEMILI, I.; and BELGHITH, A. Wireless body area networks: Applications and technologies. **Procedia Computer Science**, v. 83, p. 1274–1281, 2016.

95 TOBÓN, D. P.; FALK, T. H.; and MAIER, M. Context awareness in WBANs: a survey on medical and non-medical applications. **IEEE Wireless Communications**, v. 20, n. 4, p. 30–37, 2013.

96 BERTRAND, A. Distributed signal processing for wireless EEG sensor networks. **IEEE Transactions on neural systems and rehabilitation engineering**, 23(6):923–935, 2015.

97 ABBASI, A. Z.; ISLAM, N.; SHAIKH, Z. A.; *et al.* A review of wireless sensors and networks' applications in agriculture. **Computer Standards & Interfaces**, v. 36, n. 2, p. 263–270, 2014.

98 PLATA-CHAVES, J.; BERTRAND, A.; MOONEN, M.; THEODORIDIS, S.; and ZOUBIR, A. M. Heterogeneous and multitask wireless sensor networks–algorithms, applications, and challenges. **IEEE Journal of Selected Topics in Signal Processing**, v. 11, n. 3, p. 450–465, 2017.

99 FEITOSA, A. E.; NASCIMENTO, V. H.; and LOPES, C. G. Low complexity distributed estimation for IoT sensor networks. In: PROC. 2021 IEEE Statistical Signal Processing Workshop (SSP). 2021. P. 136–140.

100 COELHO, R. M.; LOPES, C. G.; and FERRO, H. F. Adaptive IIR diffusion networks for IoT applications. In: PROC. 2021 IEEE Statistical Signal Processing Workshop (SSP). 2021. P. 141–145.

101 POTTIE, G. J. Wireless sensor networks. In: PROC. 1998 Information Theory Workshop (Cat. No. 98EX131). 1998. P. 139–140.

102 HAARTSEN, J. C. The Bluetooth radio system. **IEEE Personal Communications**, v. 7, n. 1, p. 28–36, 2000.

103 CROW, B. P.; WIDJAJA, I.; KIM, J. G.; and SAKAI, P. T. IEEE 802.11 wireless local area networks. **IEEE Communications Magazine**, v. 35, n. 9, p. 116–126, 1997.

104 LANSFORD, J.; STEPHENS, A.; and NEVO, R. Wi-fi (802.11 b) and Bluetooth: enabling coexistence. **IEEE Network**, v. 15, n. 5, p. 20–27, 2001.

105 BRITISH BROADCASTING CORPORATION. First 3G mobiles launched in Japan. Available: <http://news.bbc.co.uk/2/hi/business/1572372.stm>, May, 2024.

106 BULT, K.; BURSTEIN, A.; CHANG, D.; DONG, M.; FIELDING, M.; KRUGLICK, E.; HO, J.; LIN, F., LIN, T.-H.; KAISER, W. J.; *et al.* Low power systems for wireless microsensors. In: PROC. 1996 International Symposium on Low Power Electronics and Design. 1996. P. 17–21.

107 DONG, M. J.; YUNG, K. G.; and KAISER, W. J. Low power signal processing architectures for network microsensors. In: PROC. 1997 International Symposium on Low Power Electronics and Design. 1997. P. 173–177.

108 LIN, T.-H.; SANCHEZ, H.; ROFOUGARAN, R.; and KAISER, W. J. CMOS front end components for micropower RF wireless systems. In: PROC. 1998 international symposium on Low power electronics and design. 1998. P. 11–15.

109 POTTIE, G. J. and KAISER, W. J. Wireless integrated network sensors. **Communications of the ACM**, v. 43, n. 5, p. 51–58, 2000.

110 ESTRIN, D.; GIROD, L.; POTTIE, G. J.; and SRIVASTAVA, M. Instrumenting the world with wireless sensor networks. In: PROC. 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). 2001. v. 4, p. 2033–2036. IEEE, 2001.

111 RABAEY, J. M.; AMMER, M. J.; DA SILVA, J. L.; PATEL, D.; and ROUNDY, S. Picoradio supports ad hoc ultra-low power wireless networking. **Computer**, v. 33, n. 7, p. 42–48, 2000.

112 ERGEN, S. C. Zigbee/IEEE 802.15. 4 summary. **UC Berkeley, September**, v. 10, n. 17, p. 11, 2004.

113 CARLSON, D.; SHAMSI, M.; SCHNAARE, T.; DAUGHERTY, D.; POTTER, J.; NIXON, M.; *et al.* IEC 62591 WirelessHART® system engineering guide. **Revision 3.0 ed.: Emerson Process Management**, 2012.

114 SEXTON, D. SP100. 11a overview. **DOE Award DE-FC36-02GO14001, GE Global Research, Research Triangle Park, NC**, 2007.

115 CULLER, D.; CHAKRABARTI, S.; and INFUSION, I. P. 6LoWPAN: Incorporating IEEE 802.15. 4 into the IP architecture. **White paper**, 2009.

116 PREDD, J. B.; KULKARNI, S. B.; and POOR, H. V. Distributed learning in wireless sensor networks. **IEEE Signal Processing Magazine**, v. 23, n. 4, p. 56–69, 2006.

117 LOPES, C. G. and SAYED, A. H. Distributed processing over adaptive networks. In: PROC. Adaptive Sensor Array Processing Workshop, MIT Lincoln Laboratory. 2006. P. 1–5.

118 TUGNAIT, J. K.; LIU, H.; GONG, G.; and LI, T. Editorial. **EURASIP Journal on Wireless Communications and Networking**, v. 2004, n. 1, 2004.

119 STROGATZ, S. H. Exploring complex networks. **Nature**, v. 410, n. 6825, p. 268–276, 2001.

120 BRODER, A.; KUMAR, R.; MAGHOUL, F.; RAGHAVAN, P.; RAJAGOPALAN, S.; STATA, R.; TOMKINS, A.; and WIENER, J. Graph structure in the web. **Computer networks**, v. 33, p. 309–320, 2000.

121 FALOUTSOS, M.; FALOUTSOS, P.; and FALOUTSOS, C. On power-law relationships of the internet topology. **ACM SIGCOMM Computer Communication Review**, v. 29, n. 4, p. 251–262, 1999.

122 DIMENSIONS PUBLICATIONS ANALYTICAL VIEWS. Dimensions [online]. <https://ap.dimensions.ai/analytics/publication/overview/timeline?search_mode=content&search_text=wireless%20sensor%20networks&search_type=kws&search_field=text_search&year_from=1990&year_to=2023>. Accessed on 05/18/2024.; MAY 2024.

123 SCHIZAS, I. D.; RIBEIRO, A.; and GIANNAKIS, G. B. Consensus in ad hoc WSNs with noisy links–part I: Distributed estimation of deterministic signals. **IEEE Transactions on Signal Processing**, v. 56, n. 1, p. 350–364, 2007.

124 SCHIZAS, I. D.; GIANNAKIS, G. B.; ROUMELIOTIS, S. I.; and RIBEIRO, A. Consensus in ad hoc WSNs with noisy links – part II: Distributed estimation and smoothing of random signals. **IEEE Transactions on Signal Processing**, v. 56, n. 4, p. 1650–1666, 2008.

125 SAYED, A. H. **Adaptive Filters**. John Wiley & Sons, NJ, 2008.

126 HAYKIN, S. **Adaptive Filter Theory**. Pearson, Upper Saddle River, 5th edition, 2014.

127 DI LORENZO, P.; BARBAROSSA, S.; and SAYED, A. H. Decentralized resource assignment in cognitive networks based on swarming mechanisms over random graphs. **IEEE Transactions on Signal Processing**, v. 60, n. 7, p. 3755–3769, 2012.

128 INTANAGONWIWAT, C.; GOVINDAN, R.; ESTRIN, D.; HEIDEMANN, J.; and SILVA, F. Directed diffusion for wireless sensor networking. **IEEE/ACM Transactions on Networking**, v. 11, n. 1, p. 2–16, 2003.

129 ALANYALI, M.; VENKATESH, S.; SAVAS, O.; and AERON, S. Distributed Bayesian hypothesis testing in sensor networks. In: PROC. 2004 American control conference. 2004. v. 6, P. 5369–5374.

130 DELOUILLE, V.; NEELAMANI, R.; and BARANIUK, R. Robust distributed estimation in sensor networks using the embedded polygons algorithm. In: PROC. 3rd International Symposium on Information Processing in Sensor Networks (IPSN). 2004. P. 405–413.

131 LUO, Z.-Q. An isotropic universal decentralized estimation scheme for a bandwidth constrained ad hoc sensor network. **IEEE Journal on Selected Areas in Communications**, v. 23, n. 4, p. 735–744, 2005.

132 DEGROOT, M. H. Reaching a consensus. **Journal of the American Statistical Association**, v. 69, n. 345, p. 118–121, 1974.

133 BERGER, R. L. A necessary and sufficient condition for reaching a consensus using DeGroot's method. **Journal of the American Statistical Association**, v. 76, n. 374, p. 415–418, 1981.

134 ARORA, S. and BARAK, B. **Computational complexity: a modern approach**. Cambridge University Press, 2009.

135 JOHANSSON, B.; KEVICZKY, T.; JOHANSSON, M.; JOHANSSON, K. H. Subgradient methods and consensus algorithms for solving convex optimization problems. In: PROC. 2008 IEEE Conference on Decision and Control. 2008. P. 4185–4190

136 SCHIZAS, I. D.; MATEOS, G.; and GIANNAKIS, G. B. Distributed LMS for consensus-based in-network adaptive processing. **IEEE Transactions on Signal Processing**, v. 57, n. 6, p. 2365–2382, 2009.

137 DIMAKIS, A. G.; KAR, S.; MOURA, J. M. F.; RABBAT, M. G.; and SCAGLIONE, A. Gossip algorithms for distributed signal processing. **Proceedings of the IEEE**, v. 98, n. 11, p. 1847–1864, 2010.

138 LOPES, C. G. and SAYED, A. H. Diffusion least-mean squares over adaptive networks. In: PROC. 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2007. v. 3, P. 917–920.

139 TU, S.-Y. and SAYED, A. H. Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks. **IEEE Transactions on Signal Processing**, v. 60, n. 12, p. 6217–6234, 2012.

140 ABDOLEE, R. and CHAMPAGNE, B. Distributed blind adaptive algorithms based on constant modulus for wireless sensor networks. In: PROC. 2010 International Conference on Wireless and Mobile Communications. 2010. P. 303–308.

141 BOGDANOVIĆ, N., PLATA-CHAVES, J.; and BERBERIDIS, K. Distributed incremental-based LMS for node-specific adaptive parameter estimation. **IEEE Transactions on Signal Processing**, v. 62, n. 20, p. 5382–5397, 2014.

142 LU, L. and ZHAO, H. Diffusion leaky LMS algorithm: Analysis and implementation. **Signal Processing**, v. 140, p. 77–86, 2017.

143 YU, H. and XIA, X. Adaptive consensus of multi-agents in networks with jointly connected topologies. **Automatica**, v. 48, n. 8, p. 1783–1790, 2012.

144 XIAO, F. and CHEN, T. Adaptive consensus in leader-following networks of heterogeneous linear systems. **IEEE Transactions on Control of Network Systems**, v. 5, n. 3, p. 1169–1176, 2017.

145 ZHANG, H.; ZHOU, X.; WANG, Z.; YAN, H.; and SUN, J. Adaptive consensus-based distributed target tracking with dynamic cluster in sensor networks. **IEEE Transactions on cybernetics**, v. 59, n. 5, p. 1580–1591, 2018.

146 ZHAO, X. and SAYED, A. H. Asynchronous adaptation and learning over networks – part I: Modeling and stability analysis. **IEEE Transactions on Signal Processing**, v. 63, n. 4, p. 811–826, 2014.

147 ZHAO, X. and SAYED, A. H. Asynchronous adaptation and learning over networks – part II: Performance analysis. **IEEE Transactions on Signal Processing**, v. 63, n. 4, p. 827–842, 2014.

148 ZHAO, X. and SAYED, A. H. Asynchronous adaptation and learning over networks – part III: Comparison analysis. **IEEE Transactions on Signal Processing**, v. 63, n. 4, p. 843–858, 2014.

149 LIU, W.; PRÍNCIPE, J. C.; and HAYKIN, S. **Kernel adaptive filtering: a comprehensive introduction**, volume 57. John Wiley & Sons, 2011.

150 ENGEL, Y.; MANNOR, S.; and MEIR, R. The kernel recursive least-squares algorithm. **IEEE Transactions on Signal Processing**, v. 52, n. 8, p. 2275–2285, 2004.

151 RICHARD, C.; BERMUDEZ, J. C. M.; and HONEINE, P. Online prediction of time series data with kernels. **IEEE Transactions on Signal Processing**, v. 57, n. 3, p. 1058–1067, 2008.

152 SIMIĆ, S. N. and SASTRY, S. Distributed environmental monitoring using random sensor networks. In **Information Processing in Sensor Networks**, p. 582–592. Springer, 2003.

153 LI, X.; SHI, Q.; XIAO, S.; DUAN, S; and CHEN, F. A robust diffusion minimum kernel risk-sensitive loss algorithm over multitask sensor networks. **Sensors**, v. 19, n. 10, p. 2339, 2019.

154 NASSIF, R.; RICHARD, C.; FERRARI, A.; and SAYED, A. H. Multitask diffusion adaptation over asynchronous networks. **IEEE Transactions on Signal Processing**, v. 64, n. 11, p. 2835–2850, 2016.

155 DJURIĆ, P. M. Editorial. **IEEE Transactions on Signal and Information Processing over Networks**, v. 1, n. 1, p. 1–1, 2015.

156 O'DEA, S. Number of smartphones sold to end users worldwide from 2007 to 2021 (in million units). Statista [online]. <https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/>. Accessed on 11/29/2024; May 2024.

157 PERRIN, A. Social media usage. **Pew Research Center**, v. 125, p. 52–68, 2015.

158 MANYIKA, J.; CHUI, M.; BROWN, B.; BUGHIN, J.; DOBBS, R.; ROXBURGH, C.; BYERS, A. H.; *et al.* **Big data: The next frontier for innovation, competition, and productivity**. McKinsey Global Institute, 2011.

159 ZWOLENSKI, M. and WEATHERILL, L. The digital universe: Rich data and the increasing value of the internet of things. **Journal of Telecommunications and the Digital Economy**, v. 2, n. 3, p. 47, 2014.

160 SANDRYHAILA, A. and MOURA, J. M. F. Discrete signal processing on graphs. **IEEE Transactions on Signal Processing**, v. 61, n. 7, p. 1644–1656, 2013.

161 DI LORENZO, P.; BARBAROSSA, S.; BANELLI, P.; and SARDELLITTI, S. Adaptive least mean squares estimation of graph signals. **IEEE Transactions on Signal and Information Processing over Networks**, v. 2, n. 4, p. 555–568, 2016.

162 SHUMAN, D. I.; NARANG, S. K.; FROSSARD, P.; ORTEGA, A.; and VANDERGHEYNST, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. **IEEE Signal Processing Magazine**, v. 30, n. 3, p. 83–98, 2013.

163 ORTEGA, A.; FROSSARD, P.; KOVAČEVIĆ, J.; MOURA, J. M. F., and VANDERGHEYNST, P. Graph signal processing: Overview, challenges, and applications. **Proceedings of the IEEE**, v. 106, n. 5, p. 808–828, 2018.

164 DONG, X.; THANOU, D.; RABBAT, M.; and FROSSARD, P. Learning graphs from data: A signal representation perspective. **IEEE Signal Processing Magazine**, v. 36, n. 3, p. 44–63, 2019.

165 GIANNAKIS, G. B.; Yanning Shen, and Georgios Vasileios Karanikolas. Topology identification and learning over graphs: Accounting for nonlinearities and dynamics. **Proceedings of the IEEE**, v. 106, n. 5, p. 787–807, 2018.

166 SANDRYHAILA, A. and MOURA, J. M. F. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. **IEEE Signal Processing Magazine**, v. 31, n. 5, p. 80–90, 2014.

167 MOURA, J. M. F. Chapter 8 - Graph Signal Processing. In DJURIĆ P. M. and RICHARD, C. (Ed.). **Cooperative and Graph Signal Processing**, p. 239–259. Academic Press, 2018.

168 LIU, B.; DING, Z.; and LV, C. Distributed training for multi-layer neural networks by consensus. **IEEE Transactions on neural networks and learning systems**, v. 31, n. 5, p. 1771–1778, 2019.

169 LIU, B. and DING, Z. Distributed heuristic adaptive neural networks with variance reduction in switching graphs. **IEEE Transactions on Cybernetics**, v. 51, n. 7, p. 3836–3844, 2019.

170 VLASKI, S. and SAYED, A. H. Competing adaptive networks. In: PROC. 2021 IEEE Statistical Signal Processing Workshop (SSP). 2021. P. 71–75.

171 WANG, Z.; PAVAN, F. R. M.; and SAYED, A. H. Decentralized GAN training through diffusion learning. In: PROC. 2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP). 2022. P. 1–6.

172 RIEKE, N.; HANCOX, J.; LI, W.; MILLETARI, F.; ROTH, H. R.; ALBARQOUNI, S.; BAKAS, S.; GALTIER, M. N.; LANDMAN, B. A.; MAIER-HEIN, K.; *et al.* The future of digital health with federated learning. **NPJ Digital Medicine**, v. 3, 1, p. 1–7, 2020.

173 VLASKI, S. and SAYED, A. H. Distributed learning in non-convex environments–part I: Agreement at a linear rate. **IEEE Transactions on Signal Processing**, v. 69, p. 1242–1256, 2021.

174 VLASKI, S. and SAYED, A. H. Distributed learning in non-convex environments–part II: Polynomial escape from saddle-points. **IEEE Transactions on Signal Processing**, v. 69, p. 1257–1270, 2021.

175 KONEČNỲ, J.; McMAHAN, H. B.; YU, F. X.; RICHTÁRIK, P.; SURESH, A. T.; and BACON, D. Federated learning: Strategies for improving communication efficiency. **arXiv preprint arXiv:1610.05492**, 2016.

176 KONEČNỲ, J.; McMAHAN, H. B.; RAMAGE, D.; and RICHTÁRIK, P. Federated optimization: Distributed machine learning for on-device intelligence. **arXiv preprint arXiv:1610.02527**, 2016.

177 LI, T.; SAHU, A. K.; TALWALKAR, A.; and SMITH, V. Federated learning: Challenges, methods, and future directions. **IEEE Signal Processing Magazine**, v. 37, n. 3, p. 50–60, 2020.

178 NIKNAM, S.; DHILLON, H. S.; and REED, J. H. Federated learning for wireless communications: Motivation, opportunities, and challenges. **IEEE Communications Magazine**, v. 58, n. 6, p. 46–51, 2020.

179 BONAWITZ, K.; EICHNER, H.; GRIESKAMP, W.; HUBA, D.; INGERMAN, A.; IVANOV, V.; KIDDON, C.; KONEČNỲ, J.; MAZZOCCHI, S.; McMAHAN, B.; *et al.* Towards federated learning at scale: System design. **Proceedings of Machine Learning and Systems**, v. 1, p. 374–388, 2019.

180 BERTRAND, A.; MOONEN, M.; and SAYED, A. H. Diffusion bias-compensated RLS estimation over adaptive networks. **IEEE Transactions on Signal Processing**, v. 58, n. 11, p. 5212–5224, 2011.

181 ARABLOUEI, R.; DOĞANÇAY, K.; WERNER, S.; and HUAN, Y.-F. Adaptive distributed estimation based on recursive least-squares and partial diffusion. **IEEE Transactions on Signal Processing**, v. 62, n. 14, p. 3510–3522, 2014.

182 LIU, Z.; LIU, Y.; and LI, C. Distributed sparse recursive least-squares over networks. **IEEE Transactions on Signal Processing**, v. 62, n. 6, p. 1386–1395, 2014.

183 BAQI, S. A.; ZERGUINE, A.; and BIN SAEED, M. O. Diffusion normalized least mean squares over wireless sensor networks. In: PROC. 2013 International Wireless Communications and Mobile Computing Conference (IWCMC). 2013. P. 1454–1457.

184 JUNG, S. M.; SEO, J.-H.; and PARK, P. G. A variable step-size diffusion normalized least-mean-square algorithm with a combination method based on mean-square deviation. **Circuits, Systems, and Signal Processing**, v. 34, n. 10, p. 3291–3304, 2015.

185 LI, L. and CHAMBERS, J. A. Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology. In: PROC. 2009 IEEE/SP Workshop on Statistical Signal Processing (SSP). 2009. P. 757–760.

186 ABADI, M. S. E. and SHAFIEE, M. S. Distributed estimation over an adaptive diffusion network based on the family of affine projection algorithms. **IEEE Transactions on Signal and Information Processing over Networks**, v. 5, n. 2, p. 234–247, 2018.

187 CHOUVARDAS, S.; SLAVAKIS, K.; and THEODORIDIS, S. Adaptive robust distributed learning in diffusion sensor networks. **IEEE Transactions on Signal Processing**, v. 59, n. 10, p. 4692–4707, 2011.

188 RABBAT, M. and NOWAK, R. Distributed optimization in sensor networks. In: Proc. 3rd International Symposium on Information Processing in Sensor Networks (IPSN). 2004. P. 20–27.

189 LI, Z. and GUAN, S. Diffusion normalized Huber adaptive filtering algorithm. **Journal of the Franklin Institute**, v. 255, n. 8, p. 3812–3825, 2018.

190 YU, Y.; ZHAO, H.; WANG, W. and LU, L. Robust diffusion Huber-based normalized least mean square algorithm with adjustable thresholds. **Circuits, Systems, and Signal Processing**, v. 39, n. 4, p. 2065–2093, 2020.

191 MARKOVSKY, I. and VAN HUFFEL, S. Overview of total least-squares methods. **Signal Processing**, v. 87, n. 10, p. 2283–2302, 2007.

192 ARABLOUEI, R.; WERNER, S.; and DOĞANÇAY, K. Diffusion-based distributed adaptive estimation utilizing gradient-descent total least-squares. In: PROC. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. 2013. P. 5308–5312.

193 LI, C.; HUANG, S.; LIU, Y.; and LIU, Y. Distributed TLS over multitask networks with adaptive intertask cooperation. **IEEE Transactions on Aerospace and Electronic Systems**, v. 52, n. 6, p. 3036–3052, 2016.

194 WANG, Z.; JIA, L.; and YANG, Z. Multi-task total least-squares adaptation over networks. In: PROC. 2018 Chinese Control Conference (CCC). 2018. P. 4300–4304.

195 LI, L.; ZHAO, H.; and LV, S. Diffusion recursive total least square algorithm over adaptive networks and performance analysis. **Signal Processing**, v. 182, p. 107954, 2021.

196 ZHAO, H.; CHEN, Y.; and LV, S. Robust diffusion total least mean m-estimate adaptive filtering algorithm and its performance analysis. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 69, n. 2, p. 654–658, 2022.

197 MODALAVALASA, S.; SAHOO, U. K.; SAHOO, A. S.; and BARAHA, S. A review of robust distributed estimation strategies over wireless sensor networks. **Signal Processing**, v. 188, p. 108150, 2021.

198 SOBRON, I.; DINIZ, P. S. R.; MARTINS, W. A.; and VELEZ, M. Energy Detection Technique for Adaptive Spectrum Sensing. **IEEE Transactions on Communications**, v. 63, n. 3, p. 617–627, 2015.

199 MATTA, V.; BRACA, P.; MARANO, S.; and SAYED, A. H. Diffusion-based adaptive distributed detection: Steady-state performance in the slow adaptation regime. **IEEE Transactions on Information Theory**, v. 62, n. 8, p. 4710–4732, 2016.

200 MATTA, V.; BRACA, P.; MARANO, S.; and SAYED, A. H. Distributed detection over adaptive networks: Refined asymptotics and the role of connectivity. **IEEE Transactions on Signal and Information Processing over Networks**, v. 2, n. 4, p. 442–460, 2016.

201 AL-SAYED, S.; PLATA-CHAVES, J.; MUMA, M.; MOONEN, M.; and Abdelhak M Zoubir. Node-specific diffusion LMS-based distributed detection over adaptive networks. **IEEE Transactions on Signal Processing**, v. 66, n. 3, p. 682–697, 2017.

202 FEITOSA, A. E.; NASCIMENTO, V. H.; and LOPES, C. G. Adaptive detection in distributed networks using maximum likelihood detector. **IEEE Signal Processing Letters**, v. 25, n. 7, p. 974–978, 2018.

203 LIU, Y.; LI, C.; and ZHANG, Z. Diffusion sparse least-mean squares over networks. **IEEE Transactions on Signal Processing**, v. 60, n. 8, p. 4480–4485, 2012.

204 CHOUVARDAS, S.; SLAVAKIS, K.; KOPSINIS, Y.; and THEODORIDIS, S. A sparsity promoting adaptive algorithm for distributed learning. **IEEE Transactions on Signal Processing**, v. 60, n. 10, p. 5412–5425, 2012.

205 DI LORENZO, P. and SAYED, A. H. Sparse distributed learning based on diffusion adaptation. **IEEE Transactions on signal processing**, v. 61, n. 6, p. 1419–1433, 2012.

206 DI LORENZO, P. Diffusion adaptation strategies for distributed estimation over Gaussian Markov random fields. **IEEE Transactions on Signal Processing**, v. 62, n. 21, p. 5748–5760, 2014.

207 DINIZ, P. S. R.; YAZDANPANAH, H; and LIMA, M. V. S. Feature LMS Algorithms. In: PROC. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing. 2018. P. 4144–4148.

208 YING, B.; YUAN, K.; and SAYED, A. H. Supervised learning under distributed features. **IEEE Transactions on Signal Processing**, v. 67, n. 4, p. 977–992, 2019.

209 LIU, Y.; ZHANG, X.; KANG, Y.; LI, L.; CHEN, T.; HONG, M.; and YANG, Q. FedBCD: A communication-efficient collaborative learning framework for distributed features. **IEEE Transactions on Signal Processing**, 70:4277–4290, 2022.

210 MUSLUOGLU, C. A.; and BERTRAND, A. A unified algorithmic framework for distributed adaptive signal and feature fusion problems – part I: Algorithm derivation. **IEEE Transactions on Signal Processing**, v. 71, p. 1863–1878, 2023.

211 BLONDEL, V. D.; HENDRICKX, J. M.; OLSHEVSKY, A.; and TSITSIKLIS, J. N. Convergence in multiagent coordination, consensus, and flocking. In: PROC. of IEEE Conference on Decision and Control European Control Conference. 2005. P. 2996–3000; 2005.

212 METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M.; TELLER, A. H.; and TELLER, E. Equation of state calculations by fast computing machines. **The journal of chemical physics**, v. 21, n. 6, p. 1087–1092, 1953.

213 HASTINGS, W. K. Monte Carlo sampling methods using Markov chains and their applications. **Biometrika**, v. 57, p.97-109, 1970.

214 ZHAO, X. and SAYED, A. H. Performance limits for distributed estimation over LMS adaptive networks. **IEEE Transactions on Signal Processing**, v. 60, n. 10, p. 5107–5124, 2012.

215 TU, S.-Y. and SAYED, A. H. Optimal combination rules for adaptation and learning over networks. In: PROC. 2011 IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP). 2011. P. 317–320.

216 ZHAO, X.; TU, S.-Y.; and SAYED, A. H. Diffusion adaptation over networks under imperfect information exchange and non-stationary data. **IEEE Transactions on Signal Processing**, v. 60, n. 7, p. 3460–3475, 2012.

217 YU, C.-K. and SAYED, A. H. A strategy for adjusting combination weights over adaptive networks. In: PROC. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2013. P. 4579–4583.

218 BOYD, S.; DIACONIS, P.; and XIAO, L. Fastest mixing Markov chain on a graph. **SIAM review**, v. 46, n. 4, p. 667–689, 2004.

219 TAKAHASHI, N.; YAMADA, I.; and SAYED, A. H. Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis. **IEEE Transactions on Signal Processing**, v. 58, n. 9, p. 4795–4810, 2010.

220 FERNANDEZ-BES, J.; ARENAS-GARCÍA, J.; and SAYED, A. H. Adjustment of combination weights over adaptive diffusion networks. In: PROC. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2014. P. 6409–6413.

221 FERNANDEZ-BES, J.; AZPICUETA-RUIZ, L. A.; ARENAS-GARCÍA, J., and SILVA, M. T. M. Distributed estimation in diffusion networks using affine least-squares combiners. **Digital Signal Processing**, v. 36, p. 1–14, 2015.

222 NAKAI, A. and HAYASHI, K. An adaptive combination rule for diffusion LMS based on consensus propagation. In: PROC. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018. P. 3839–3843.

223 FERNANDEZ-BES, J.; ARENAS-GARCÍA, J.; SILVA, M. T. M.; and AZPICUETA-RUIZ, L. A. Adaptive diffusion schemes for heterogeneous networks. **IEEE Transactions on Signal Processing**, v. 65, n. 21, p. 5661–5674, 2017.

224 MOALLEMI, C. C. and VAN ROY, B. Consensus propagation. **IEEE Transactions on Information Theory**, v. 52, n. 11, p. 4753–4766, 2006.

225 ABDOLEE, R. and VAKILIAN, V. An iterative scheme for computing combination weights in diffusion wireless networks. **IEEE Wireless Communications Letters**, v. 6, n. 4, p. 510–513, 2017.

226 LOPES, C. G.; CHAMON, L. F. O.; and NASCIMENTO, V. H. Towards spatially universal adaptive diffusion networks. In: PROC. 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP). 2014. P. 803–807.

227 SEO, J. H.; JUNG, S. M.; and PARK, P. G. A diffusion subband adaptive filtering algorithm for distributed estimation using variable step size and new combination method based on the msd. **Digital Signal Processing**, v. 48, p. 361–369, 2016.

228 ERGINBAS, Y. E.; VLASKI, S.; and SAYED, A. H. Gramian-based adaptive combination policies for diffusion learning over networks. In PROC. 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2021. P. 5215–5219.

229 ERGEN, M. and VARAIYA, P. Decomposition of energy consumption in IEEE 802.11. In PROC. 2007 IEEE International Conference on Communications. 2007. P. 403–408.

230 FEENEY L. M. and NILSSON, M. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In: PROC. IEEE Conference on Computer Communications (INFOCOM). 2001. v. 3, p. 1548–1557.

231 ALISTARCH, D.; GRUBIC, D.; LI, J.; TOMIOKA, R.; and VOJNOVIC, M. QSGD: Communication-efficient SGD via gradient quantization and encoding. **Proc. Advances in neural information processing systems**, p. 1548–1557, 2017.

232 STICH, S. U.; CORDONNIER, J.-B.; and JAGGI, M. Sparsified SGD with memory. **Proc. Advances in Neural Information Processing Systems**, 2018, p. 1709–1720.

233 NASSIF, R.; VLASKI, S.; RICHARD, C.; CHEN, J.; and SAYED, A. H. Multitask learning over graphs: An approach for distributed, streaming machine learning. **IEEE Signal Processing Magazine**, v. 37, n. 3, 14–25, 2020.

234 SCARDAPANE, S.; CHEN, J.; and RICHARD, C. Adaptation and learning over networks for nonlinear system modeling. In COMMINIELLO, D. and PRÍNCIPE, J. C. (Ed.). **Adaptive learning methods for nonlinear system modeling**, chapter 10, p. 223–242. Butterworth-Heineman, 2018.

235 SCHOLKOPF, B. and SOMLA, A. J. **Learning with Kernels: support vector machines, regularization, optimization, and beyond**. MIT Press, Cambridge, 2002.

236 STEINWART, I. and CHRISTMANN, A. **Support vector machines**. Springer Science & Business Media, 2008.

237 SHIN, B.-S.; PAUL, H.; YUKAWA, M.; and DEKORSY, A. Distributed nonlinear regression using in-network processing with multiple Gaussian kernels. In PROC. 2017 IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). 2017. P. 1–5.

238 HONG, S. and CHAE, J. Distributed online learning with multiple kernels. **IEEE Transactions on neural networks and learning systems**, v. 34, n. 3, p. 1263–1277, 2021.

239 SHIN, B.-S.; YUKAWA, M.; CAVALCANTE, R. L. G.; and DEKORSY, A. A hybrid dictionary approach for distributed kernel adaptive filtering in diffusion networks. In: PROC. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018. P. 3414–3418.

240 BOUBOULIS, P.; POUGKAKIOTIS, S.; and THEODORIDIS, S. Efficient KLMS and KRLS algorithms: A random Fourier feature perspective. In: PROC. 2016 IEEE Statistical Signal Processing Workshop (SSP). 2016. P. 1–5.

241 DONG, X.; THANOU, D.; TONI, L.; BRONSTEIN, M.; and FROSSARD, P. Graph signal processing for machine learning: A review and new perspectives. **IEEE Signal Processing Magazine**, v. 37, n. 6, p. 117–127, 2020.

242 LATOUCHE, P. and ROSSI, F. Graphs in machine learning: an introduction. In: PROC. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). 2015. P. 207–218.

243 BONDY, J. A. and MURTY, U. S. R. **Graph Theory With Applications**. Macmillan Press Ltd, 1976.

244 HUA, F.; NASSIF, R.; RICHARD, C.; WANG, H.; and SAYED, A. H. Online distributed learning over graphs with multitask graph-filter models. **IEEE Transactions on Signal and Information Processing over Networks**, v. 6, p. 63–77, 2020.

245 ALINAGHI, A.; WEISS, S.; STANKOVIC, V.; and PROUDLER, I. Graph filter design for distributed network processing: a comparison between adaptive algorithms. In: PROC. 2021 Sensor Signal Processing for Defence Conference (SSPD). 2021. P. 1–5.

246 GOGINENI, V. C.; ELIAS, V. R. M.; MARTINS, W. A.; and WERNER, S. Graph diffusion kernel LMS using random Fourier features. In: PROC. 2020 Asilomar Conference on Signals, Systems, and Computers. 2020. P. 1528–1532.

247 ELIAS, V. R. M.; GOGINENI, V. C.; MARTINS, W. A.; and WERNER, S. Kernel regression over graphs using random Fourier features. **IEEE Transactions on Signal Processing**, v. 70, p. 936–949, 2022.

248 Historical temperature dataset. Available: https://portal.inmet.gov.br/ (in Portuguese). Data used in the simulations also available at: https://github.com/dgtiglea/Daily-Average-Temperature-Brazilian-Stations, Sep. 2020.

249 PERRAUDIN, N.; PARATTE, J.; SHUMAN, D. I. Shuman; KALOFOLIAS, V.; VANDERGHEYNST, P.; and HAMMOND, D. K. GSPBOX: A toolbox for signal processing on graphs. **arXiv**, preprint:1408.5781, 2014.

250 LOPES, C. G.; NASCIMENTO, V. H.; and CHAMON, L. F. O. Distributed universal adaptive networks. **IEEE Transactions on Signal Processing**, v. 71, p. 1817–1832, 2023.

251 HOU, X.; ZHAO, H.; LONG, X. Graph diffusion kernel maximum correntropy criterion over sensor network and its performance analysis. **IEEE Sensors Journal**, v. 23, p. 14583–13591, 2023.

252 XIONG, K. and WANG, S. The online random Fourier features conjugate gradient algorithm. **IEEE Signal Processing Letters**, v. 26, n. 5, p. 740–744, 2019.

253 BUENO, A. A. and SILVA, M. T. M. Gram-Schmidt-based sparsification for kernel dictionary. **IEEE Signal Processing Letters**, v. 27, p. 1130–1134, 2020.

254 LI, Z; TON, J.-F.; OGLIC, D.; and SEJDINOVIC, D. Towards a unified analysis of random Fourier features. In: PROC. International Conference on Machine Learning. 2019. P. 3905–3914.

255 BACCIU, D.; ERRICA, F.; MICHELI, A.; and PODDA, M. A gentle introduction to deep learning for graphs. **Neural Networks**, v. 129, p. 203–221, 2020.

256 WEI, X.; YU, R.; and SUN, J. View-GCN: View-based graph convolutional network for 3d shape analysis. In: PROC. IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. P. 1850–1859.

257 XIE, Y.; YU, Z.; ZHANG, J.; WANG, Z.; and JI, S. Self-supervised learning of graph neural networks: A unified review. **IEEE Transactions on pattern analysis and machine intelligence**, v. 45, n. 2, p. 2412–2429, 2022.

258 TIGLEA, D. G.; CANDIDO, R.; and SILVA, M. T. M. An adaptive sampling technique for graph diffusion LMS algorithm. In: PROC. 2019 European Signal Processing Conference (EUSIPCO). 2019. P. 1–5.

259 TIGLEA, D. G.; CANDIDO, R.; and SILVA, M. T. M. A sampling algorithm for diffusion networks. In: PROC. 2020 European Signal Processing Conference (EUSIPCO). 2021. P. 2175–2179.

260 TIGLEA, D. G. A low-cost adaptive algorithm for sampling and censoring in diffusion networks. Master's thesis, University of São Paulo, São Paulo, Brazil, Available at <https://www.teses.usp.br/teses/disponiveis/3/3142/tde-20012021-154434/pt-br.php> 2020. (in Portuguese).

261 TIGLEA, D. G.; CANDIDO, R.; and SILVA, M. T. M. A sampling algorithm for multitask adaptive diffusion networks. In: Anais do Simpósio Brasileiro de Telecomunicações (SBrT'2023). 2023. P. 1–5 (in Portuguese).

262 GARCÍA, J. A.; AZPICUETA-RUIZ, L. A.; SILVA, M. T. M.; NASCIMENTO, V. H.; and SAYED, A. H. Combinations of adaptive filters: Performance and convergence properties. **IEEE Signal Processing Magazine**, v. 33, n. 1, p. 120–140, 2016.

263 LÁZARO-GREDILLA, M.; AZPICUETA-RUIZ, L. A.; FIGUEIRAS-VIDAL, A. R.; and GARCÍA, J. A. Adaptively biasing the weights of adaptive filters. **IEEE Transactions on Signal Processing**, v. 58, n. 7, p. 3890–3895, Jul. 2010.

264 FERNANDEZ-BES, J.; ARENAS-GARCÍA, J.; SILVA, M. T. M.; and AZPICUETA-RUIZ, L. A. Adaptive diffusion schemes for heterogeneous networks. **IEEE Transactions on Signal Processing**, v. 65, p. 5661–5674, 2017.

265 STRUTZ, T. Estimation of measurement-noise variance for variable-step-size NLMS filters. In PROC. 2019 2European Signal Processing Conference (EUSIPCO). 2019. P. 1–5.

266 CHEN, S.; VARMA, R.; SANDRYHAILA, A.; and KOVAČEVIĆ, J. Discrete signal processing on graphs: Sampling theory. **IEEE Transactions on Signal Processing**, v. 63, n. 24, p. 6510–6523, 2015.

267 TISTSVERO, M.; BARBAROSSA, S.; and DI LORENZO, P. Signals on graphs: Uncertainty principle and sampling. **IEEE Transactions on Signal Processing**, v. 64, n. 18, p. 4845–4860, 2016.

268 ZHENG, Z.; LIU, Z.; and HUANG, M. Diffusion least mean square/fourth algorithm for distributed estimation. **Signal Processing**, v. 134, n. 268–274, 2017.

269 KWONG, R. H. and JOHNSTON, E. W. A variable step size LMS algorithm. **IEEE Transactions on Signal Processing**, v. 40, p. 1633–1642, Jul. 1992.

270 ABOULNASR, T. and MAYYAS, K. A robust variable step-size LMS-type algorithm: analysis and simulations. **IEEE Transactions on signal processing**, v. 45, n. 3, p. 631–639, 1997.

271 BENESTY, J.; REY, H.; VEGA, L. R.; and TRESSENS, S. A nonparametric VSS NLMS algorithm. **IEEE Signal Processing Letters**, v. 13, n. 10, p. 581–584, 2006.

272 HUANG, H.-C.; and LEE, J. A new variable step-size NLMS algorithm and its performance analysis. **IEEE Transactions on Signal Processing**, v. 60, n. 4, p. 2055–2060, 2011.

273 ZHU, Y.-G.; LI, Y.-G.; GUAN, S.-Y.; and CHEN, Q.-S. A novel variable step-size NLMS algorithm and its analysis. **Procedia Engineering**, v. 29, p. 1181–1185, 2012.

274 HAMIDIA, M. and AMROUCHE, A. Improved variable step-size NLMS adaptive filtering algorithm for acoustic echo cancellation. **Digital Signal Processing**, v. 49, p. 44–55, 2016.

275 BERSHAD, N. J. and BERMUDEZ, J. C. M. A switched variable step size NLMS adaptive filter. **Digital Signal Processing**, v. 101, p. 102730, 2020.

276 TIGLEA, D. G.; CANDIDO, R.; and SILVA, M. T. M. A variable step size adaptive algorithm with simple parameter selection. **IEEE Signal Processing Letters**, v. 29, p. 1774–1778, 2022.

277 TU, S.-Y. and SAYED, A. H. On the influence of informed agents on learning and adaptation over networks. **IEEE Transactions on Signal Processing**, v. 61, n. 6, p. 1339–1356, 2012.

278 SAYED, A. H.; TU, S.-Y.; and CHEN, J. Online learning and adaptation over networks: More information is not necessarily better. In: PROC. 2013 Information Theory and Applications Workshop (ITA). 2013. P. 1–8.

279 SAYED, A. H.; TU, S.-Y.; CHEN, J.; ZHAO, X.; and TOWFIC, Z. J. Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior. **IEEE Signal Processing Magazine**, v. 30, n. 3, 155–171, 2013.

280 NASCIMENTO, V. H. and SILVA, M. T. M. Adaptive filters. In CHELLAPA, R. and THEODORIDIS, S. (Ed.). **Academic Press Library in Signal Processing: Signal Processing Theory and Machine Learning**, v. 1, chapter 12, p. 619–761. Academic Press, Chennai, 2014.

281 MEYER, C. D. **Matrix analysis and applied linear algebra**, volume 2. Siam, 2000.

282 LEE, J.-W.; KONG, J. T.; SONG, W.-J.; and KIM, S. E. Data-reserved periodic diffusion LMS with low communication cost over networks. **IEEE Access**, v. 6, p. 54636–54650, 2018.

283 LASSERRE, J. B. A trace inequality for matrix product. **IEEE Transactions on Automatic Control**, v. 40, n. 8, p. 1500–1501, 1995.

APPENDIX A - ESTIMATION OF THE DUTY CYCLE OF THE SAMPLING IN THE AS-DNLMS ALGORITHM

In order to estimate upper or lower bounds for $\breve{\eta}_k$, we must understand under which circumstances node k remains sampled for the greatest (or lowest) number of iterations in the mean. This can be achieved by estimating the maximum and minimum values $\mathrm{E}\{\alpha_k(n)\}$ and $\mathrm{E}\{\Delta\alpha_k(n)\}$ can assume in the mean during steady state when node k is sampled (i.e., $\zeta_k = 1$). Performing the same analysis for $\zeta_k = 0$, we can determine upper and lower bounds for $\overline{\eta}_k$. For simplicity, we assume in our calculations that (3.18) is satisfied, although the final result is generalized in Section 3.1.4 for all $\beta > 0$.

Firstly, let us assume that at the iteration n, $\alpha_k(n)$ is negative but approximately zero, which we denote by $\alpha_k(n) = 0^-$. In this case, taking expectations from both sides of (3.9) yields

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)=0^-\} = \mu_\zeta \phi_0' \sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{\varepsilon_i^2(n)\}, \tag{A.1}$$

Thus, at $n+1$ the sampling of node k resumes and, recalling (3.18), from that iteration onward, α_k begins to decrease until it becomes negative again, meaning that (A.1) yields the maximum value α_k can assume in the mean in steady state. Moreover, assuming (3.21), (A.1) yields a different value for each node k that lies in

$$\mu_\zeta \phi_0' \sigma_{\min}^2 \leqslant \mathrm{E}\{\alpha_{k_{\max}}^{\text{s.s.}}\} \leqslant \mu_\zeta \phi_0' \sigma_{\max}^2, \tag{A.2}$$

where $\mathrm{E}\{\alpha_{k_{\max}}^{\text{s.s.}}\}$ denotes the maximum value $\alpha_k(n)$ can assume in the mean in steady state. Analogously, we now assume that at a certain iteration n, $\alpha_k(n)$ is positive but approximately zero, which we denote by $\alpha_k(n) = 0^+$. Making this replacement in (3.9) and taking expectations, we obtain

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)=0^+\} = \mu_\zeta \phi_0' \mathrm{E}\left\{ \sum_{i\in\mathcal{N}_k} c_{ik}\varepsilon_i^2(n) - \beta \right\}. \tag{A.3}$$

Considering (3.18), we conclude from (A.3) that $\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n) = 0^+\} < 0$, meaning that node k ceases to be sampled at iteration $n+1$ and, therefore, from that iteration onward, α_k begins to increase until it becomes negative again. Thus, (A.3) provides the minimum value α_k can assume in the mean during steady state, which lies in the range

$$\mu_\zeta \phi_0' (\sigma_{\min}^2 - \beta) \leqslant \mathrm{E}\{\alpha_{k_{\min}}^{\text{s.s.}}\} \leqslant \mu_\zeta \phi_0' (\sigma_{\max}^2 - \beta), \tag{A.4}$$

where $\mathrm{E}\{\alpha^{\text{s.s.}}_{k_{\min}}\}$ denotes the minimum value $\alpha_k(n)$ can assume in the mean in steady state, $k = 1, \cdots, V$.

Since $\mathrm{E}\{\alpha_k(n)\}$ keeps oscillating around the point $\mathrm{E}\{\alpha_k(n)\} = 0$ during steady state, we replace $\phi'[\alpha_k(n)]$ in (3.12) by its first-order Taylor expansion around $\alpha_k(n) = 0$, which is simply equal to the constant ϕ'_0. Thus, when node k is being sampled ($\zeta_k = 1$), subtracting $\alpha_k(n)$ from both sides of (3.12) and taking expectations yields

$$-\mu_\zeta\phi'_0(\beta-\sigma^2_{\min}) \leqslant \mathrm{E}\{\Delta\alpha_k(n)\} \leqslant -\mu_\zeta\phi'_0(\beta-\sigma^2_{\max}) < 0. \tag{A.5}$$

Analogously, when the node is not sampled ($\zeta_k = 0$),

$$\mu_\zeta\phi'_0\sigma^2_{\min} \leqslant \mathrm{E}\{\Delta\alpha_k(n)\} \leqslant \mu_\zeta\phi'_0\sigma^2_{\max}. \tag{A.6}$$

From a certain iteration n_0 onward, we consider the model

$$\mathrm{E}\{\alpha_k(n_0+\eta_k)\} = \mathrm{E}\{\alpha_k(n_0)\} + \eta_k\mathrm{E}\{\Delta\alpha_k(n)\}. \tag{A.7}$$

In order to estimate an upper bound $\breve{\eta}_{\max}$ for $\breve{\eta}_k$, we assume that $\mathrm{E}\{\alpha_k(n_0)\} = \mathrm{E}\{\alpha^{\text{s.s.}}_{k_{\max}}\}$ and calculate the expected number of iterations required for $\mathrm{E}\{\alpha_k(n)\}$ to fall below zero in the scenario where the node is sampled for the maximum number of iterations. This occurs if $\mathrm{E}\{\alpha_k(n_0)\} = \mu_\zeta\phi'_0\sigma^2_{\max}$, which is the upper bound for $\mathrm{E}\{\alpha^{\text{s.s.}}_{k_{\max}}\}$, and $\mathrm{E}\{\Delta\alpha_k(n)\} = -\mu_\zeta\phi'_0(\beta - \sigma^2_{\max})$, which is the least negative variation for $\mathrm{E}\{\Delta\alpha_k(n)\}$ according to (A.5). Making $\breve{\eta}_k = \breve{\eta}_{\max}$, setting $\mathrm{E}\{\alpha_k(n_0 + \breve{\eta}_{\max})\} = 0$ in (A.7), and taking into account the fact that the node must be sampled at least once during each cycle, after some algebra we obtain (3.22). Analogously, using (A.7) for the lower bound $\breve{\eta}_k = \breve{\eta}_{\min}$, we get (3.23).

For $\overline{\eta}_k$, we replace $\breve{\eta}_k$ in (A.7) by $\overline{\eta}_k$ and consider that at the iteration n_0, $\mathrm{E}\{\alpha_k(n_0)\} = \mathrm{E}\{\alpha^{\text{s.s.}}_{k_{\min}}\}$. Thus, the upper bound $\overline{\eta}_{\max}$ for $\overline{\eta}_k$ can be obtained by setting $\mathrm{E}\{\alpha_k(n_0)\} = \mu_\zeta\phi'_0\sigma^2_{\min}$, which is the lower bound for $\mathrm{E}\{\alpha^{\text{s.s.}}_{k_{\min}}\}$, and $\mathrm{E}\{\Delta\alpha_k(n)\} = \mu_\zeta\phi'_0\sigma^2_{\min}$, which is the minimum value for $\mathrm{E}\{\Delta\alpha_k(n)\}$ according to (A.6). Thus, (3.24) is obtained. Finally, as an estimate for the lower bound $\overline{\eta}_{\min}$ of $\overline{\eta}_k$, we get (3.25).

APPENDIX B - ESTIMATION OF THE DUTY CYCLE OF THE SAMPLING IN THE DTAS-DNLMS AND DTRAS-DNLMS ALGORITHMS

In order to estimate $\breve{\eta}_k$ and $\overline{\eta}_k$, we must study the behavior of α_k once the algorithm achieves the steady state in terms of MSE. For simplicity, we assume static and deterministic weights $\{c_{ik}\}$ and the statistical independence between $\phi'_k(n)$ and the terms between brackets in (3.41). Simulation results suggest that this approximation is reasonable. Since $\alpha_k \geqslant 0$ and $\zeta_k = 1$ when node k is sampled, taking expectations from both sides of (3.41) in this case yields

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)\geqslant 0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_\zeta \mathrm{E}\{\phi'_k(n)\}\left[\sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{\varepsilon_i^2(n)\} - \gamma\mathrm{E}\{\widehat{\sigma}^2_{\mathcal{N}_k}(n)\}\right]. \quad \text{(B.1)}$$

In contrast, since $\alpha_k < 0$ and $\zeta_k = 0$ when node k is not sampled, we conclude from (3.41) that, in this case,

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)<0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_\zeta \mathrm{E}\{\phi'_k(n)\}\cdot \sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{\varepsilon_i^2(n)\}. \quad \text{(B.2)}$$

Since α_k keeps oscillating around the point $\alpha_k = 0$ in steady state, we replace ϕ'_k in (B.1) and (B.2) with its first-order Taylor expansion around $\alpha_k = 0$, which is equal to the constant ϕ'_0. Assuming that $E\{\varepsilon_i^2(n)\} \approx \sigma^2_{v_i}$ and making these replacements in (B.1) and (B.2), we then obtain

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)\geqslant 0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_\zeta\phi'_0\sigma^2_{\mathcal{N}_k}(1-\gamma) \quad \text{(B.3)}$$

and

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)<0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_\zeta\phi'_0\sigma^2_{\mathcal{N}_k}. \quad \text{(B.4)}$$

Defining $\Delta\alpha_k(n) = \alpha_k(n+1) - \alpha_k(n)$, we get

$$\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n)\geqslant 0\} = -\mu_\zeta\phi'_0\sigma^2_{\mathcal{N}_k}(\gamma-1), \quad \text{(B.5)}$$

where we have rearranged the expression since $1-\gamma < 0$, and

$$\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n)<0\} = \mu_\zeta\phi'_0\sigma^2_{\mathcal{N}_k}. \quad \text{(B.6)}$$

Analyzing (B.3) to (B.6), it is possible to determine the minimum and maximum values that α_k can assume in the mean during steady state. Let us consider that, at a certain iteration n, α_k is positive but very close to zero. Denoting this situation by $\alpha_k(n) = 0_+$, we conclude from (B.3) and (B.5) that

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)=0_+\}=-\mu_\zeta\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1). \tag{B.7}$$

Thus, we observe that $\alpha_k(n+1) < 0$. On the other hand, from (B.6) we conclude that $\mathrm{E}\{\Delta\alpha_k(n+1)\} > 0$, meaning that α_k will begin to increase in the following iteration. Hence, (B.7) provides the minimum value that α_k can achieve in the mean during steady state, i.e. $\mathrm{E}\{\alpha_{k_{\min}}^{\text{s.s.}}\}=-\mu_\zeta\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1)$.

Analogously, if we assume that at a certain iteration n, α_k is negative but close to zero, which we denote by $\alpha_k(n) = 0_-$, we obtain from (B.4) and (B.6) that

$$\mathrm{E}\{\alpha_{k_{\max}}^{\text{s.s.}}\}=\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)=0_-\}=\mu_\zeta\phi_0'\sigma_{\mathcal{N}_k}^2 \tag{B.8}$$

is the maximum value α_k can assume in the mean during steady state.

Hence, in order to estimate the expected number $\overline{\eta}_k$ of iterations per cycle in which node k is sampled, we can divide $\mathrm{E}\{\alpha_{k_{\max}}^{\text{s.s.}}\}$ by the absolute value of $\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n)\geqslant 0\}$, which will provide the number of iterations needed for $\mathrm{E}\{\alpha_k\}$ to become negative after achieving its peak value. Using (B.5), we thus obtain

$$\breve{\eta}_k=\frac{\mu_\zeta\phi_0'\sigma_{\mathcal{N}_k}^2}{\mu_\zeta\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1)}=\frac{1}{\gamma-1}. \tag{B.9}$$

In order to obtain $\overline{\eta}_k$, we follow an analogous procedure for $\mathrm{E}\{\alpha_{k_{\min}}^{\text{s.s.}}\}$. Taking (B.6) into account, we arrive at

$$\overline{\eta}_k=\frac{\mu_\zeta\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1)}{\mu_\zeta\phi_0'\sigma_{\mathcal{N}_k}^2}=\gamma-1. \tag{B.10}$$

However, taking into account the fact that $\breve{\eta}_k$ and $\overline{\eta}_k$ represent a certain number of iterations, we should expect them to be natural numbers. Since we are interested in the maximum value that $\breve{\eta}_k$ can assume, we replace it by its ceiling. Analogously, since we seek the minimum value that $\overline{\eta}_k$ can assume, we replace it by its floor. Moreover, taking into account that $\breve{\eta}_k$ and $\overline{\eta}_k$ should be greater than or equal to one, we arrive at (3.53) and (3.54).

APPENDIX C - OBTAINING χ FROM p_χ

Analyzing (3.58) and using the definition of cumulative distribution function, we conclude that, for $x > 1$,

$$F_X(x) = a_3 + \int_1^x \frac{a_4}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(\varrho - a_1)^2}{2a_2^2}\right] d\varrho. \tag{C.1}$$

If we denote the pdf of a Normal random variable with mean a_1 and standard deviation a_2 by $g_X(x)$, i.e.,

$$g_X(x) = \frac{1}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(x - a_1)^2}{2a_2^2}\right], \tag{C.2}$$

and its cdf by

$$G_X(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(\varrho - a_1)^2}{2a_2^2}\right] d\varrho, \tag{C.3}$$

we may recast (C.1) as

$$F_X(x) = a_3 + a_4\left[G_X(x) - G_X(1)\right]. \tag{C.4}$$

Since $\lim_{x\to\infty} F_X(x) = 1$, we can write a_4 in terms of a_1, a_2 and a_3. Analyzing (C.4) for $x\to\infty$, we get $1 = a_3 + a_4\left[1 - G_X(1)\right]$, from which we conclude that

$$a_4 = \frac{1 - a_3}{1 - G_X(1)}. \tag{C.5}$$

Thus, (C.1) can be recast as

$$F_X(x) = a_3 + \frac{1 - a_3}{1 - G_X(1)}\left[G_X(x) - G_X(1)\right]. \tag{C.6}$$

Finally, since $G_X(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x - a_1}{a_2\sqrt{2}}\right)\right]$, where $\text{erf}(\cdot)$ denotes the error function, we may recast (C.6) as

$$F_X(x) = a_3 + \frac{(1 - a_3)}{1 - \text{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right)}\left[\text{erf}\left(\frac{x - a_1}{a_2\sqrt{2}}\right) - \text{erf}\left(\frac{1 - a_1}{a_2\sqrt{2}}\right)\right]. \tag{C.7}$$

Replacing (C.7) in (3.59) and making $x = \chi$, we finally get

$$\operatorname{erf}\left(\frac{\chi - a_1}{a_2\sqrt{2}}\right) > \frac{1 - p_\chi - a_3}{1 - a_3}\left[1 - \operatorname{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right)\right] + \operatorname{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) = \frac{p_\chi}{1-a_3}\cdot\operatorname{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) + \frac{1-p_\chi-a_3}{1-a_3}. \tag{C.8}$$

It should be noted that if

$$\frac{p_\chi}{1-a_3}\cdot\operatorname{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) + \frac{1-p_\chi-a_3}{1-a_3} = \frac{1 - a_3 - p_\chi \operatorname{erfc}\left(\frac{1-a_1}{a_2\sqrt{2}}\right)}{1-a_3} \leqslant -1, \tag{C.9}$$

where erfc denotes the complementary error function, any value of χ satisfies (C.8). Moreover, there is always a solution to (C.8). The only case in which this would not happen is if

$$\frac{1 - a_3 - p_\chi \operatorname{erfc}\left(\frac{1-a_1}{a_2\sqrt{2}}\right)}{1-a_3} > 1, \tag{C.10}$$

i.e.,

$$p_\chi \operatorname{erfc}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) < 0, \tag{C.11}$$

which is impossible since $p_\chi \geqslant 0$ and $\operatorname{erfc}(x) > 0,\ \forall x$. Thus, assuming that (C.9) does not hold, (3.60) can be straightforwardly obtained from (C.8). If (C.9) does hold, then any choice for χ is equally fitting.

APPENDIX D - OBTAINING THE RECURSION FOR $\xi_{kk}(n)$

Taking the expectations from both sides of (4.13), we get

$$\mathrm{E}\{\|\tilde{\mathbf{w}}_k(n)\|^2\} = \xi_{kk}(n) = \sum_{i\in\mathcal{N}_k}\sum_{j\in\mathcal{N}_k} c_{ik}c_{jk}\check{x}_{ji}(n), \tag{D.1}$$

where we have defined

$$\begin{aligned}\check{x}_{ji}(n) \triangleq \mathrm{E}\Big\{&\Big\{[\mathbf{I}_M - \mu_j\zeta_j(n)\mathbf{u}_j(n)\mathbf{u}_j^{\mathrm{T}}(n)]\tilde{\mathbf{w}}_j(n-1) - \mu_j\zeta_j(n)\mathbf{u}_j(n)v_j(n)\Big\}^{\mathrm{T}}\\ &\cdot\Big\{[\mathbf{I}_M - \mu_i\zeta_i(n)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)]\tilde{\mathbf{w}}_i(n-1) - \mu_i\zeta_i(n)\mathbf{u}_i(n)v_i(n)\Big\}\Big\}.\end{aligned} \tag{D.2}$$

The analysis of (D.2) can be broken down into two cases: i) when $j = i$, and ii) when $j \neq i$. In the first situation, using **A2** and **A4**, and observing that $\mathrm{E}\{\zeta_i(n)\} = \mathrm{E}\{\zeta_i^2(n)\} = p_{\zeta_i}$, we can write

$$\begin{aligned}\check{x}_{ii}(n) =\ &\mathrm{E}\{\tilde{\mathbf{w}}_i^{\mathrm{T}}(n-1)\tilde{\mathbf{w}}_i(n-1)\} - 2\mu_i p_{\zeta_i}\mathrm{E}\{\tilde{\mathbf{w}}_i^{\mathrm{T}}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)\tilde{\mathbf{w}}_i(n-1)\}\\ &+ \mu_i^2 p_{\zeta_i}\mathrm{E}\{\tilde{\mathbf{w}}_i^{\mathrm{T}}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)\tilde{\mathbf{w}}_i(n-1)\} + \mu_i^2 p_{\zeta_i}\sigma_{v_i}^2\mathrm{E}\{\mathbf{u}_i^{\mathrm{T}}(n)\mathbf{u}_i(n)\}.\end{aligned} \tag{D.3}$$

Using Assumptions **A1** and **A3**, and following similar procedures to those used in the analysis of the MSD of the LMS algorithm, we may write (see pages 803–807 of [280])

$$\mathrm{E}\{\tilde{\mathbf{w}}_i^{\mathrm{T}}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)\tilde{\mathbf{w}}_i(n-1)\} = \sigma_{u_i}^2\xi_{ii}(n-1) \tag{D.4}$$

and

$$\mathrm{E}\{\tilde{\mathbf{w}}_i^{\mathrm{T}}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)\tilde{\mathbf{w}}_i(n-1)\} = \sigma_{u_i}^4(M+2)\xi_{ii}(n-1). \tag{D.5}$$

Thus, (D.3) can be recast as

$$\check{x}_{ii}(n) = \tau_{ii}\xi_{ii}(n-1) + \mu_i^2 p_{\zeta_i} M\sigma_{u_i}^2\sigma_{v_i}^2, \tag{D.6}$$

with τ_{ii} defined as in (4.15). Let us now analyze the case in which $j \neq i$. To make this distinction clearer, we shall replace the index i by ℓ in this case. From **A4**, we can observe that

$$\mathrm{E}\{\zeta_j(n)\zeta_\ell(n)\} = \mathrm{E}\{\zeta_j(n)\}\mathrm{E}\{\zeta_\ell(n)\} = p_{\zeta_j}p_{\zeta_\ell}. \tag{D.7}$$

Using (D.7), **A2** and **A4**, we can rewrite (D.2) for $\ell \neq j$ as

$$\begin{aligned}\breve{x}_{j\ell}(n) &= \mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\widetilde{\mathbf{w}}_\ell(n-1)\} - \mu_j p_{\zeta_j}\mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &- \mu_\ell p_\ell \mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\mathbf{u}_\ell(n)\mathbf{u}_\ell^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &+ \mu_j\mu_\ell p_{\zeta_j}p_{\zeta_\ell}\mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^{\mathrm{T}}(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\end{aligned} \tag{D.8}$$

Using **A1**, **A4**, and **A5**, from (D.8) we can write

$$\begin{aligned}\mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n-1)\} &= \mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\mathbf{u}_i(n)\mathbf{u}_i^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n-1)\}\\ &= \sigma_{u_j}^2\xi_{j\ell}(n-1)\end{aligned} \tag{D.9}$$

for any pair of nodes ℓ and j, $\ell \neq j$. Furthermore, we notice that in the fourth-order moment that appears in (D.3) is not present in (D.8), and that we may write

$$\mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^{\mathrm{T}}(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n-1)\} = \sigma_{u_j}^2\sigma_{u_\ell}^2\xi_{j\ell}(n-1). \tag{D.10}$$

Therefore, with $\tau_{j\ell}$ defined as in (4.15), we can write

$$\breve{x}_{j\ell}(n) = \tau_{j\ell}\xi_{j\ell}(n-1), \tag{D.11}$$

Thus, replacing (D.6) and (D.11) in (D.1) leads to (4.14).

As evidenced by (4.14) and (D.11), we also need to obtain a recursion for $\mathrm{E}\{\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n-1)\widetilde{\mathbf{w}}_\ell(n-1)\}$, for $j \neq \ell$, in order to analyze the evolution of the MSD of each node. Firstly, we should notice that we can rewrite $\widetilde{\mathbf{w}}_j^{T}(n)\widetilde{\mathbf{w}}_\ell(n)$ as a function of the local estimates, i.e.

$$\widetilde{\mathbf{w}}_j^{\mathrm{T}}(n)\widetilde{\mathbf{w}}_\ell(n) = \sum_{s\in\mathcal{N}_\ell}\sum_{r\in\mathcal{N}_j} c_{s\ell}c_{rj}\widetilde{\boldsymbol{\psi}}_r^{\mathrm{T}}(n)\widetilde{\boldsymbol{\psi}}_s(n). \tag{D.12}$$

Replacing (4.9) in (D.12) and taking the expectations from both sides, we arrive at

$$\xi_{j\ell}(n) = \sum_{s\in\mathcal{N}_\ell}\sum_{r\in\mathcal{N}_j} c_{s\ell}c_{rj}\breve{x}_{rs}(n). \tag{D.13}$$

Similarly to what we did for (D.1), the analysis of (D.13) can be broken down into two cases: when $s = r = t$, and when $s \neq r$. In the first case, we have that $c_{tj} \neq 0$ and $c_{t\ell} \neq 0$ only if the node t is in $\mathcal{N}_j \cap \mathcal{N}_\ell$. Thus, following an analogous procedure, we can write

$$\xi_{tt}(n) = \mu_t^2 p_{\zeta_t} M\sigma_{u_t}^2\sigma_{v_t}^2 + \tau_{tt}\xi_{tt}(n-1). \tag{D.14}$$

For $s \neq r$, we can write

$$\xi_{rs}(n) = \tau_{rs}\xi_{rs}(n-1) \tag{D.15}$$

Thus, replacing (D.14) and (D.15) in (D.13), we finally obtain (4.16).

APPENDIX E - ON THE MATRIX Γ

We begin by noting that (4.16) can be recast as

$$\xi_{j\ell}(n) = \sum_{r=1}^{V}\sum_{s=1}^{V} c_{rj}c_{s\ell}[(\tau_b - \tau_a)\delta_{rs} + \tau_a]\xi_{rs}(n-1) + \mu^2 p_\zeta M\sigma_u^2 \sum_{z=1}^{V} c_{zj}c_{z\ell}\sigma_{v_z}^2, \tag{E.1}$$

for any arbitrary j and ℓ, by simply changing the order in which the elements are added, where δ_{rs} is the Kronecker delta, i.e.,

$$\delta_{rs} = \begin{cases} 1, \text{ if } r = s, 0, \text{ otherwise} \end{cases} . \tag{E.2}$$

Thus, if $r = s$, $\xi_{rr}(n-1)$, which corresponds to the MSD of node r, is multiplied by τ_b and by $c_{rj}c_{r\ell}$. In contrast, if $r \neq s$, $\xi_{rs}(n-1)$ corresponds to the trace of the covariance matrix between $\widetilde{\mathbf{w}}_r(n-1)$ and $\widetilde{\mathbf{w}}_s(n-1)$, and is multiplied by τ_a and by $c_{rj}c_{s\ell}$. Thus, if we examine the vector $\boldsymbol{\xi}(n)$ in (4.6), we notice that it consists of V elements between each pair of consecutive MSD's in the vector, and $V(V-1)$ elements related to cross-terms.

Thus, we conclude that the matrix $\boldsymbol{\Gamma}$ that appears in (4.17) is a matrix that has V columns filled with τ_b, and between each pair of consecutive columns, there are V columns filled with τ_a. These columns are multiplied element-wise by the corresponding combination weights. As an example, let us consider a network formed by only two connected nodes. In this case, we have

$$\boldsymbol{\Gamma} = \begin{bmatrix} \tau_b c_{11}^2 & \tau_a c_{21}c_{11} & \tau_a c_{11}c_{21} & \tau_b c_{21}^2 \\ \tau_b c_{12}c_{11} & \tau_a c_{22}c_{11} & \tau_a c_{12}c_{21} & \tau_b c_{22}c_{21} \\ \tau_b c_{11}c_{12} & \tau_a c_{21}c_{12} & \tau_a c_{11}c_{22} & \tau_b c_{21}c_{22} \\ \tau_b c_{12}^2 & \tau_a c_{22}c_{12} & \tau_a c_{12}c_{22} & \tau_b c_{22}^2 \end{bmatrix}. \tag{E.3}$$

We should notice that there are $V = 2$ columns that are related to the MSD's, and, in between them, we have also two columns, which are related to the covariances. If we focus on the combination weights, we can see that the matrix $\boldsymbol{\Gamma}$ carries information from $(\mathbf{C}^{\mathrm{T}}) \otimes (\mathbf{C}^{\mathrm{T}}) = (\mathbf{C} \otimes \mathbf{C})^{\mathrm{T}}$, where the equality follows from the properties of the Kronecker product. It also carries information from τ_a and τ_b. Hence, we can see $\boldsymbol{\Gamma}$ as the element-wise multiplication of two matrices, as in (4.21): $\overline{\mathbf{C}}$, which is related to the combination weights as in (3.40), and $\boldsymbol{\Omega}$, which is related to τ_b and τ_a as in (4.36).

APPENDIX F - ON THE UPPER BOUND FOR THE NMSD FOR A GIVEN p_ζ

For compactness of notation, let us introduce the quantity

$$\varkappa \triangleq \mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \tau_a \boldsymbol{\mathcal{C}}]^{-1}\boldsymbol{\sigma}. \tag{F.1}$$

Replacing $\varkappa$ in Eq. (4.56), we thus obtain

$$\mathrm{NMSD}_{\tau_a}(\infty) = \frac{\mu^2 p_\zeta M \sigma_u^2 \varkappa}{V}. \tag{F.2}$$

Since $\varkappa$ is a scalar, and the trace of a scalar is the scalar itself, we can write

$$\varkappa = \mathrm{Tr}(\varkappa) = \mathrm{Tr}\left\{\mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \tau_a \boldsymbol{\mathcal{C}}]^{-1}\boldsymbol{\sigma}\right\}. \tag{F.3}$$

The cyclic property of the trace operator states that we can write $\mathrm{Tr}(\mathbf{M}_1\mathbf{M}_2\mathbf{M}_3) = \mathrm{Tr}(\mathbf{M}_2\mathbf{M}_3\mathbf{M}_1)$ for any arbitrary matrices $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$ of appropriate dimensions. Applying this property twice to (F.3), we can rewrite $\varkappa$ as

$$\varkappa = \mathrm{Tr}\left\{\boldsymbol{\sigma}\mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \tau_a \boldsymbol{\mathcal{C}}]^{-1}\right\}. \tag{F.4}$$

Defining $\boldsymbol{\Sigma} \triangleq \boldsymbol{\sigma}\mathbf{b}^{\mathrm{T}}$ as in (4.59) and $\mathbf{G} \triangleq [\mathbf{I}_{V^2} - \tau_a \boldsymbol{\mathcal{C}}]^{-1}$, (F.4) can be recast more compactly as

$$\varkappa = \mathrm{Tr}(\boldsymbol{\Sigma}\mathbf{G}). \tag{F.5}$$

At this point, it is worth noting that, if $\boldsymbol{\mathcal{C}}$ is symmetric, so is the matrix $\mathbf{I}_{V^2} - \tau_a \boldsymbol{\mathcal{C}}$. Since the inverse of a symmetric matrix is also symmetric, this means that, in this case, $\mathbf{G}$ is also symmetric. Since $\boldsymbol{\mathcal{C}} = (\mathbf{C} \otimes \mathbf{C})^{\mathrm{T}}$, the matrix $\boldsymbol{\mathcal{C}}$ is symmetric if, and only if, $\mathbf{C}$ is symmetric. This is guaranteed to occur if the Metropolis rule is adopted, regardless of the network topology, for example [1]. In the adaptive diffusion networks, rules for the selection of the combination weights that lead to a symmetric matrix $\mathbf{C}$ regardless of the topology are sometimes referred to as "doubly stochastic policies" [1]. Besides the Metropolis rule, another example of doubly stochastic policy is the Laplacian rule, which corresponds to Rule 4) of Table 1. If we adopt the Uniform rule instead, this is not guaranteed, since it is not a doubly stochastic policy [1]. Nonetheless, the matrix $\mathbf{C}$ may still be symmetric, depending on the network topology.

On the other hand, the matrix $\boldsymbol{\Sigma}$ is not symmetric, and in general is not positive semidefinite. Nonetheless, it has been shown in [283] that, for a real $N \times N$ matrix $\mathbf{M}_1$, and for a real symmetric $N \times N$ matrix $\mathbf{M}_2$, we may write

$$\sum_{i=1}^{N} \lambda_i(\bar{\mathbf{M}}_1)\lambda_{N-i+1}(\mathbf{M}_2) \leqslant \mathrm{Tr}(\mathbf{M}_1\mathbf{M}_2) \leqslant \sum_{i=1}^{N} \lambda_i(\bar{\mathbf{M}}_1)\lambda_i(\mathbf{M}_2), \tag{F.6}$$

where $\lambda_i(\cdot)$ denotes the i-th largest eigenvalue, and $\bar{\mathbf{M}}_1 \triangleq \frac{1}{2}\left(\mathbf{M}_1 + \mathbf{M}_1^{\mathrm{T}}\right)$. We remark that, with this notation, we have that $\lambda_1(\cdot) \geqslant \lambda_2(\cdot) \geqslant \cdots \geqslant \lambda_N(\cdot)$. In other words, $\lambda_1(\cdot)$ represents the greatest eigenvalue of its argument, i.e., $\lambda_1(\cdot) = \lambda_{\max}(\cdot)$, and $\lambda_N(\cdot)$ denotes the smallest one, i.e., $\lambda_N(\cdot) = \lambda_{\min}(\cdot)$.

In our context, if $\mathbf{C}$ is symmetric, we can replace $\mathbf{M}_2$ with $\mathbf{G}$ in (F.6), and $\bar{\mathbf{M}}_1$ with

$$\bar{\mathbf{\Sigma}} \triangleq \frac{1}{2}\left(\mathbf{\Sigma} + \mathbf{\Sigma}^{\mathrm{T}}\right)$$

as in (4.58). Doing so, we straightforwardly obtain

$$\sum_{i=1}^{V^2} \lambda_i(\bar{\mathbf{\Sigma}})\lambda_{V^2-i+1}(\mathbf{G}) \leqslant \mathrm{Tr}(\mathbf{\Sigma}\mathbf{G}) \leqslant \sum_{i=1}^{V^2} \lambda_i(\bar{\mathbf{\Sigma}})\lambda_i(\mathbf{G}), \tag{F.7}$$

Focusing on the upper bound, we remark that we can write

$$\varkappa = \mathrm{Tr}(\mathbf{\Sigma}\mathbf{G}) \leqslant \sum_{i=1}^{V^2} \lambda_i(\bar{\mathbf{\Sigma}})\lambda_i(\mathbf{G}) \leqslant \sum_{i=1}^{V^2} |\lambda_i(\bar{\mathbf{\Sigma}})\lambda_i(\mathbf{G})|. \tag{F.8}$$

At this point, it would be useful to estimate $\lambda_{\max}(\mathbf{G})$. To do so, we begin by noticing that $\mathcal{C}$ is a right-stochastic matrix. As mentioned in Sec. 4.1, one property of such matrices is that their spectral radius is equal to one [211, 281]. Moreover, at least one of their eigenvalues is necessarily equal to 1. As a result, we know for sure that $\lambda_{\max}(\mathcal{C}) = 1$. Thus, we have that $\rho(\mathcal{C}) = 1$, and therefore $\lambda_i(\mathcal{C}) \in [-1,1]$ for $i = 1, 2, \cdots, V^2$. Thus, we conclude that $\lambda_i(\tau_a\mathcal{C}) \in [-\tau_a,\tau_a]$, $\lambda_i(\mathbf{I} - \tau_a\mathcal{C}) \in [1 - \tau_a, 1 + \tau_a]$, and, consequently,

$$\lambda_i(\mathbf{G}) \in \left[\frac{1}{1+\tau_a}, \frac{1}{1-\tau_a}\right] \tag{F.9}$$

for $i = 1, \cdots, V^2$, where we used the fact that, if a certain invertible matrix $\mathbf{M}$ has an eigenvalue of λ with an associated eigenvector $\mathbf{v}$, then

$$\mathbf{M}_1\mathbf{v} = \lambda\mathbf{v} \rightarrow \mathbf{M}_1^{-1}\mathbf{M}_1\mathbf{v} = \lambda\mathbf{M}_1^{-1}\mathbf{v}. \tag{F.10}$$

Since $\mathbf{M}_1^{-1}\mathbf{M}_1 = \mathbf{I}$, we thus conclude that

$$\mathbf{v} = \lambda \mathbf{M}_1^{-1}\mathbf{v} \rightarrow \mathbf{M}_1^{-1}\mathbf{v} = \frac{1}{\lambda}\mathbf{v}, \tag{F.11}$$

i.e., $\frac{1}{\lambda}$ is the eigenvalue of $\mathbf{M}_1^{-1}$ associated with the eigenvector $\mathbf{v}$.

Proceeding with our analysis, since $\lambda_{\max}(\boldsymbol{\mathcal{C}}) = 1$, we conclude from (F.9) that

$$\lambda_{\max}(\mathbf{G}) = \frac{1}{1-\tau_a}. \tag{F.12}$$

We should notice that, since $\tau_a = (1-\mu p_\zeta \sigma_u^2)^2$, if we select

$$0 < \mu < \frac{2}{(M+2)\sigma_u^2},$$

which is a condition for the stability of the diffusion networks (see Eq. (30) of the previously submitted manuscript or Eq. (25) of the resubmitted one), then $0 \leqslant \tau_a \leqslant 1$, with $\tau_a = 1$ only if $p_\zeta = 0$. Thus, for $p_\zeta > 0$, we observe from (F.12) that $\lambda_{\max}(\mathbf{G}) > 0$. Moreover, we also notice that, since

$$-1 \leqslant \lambda_{\min}(\boldsymbol{\mathcal{C}}) \leqslant 1, \tag{F.13}$$

we have that

$$\frac{1}{1+\tau_a} \leqslant \lambda_{\min}(\mathbf{G}) \leqslant \frac{1}{1-\tau_a}. \tag{F.14}$$

Since

$$\frac{1}{1+\tau_a} > 0, \tag{F.15}$$

we thus conclude that

$$0 < \lambda_{\min}(\mathbf{G}) \leqslant \lambda_{\max}(\mathbf{G}) = \frac{1}{1-\tau_a}. \tag{F.16}$$

From (F.8) and (F.16), we notice that we can write

$$\varkappa \leqslant \sum_{i=1}^{V^2} |\lambda_i(\bar{\boldsymbol{\Sigma}})\lambda_i(\mathbf{G})| \leqslant \lambda_{\max}(\mathbf{G}) \sum_{i=1}^{V^2} |\lambda_i(\bar{\boldsymbol{\Sigma}})|. \tag{F.17}$$

Thus, we obtain

$$\varkappa \leqslant \frac{1}{1-\tau_a} \sum_{i=1}^{V^2} |\lambda_i(\bar{\boldsymbol{\Sigma}})|, \tag{F.18}$$

which, using (4.30), can be recast as

$$\varkappa \leqslant \frac{1}{\mu p_\zeta \sigma_u^2 (2 - \mu p_\zeta \sigma_u^2)} \sum_{i=1}^{V^2} |\lambda_i(\bar{\mathbf{\Sigma}})|. \tag{F.19}$$

Thus, from (F.2) and (F.19), we observe, after some algebraic manipulations, that

$$\mathrm{NMSD}_{\tau_a}(\infty) \leqslant \frac{\mu M}{2 - \mu p_\zeta \sigma_u^2} \cdot \frac{\sum_{i=1}^{V^2} |\lambda_i(\bar{\mathbf{\Sigma}})|}{V}. \tag{F.20}$$

We should notice that $\mathbf{b}^{\mathrm{T}}$ and $\boldsymbol{\sigma}$ do not depend on p_ζ at all. Therefore, the same can be said about $\lambda_i(\bar{\mathbf{\Sigma}})$. Hence, we can clearly observe from (4.57) that the upper bound for the NMSD decreases as we reduce p_ζ. As long as $\mathbf{C}$ is symmetric, this occurs regardless of the network topology, whose information lies in the $\lambda_i(\bar{\mathbf{\Sigma}})$. It is important to notice that (4.57) only holds for $p_\zeta > 0$. If we selected $p_\zeta = 0$, we would have $\tau_a = 1$ and therefore would not be able to calculate $\lambda_{\max}(\mathbf{G}) = \dfrac{1}{1 - \tau_a}$. We remark that $p_\zeta = 0$ corresponds to a situation in which the algorithm never updates its initial estimates, and is not a case of practical interest.

APPENDIX G - NETWORK PERFORMANCE WITH A K_V TOPOLOGY

Firstly, it is useful to note that

$$[\mathbf{I}_{V^2} - \tau_a \mathcal{C}]^{-1} = \sum_{n_i=0}^{\infty} (\tau_a \mathcal{C})^{n_i} = \mathbf{I}_{V^2} + \sum_{n_i=1}^{\infty} (\tau_a \mathcal{C})^{n_i}. \tag{G.1}$$

At this point, one useful property of the matrix $\mathcal{C}_{K_V} = \frac{1}{V^2}\mathbf{1}_{V^2\times V^2}$ is that $\mathcal{C}_{K_V}^2 = \mathcal{C}_{K_V}$. Thus, we have that

$$[\mathbf{I}_{V^2} - \tau_a \mathcal{C}_{K_V}]^{-1} = \mathbf{I}_{V^2} + \frac{\tau_a}{(1-\tau_a)V^2}\mathbf{1}_{V^2\times V^2}. \tag{G.2}$$

Multiplying (G.2) from the left by $\mathbf{b}^{\mathrm{T}}$ leads to

$$\mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \tau_a \mathcal{C}_{K_V}]^{-1} = \mathbf{b}^{\mathrm{T}} + \frac{\tau_a}{(1-\tau_a)V}\mathbf{1}_{V^2}^{\mathrm{T}}, \tag{G.3}$$

By applying the inverse vec operator to the right-hand side of (G.3), we obtain

$$\mathrm{vec}^{-1}\left\{\mathbf{b}^{\mathrm{T}} + \frac{\tau_a}{(1-\tau_a)V}\mathbf{1}_{V^2}\right\} = \mathbf{I}_V + \frac{\tau_a}{(1-\tau_a)V}\mathbf{1}_{V\times V}. \tag{G.4}$$

Thus, using (4.7), (G.3), and (G.4), and introducing $\xi \triangleq \mathbf{b}^{\mathrm{T}}[\mathbf{I}_{V^2} - \tau_a \mathcal{C}_{K_V}]^{-1}\mathrm{vec}\{\mathbf{C}\mathbf{R}_v\mathbf{C}^{\mathrm{T}}\}$ for compactness, we can write

$$\begin{aligned} \xi &= \mathrm{Tr}\left\{\left[\mathbf{I}_V + \frac{\tau_a}{(1-\tau_a)V}\mathbf{1}_{V\times V}\right]\frac{1}{V^2}\mathbf{1}_{V\times V}\mathbf{R}_v\mathbf{1}_{V\times V}\right\} \\ &= \frac{1}{V^2}\left(1 + \frac{\tau_a}{1-\tau_a}\right)\mathrm{Tr}\{\mathbf{1}_{V\times V}\mathbf{R}_v\mathbf{1}_{V\times V}\}, \end{aligned} \tag{G.5}$$

where we used the fact that $\mathbf{C}_{K_V} = \mathbf{C}_{K_V}^{\mathrm{T}} = \frac{1}{V}\mathbf{1}_{V\times V}$ and that $\mathbf{1}_{V\times V}^2 = V\mathbf{1}_{V\times V}$. Furthermore, since $\mathrm{Tr}\{\mathbf{M}_1\mathbf{M}_2\mathbf{M}_3\} = \mathrm{Tr}\{\mathbf{M}_2\mathbf{M}_3\mathbf{M}_1\}$ for any arbitrary matrices $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$, we get

$$\mathrm{Tr}\{\mathbf{1}_{V\times V}\mathbf{R}_v\mathbf{1}_{V\times V}\} = V\mathrm{Tr}\{\mathbf{R}_v\mathbf{1}_{V\times V}\} = V\sum_{k=1}^{V}\sigma_{v_k}^2, \tag{G.6}$$

where we took advantage from the fact that $\mathbf{R}_v$ is a diagonal matrix. Thus, we obtain

$$\xi = \frac{1}{V^2}\left(1 + \frac{\tau_a}{1-\tau_a}\right)\cdot V \cdot \sum_{k=1}^{V}\sigma_{v_k}^2 = \frac{1}{1-\tau_a}\frac{\sum_{k=1}^{V}\sigma_{v_k}^2}{V}. \tag{G.7}$$

Finally, replacing (G.7) in (4.56) leads to (4.60).