An Adaptive Algorithm for Sampling over Diffusion Networks with Dynamic Parameter Tuning and Change Detection Mechanisms

Daniel Gilio Tiglea*, Renato Candido, Magno T. M. Silva

Escola Politécnica, University of São Paulo, São Paulo, Brazil

Abstract

Recently, we proposed a sampling algorithm for diffusion networks that locally adapts the number of nodes sampled according to the estimation error. Thus, it reduces the computational cost associated with the learning task when the error is low in magnitude, e.g., during steady state, and maintains the sampling of the nodes otherwise, which enables fast convergence during the transient. However, its performance depends crucially on the choice of the parameter responsible for penalizing the sampling, which is a function of the variance of the measurement noise across the network. Inappropriate choices affect the tracking capability of the algorithm. In this paper, we propose a different solution, which automatically adjusts its own parameters based on the noise power estimation. Although its computational cost is slightly increased, this modification removes the need for *a priori* knowledge of the noise variance across the network, and increases its robustness to the presence of noisy nodes in the network. Furthermore, by implementing an adaptive reset system for the sampling mechanism, we are able to significantly improve the tracking capability of the original algorithm.

Keywords: Diffusion networks, distributed estimation, adaptive filtering, sampling, censoring.

1. Introduction

Adaptive diffusion networks have consolidated themselves in the literature as an effective tool for distributed parameter estimation and signal processing. Since they dismiss the need for a central processing unit, they present better scalability, autonomy, and flexibility in comparison with centralized approaches [1–7]. Thus, they have been recognized as effective solutions in many applications, such as target localization and tracking [8], spectrum sensing in mobile networks [9, 10], collaborative beamforming [11], modeling of biological systems [12], internet of things (IoT) applications [13], among others [1–7].

Adaptive networks consist of a set of connected nodes that are capable of sensing and processing data, as well as communicating with their nearby peers, i.e., their neighbors. The collective goal of the network is to

*Corresponding author.

Email addresses: dtiglea@lps.usp.br (Daniel Gilio Tiglea), renatocan@lps.usp.br (Renato Candido), magno.silva@usp.br (Magno T. M. Silva)

estimate a parameter vector of interest in a distributed way [1–7, 14–16]. To achieve this, each node usually collects the data available locally and computes its *local* estimate in what is called the *adaptation step*. Then, the nodes cooperate with their neighbors to reach a *global* estimate of the vector of interest. This, in its turn, is usually called the *combination step*. The order in which these steps are performed enables two possible schemes: the Adapt-Then-Combine (ATC) and Combine-Then-Adapt (CTA) strategies [1–7, 14–16].

However, in certain distributed applications, measuring and processing the data at every node and every time instant can be costly, even prohibitively so. Hence, some sort of mechanism may be required to decrease the number of updates and/or measurements [17, 18]. In this context, *sampling* only a subset of nodes at each iteration can significantly reduce the computational and memory burdens associated with the learning task, but may also affect algorithm performance. Moreover, in order to make these distributed solutions feasible for most applications, it is often necessary to reduce the amount of information sent across the network to reduce energy consumption, as is the case of wireless sensor networks [19–21]. Thus, several solutions have been proposed to reduce the energy consumption associated with the communication between nodes. Some reduce the amount of information sent in each transmission [22–28], whereas others shut links off according to selective communication policies [19, 29–35]. Finally, there is a group of solutions known as *censoring* techniques. They seek to cut the transmission from certain nodes to any of their neighbors [20, 21, 36–42], hence allowing censored nodes to turn their transmitters off. This saves energy and reduces the amount of information used in the processing [21, 37].

Recently, we proposed a sampling algorithm for adaptive diffusion networks and graph adaptive filtering that can also be used as a censoring technique [43]. It changes the number of sampled nodes based on the squared error in the neighborhood of each node. Thus, the number of sampled nodes decreases when the error is low in magnitude, allowing for fast convergence in the transient and a significant reduction in the computational cost and/or energy consumption in steady state. It has attained success in temperature estimation problems using real-world data as well as synthetic computer simulations [43]. However, the proper selection of one of its parameters depends crucially on the knowledge of the largest noise power in the network, which may not be known beforehand.

Moreover, one of the main limitations of the mechanism proposed in [43] lies in its tracking capability. Since the sampling of the nodes does not cease permanently, it can detect changes in the environment. However, under certain circumstances, its performance is noticeably affected in comparison with the case where all nodes remain sampled. It was shown in [43] that the proposed algorithm could respond satisfactorily to abrupt changes in the environment, but further experiments showed that this is not always the case. To illustrate this, in Sec. 1.1 we provide some simulation results that exemplify some of the main weaknesses of the algorithm of [43] and help us motivate the additional mechanisms proposed in this work. Further, the main contributions of this paper and its organization are outlined in Secs. 1.2 and 1.3, respectively.

1.1. Introductory Simulations

We show in Fig. 1 simulation results obtained with the diffuse Normalized Least-Mean-Squares (dNLMS) algorithm [1–3] and with the algorithm proposed in [43], named as Adaptive-Sampling dNLMS (AS-dNLMS), in a system identification problem. We consider Scenario 1 described in Sec. 8, and both algorithms were implemented in an ATC configuration over a network consisting of 25 nodes. For comparison, we included the results of the original dNLMS algorithm with all 25 nodes sampled, and with a sampling technique in which $V_s = 5$ nodes are randomly selected for sampling at each iteration. 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. In Fig. 1(a) we present the Network Mean-Square Deviation (NMSD), a performance indicator that will be detailed in Sec. 8, whereas Figs. 1(b) and (c) depict the number of nodes sampled, and of multiplications per iteration, respectively.

From Fig. 1(a), we observe that the random sampling leads to a slower convergence rate that the original dNLMS algorithm. This is reasonable since the random sampling reduces the rate at which information is fed into the algorithm, which is crucial during the transient. On the other hand, before the abrupt change in the environment, AS-dNLMS preserves the convergence rate of dNLMS, since it maintains the sampling of every node until it achieves the steady state in terms of NMSD. From Figs. 1(b) and (c), we see that AS-dNLMS samples on average two nodes per iteration during steady state, which leads to a significant reduction in the computational cost in comparison with dNLMS. It is also interesting to note that the sampling of less nodes leads to a slight decrease in steady-state NMSD. AS-dNLMS and dNLMS with $V_s = 5$ respectively achieve steady-state NMSD levels of 2.8 dB and 2 dB lower than that of dNLMS with all nodes sampled. One possible interpretation for this is that although the adaptation step is important for the convergence during the transient and for detecting changes in the environment, in steady state it introduces noise into the network, which the combination step tends to remove [43, 44]. Thus, by reducing the sampling rate during steady state, there may be a slight reduction in NMSD. However, after the abrupt change, the convergence rate of AS-dNLMS becomes slower than that of the dNLMS with $V_s = 5$. As can be seen from Fig. 1(b), AS-dNLMS does slightly increase the number of nodes sampled per iteration after the change in the environment, but this increase is too slight to improve the convergence rate.

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 their choice depends on the largest noise variance in the network. An example is depicted in Fig. 2, in which we set the parameters of the algorithm using the same rules that were employed in the simulations of Fig. 1. However, in comparison with the scenario considered in Fig. 1, the noise power at the noisiest node was multiplied by ten. 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. 1 and 2, 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

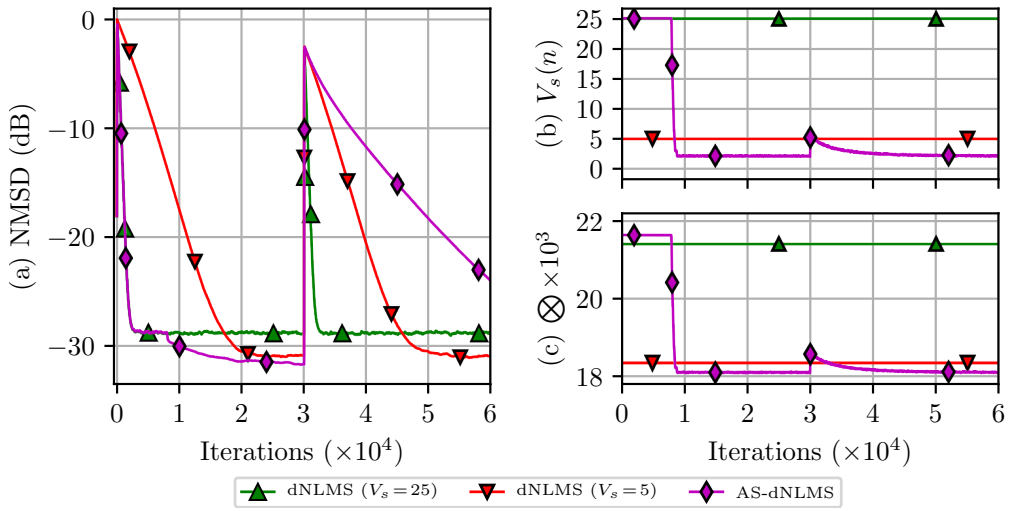

Figure 1: Comparison between dNLMS with V_s nodes randomly sampled per iteration and AS-dNLMS ($\beta = 1.9$, $\mu_s = 0.0045$). (a) NMSD along the iterations, (b) number of nodes sampled, and (c) multiplications per iteration.

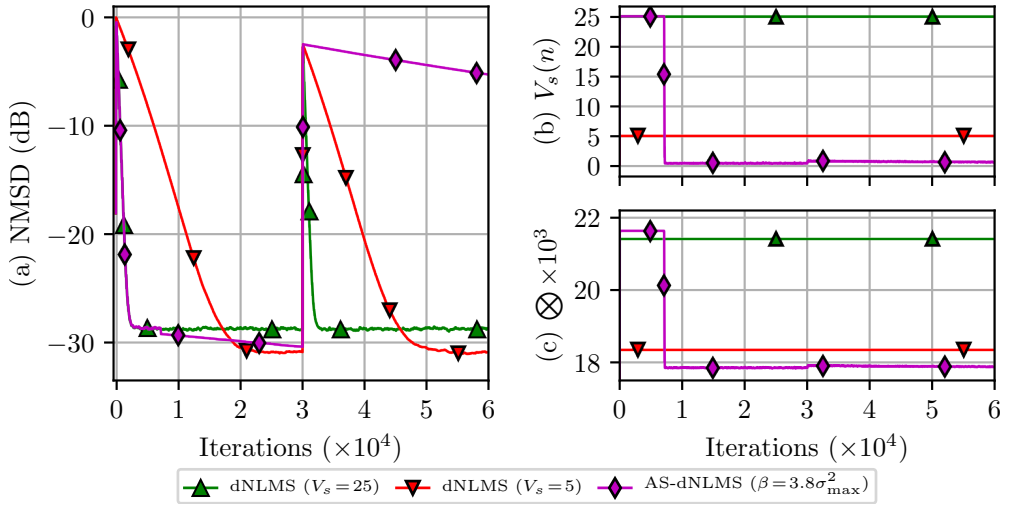

Figure 2: Comparison between dNLMS with V_s nodes randomly sampled per iteration and AS-dNLMS ($\beta = 19$, $\mu_s = 0.0045$) in Scenario 2 described in Sec. 8, in which one of the nodes is much noisier than the others. (a) NMSD curves, and (b) number of nodes sampled per iteration.

variance in this scenario is much greater than the average noise power. Thus, the selection of the parameters of the algorithm requires a deeper knowledge of the noise variance profile of the nodes, which further hinders the applicability of AS-dNLMS.

1.2. Main Contributions of the Paper

To tackle the issues illustrated by the simulations of Sec. 1.1, we propose in this paper several modifications to the AS-dNLMS algorithm that directly address all of its main limitations. The algorithm thus

derived extends our previous work in numerous ways:

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 network, e.g., due to significant variations in the measurement noise power from one node to the other, as in the simulations of Fig. 2.

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.

4. Theoretical results are derived for the proposed algorithm, enabling a well-informed selection of its parameters. Unlike what was observed for the AS-dNLMS in [43], the theoretical upper bound for the expected number of nodes sampled per iteration by the proposed algorithm does not depend on the greatest noise variance in the network.

Simulation results show the good performance of the proposed solution in different scenarios. In particular, it is shown that the proposed algorithm is more robust to the presence of noisy nodes in the network. Furthermore, in a scenario where the optimal system varies over time following a random-walk model, it is shown that it can present a significant improvement in performance in comparison with AS-dNLMS.

1.3. Organization of the Paper and Notation

The paper is organized as follows. In Sec. 2, we revisit the ATC AS-dNLMS algorithm [43]. In Sec. 3, the adaptive sampling algorithm with dynamic parameter tuning is derived, and in Sec. 4, we conduct a theoretical analysis to aid the selection of the fixed parameters of the algorithm. In Sec. 5, we propose a reset system for the sampling mechanism in order to improve the tracking capability of the algorithm. Finally, simulation results are shown in Sec. 8 and Sec. 9 closes the paper with the conclusions. For simplicity, we focus only on the ATC strategy, but our results can be straightforwardly extended to CTA as well.

Notation. We use normal fonts for scalars and boldface letters for vectors. Moreover, $(\cdot)^{\mathrm{T}}$ denotes transposition, $\mathrm{E}\{\cdot\}$ the mathematical expectation, $\exp(\cdot)$ the exponential function, $|\cdot|$ the cardinality of a set, $\mathrm{Tr}[\cdot]$ the trace of a matrix, $\Pr[\cdot]$ the probability of an event, and $\|\cdot\|$ the Euclidean norm.

2. Revisiting the ATC AS-dNLMS Algorithm

Let us consider a network of V nodes with a preset topology, such as the one of Fig. 3, which will be used in the simulations throughout this paper. Two nodes are considered neighbors if they can exchange

information, and we denote by $\mathcal{N}_k$ the neighborhood of node k, including k itself. An example is shown in Fig. 3, which depicts a network randomly generated according to the Erdös-Renyi model with $V = 25$ nodes [15]. In this case, the neighborhood of node 1 corresponds to the set of nodes $\{1, 11, 21, 24\}$, which are highlighted in red for ease of visualization, and the average number of neighbors per node is 7.

Furthermore, as illustrated in Fig. 3, we consider that each node k has access at each time instant n to an input signal $u_k(n)$ and to a reference signal

$$d_k(n) = \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}^{\mathrm{o}} + v_k(n), \tag{1}$$

where $\mathbf{u}_k(n) = [u_k(n)\ \ u_k(n–1)\ \cdots\ u_k(n–M+1)]^{\mathrm{T}}$ is an M-length regressor vector, $\mathbf{w}^{\mathrm{o}}$ is the optimal system, and $v_k(n)$ is the measurement noise at node k, which is assumed to be independent and identically distributed (iid), zero-mean with variance $\sigma_{v_k}^2$ and independent from any other signal. The objective of the network is to obtain an estimate of $\mathbf{w}^{\mathrm{o}}$ in a distributed manner by solving $\min_{\mathbf{w}} \sum_{k=1}^{V} \mathrm{E}\{[d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}]^2\}$ [1–3, 19, 20].

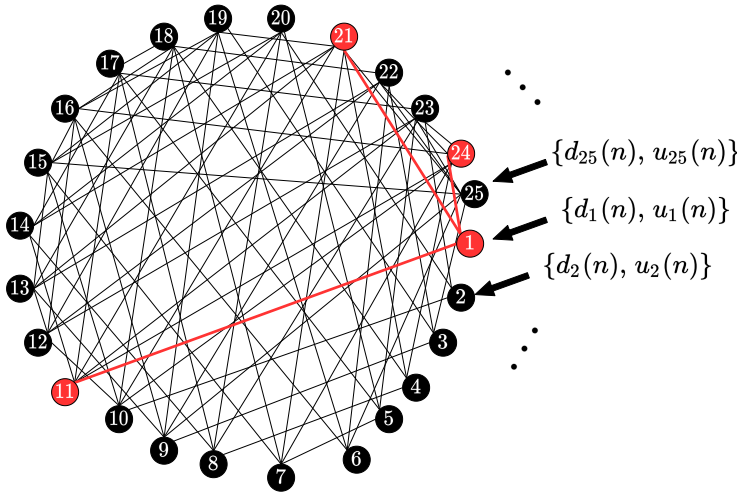

Figure 3: Example of an adaptive diffusion network and its inputs. The neighborhood of node 1 is highlighted in red.

Many adaptive algorithms were proposed for this, with the ATC dNLMS being one of the most widely used [1–3]. Based on it, in [43] we proposed the ATC AS-dNLMS algorithm. Thus, the adaptation and combination steps of ATC AS-dNLMS are respectively given by

$$\begin{cases} \boldsymbol{\psi}_k(n+1) = \mathbf{w}_k(n) + \bar{s}_k(n)\mu_k(n)\mathbf{u}_k(n)e_k(n) & \text{(2a)} \\ \mathbf{w}_k(n+1) = \sum_{j\in\mathcal{N}_k} c_{jk}\boldsymbol{\psi}_j(n+1), & \text{(2b)} \end{cases}$$

where $\bar{s}_k(n) \in \{0,1\}$ is a sampling variable and $\boldsymbol{\psi}_k$ and $\mathbf{w}_k$ represent respectively the local and combined estimates of $\mathbf{w}^{\mathrm{o}}$ at node k. Furthermore,

$$e_k(n) = d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n), \tag{3}$$

is the estimation error and $\mu_k(n) = \widetilde{\mu}_k/[\delta + \|\mathbf{u}_k(n)\|^2]$ is a normalized step size with $0 < \widetilde{\mu}_k < 2$ and a small regularization factor $\delta > 0$ [1]. Lastly, $\{c_{jk}\}$ are combination weights satisfying $c_{jk} \geqslant 0$, $\sum_{j\in\mathcal{N}_k} c_{jk} = 1$, and $c_{jk} = 0$ if $j \notin \mathcal{N}_k$ [2, 3]. Possible choices for $\{c_{jk}\}$ include the Uniform, Metropolis, and Relative Degree rules [1], as well as the Adaptive Combination Weights (ACW) algorithm [45]. In particular, ACW aims to assign greater weights to the least noisy nodes, and is obtained by solving an optimization problem in regards to $\{c_{jk}\}$. To avoid division by zero, in this paper we adopt a regularized version of it, which can be summarized as [43, 45]

$$c_{jk}(n) = \frac{[\delta_c + \widehat{\sigma}^2_{jk}(n)]^{-1}}{\sum_{\ell\in\mathcal{N}_k}[\delta_c + \widehat{\sigma}^2_{\ell k}(n)]^{-1}} \text{ if } j \in \mathcal{N}_k \text{ or } 0, \text{otherwise}, \tag{4}$$

where $\delta_c > 0$ is a small constant and σ^2_{jk} is updated as

$$\widehat{\sigma}^2_{jk}(n) = (1-\nu_k)\widehat{\sigma}^2_{jk}(n-1) + \nu_k \|\boldsymbol{\psi}_j(n+1) - \mathbf{w}_k(n)\|^2, \tag{5}$$

with $\nu_k > 0$ for $k = 1, \cdots, V$.

The difference between dNLMS and AS-dNLMS resides in the inclusion of the sampling variable $\bar{s}_k(n)$ in the correction term of (2a). Whenever $\bar{s}_k(n) = 1$, $d_k(n)$ is sampled and $e_k(n)$ is computed as in (3). In contrast, if $\bar{s}_k(n) = 0$, $d_k(n)$ is not sampled, $\mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n)$, $e_k(n)$ and $\mu_k(n)$ are not computed, and $\boldsymbol{\psi}_k(n+1) = \mathbf{w}_k(n)$.

Based on convex combinations of adaptive filters [46], instead of directly adapting $\bar{s}_k(n)$, we introduce an auxiliary variable $\alpha_k(n) \in [-\alpha^+, \alpha^+]$ such that $\bar{s}_k(n) = 0$ for $\phi[\alpha_k(n)] < 0.5$ and $\bar{s}_k(n) = 1$ otherwise, with $\phi[\cdot]$ given by [47]

$$\phi[\alpha_k(n)] \triangleq \frac{\mathrm{sgm}[\alpha_k(n)] - \mathrm{sgm}[-\alpha^+]}{\mathrm{sgm}[\alpha^+] - \mathrm{sgm}[-\alpha^+]}, \tag{6}$$

where $\mathrm{sgm}[x] = [1 + \exp(-x)]^{-1}$ is a sigmoidal function. In the literature, $\alpha^+ = 4$ is usually adopted [47]. It is interesting to notice that $\phi[\alpha^+] = 1$, $\phi[0] = 0.5$, and $\phi[-\alpha^+] = 0$. For the compactness of notation, we henceforth write $\phi[\alpha_k(n)]$ as $\phi_k(n)$. Thus, $\bar{s}_k(n)$ is related to $\alpha_k(n)$ by

$$\bar{s}_k(n) = \begin{cases} 1, \text{ if } \alpha_k(n) \geqslant 0, \\ 0, \text{ otherwise} \end{cases}. \tag{7}$$

We then introduce the following cost function [43]:

$$J_{\alpha_k}(n) = \phi_k(n)\beta\bar{s}_k(n) + \big[1 - \phi_k(n)\big] \sum_{i\in\mathcal{N}_k} c_{ik}(n) e_i^2(n), \tag{8}$$

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 (e.g., during transient), $J_{\alpha_k}(n)$ is minimized by making $\phi_k(n)$ closer to one, leading to the sampling of node k. The same holds when node k is not being sampled ($\bar{s}_k = 0$), which ensures that the sampling eventually resumes. In contrast, when node k is being sampled ($\bar{s}_k = 1$) and the error is

small in magnitude, $J_{\alpha_k}(n)$ is minimized by making $\phi_k(n)$ closer to zero, and the algorithm stops sampling node k [43].

AS-dNLMS is then obtained by taking the derivative of (8) with respect to $\alpha_k(n)$. Since we may not have access to $e_i(n)$ at every iteration due to the lack of sampling, we replace $e_i(n)$ by its latest measurement $\varepsilon_i(n)$, given by $\varepsilon_i(n) = \bar{s}_i(n)e_i(n)+[1-\bar{s}_i(n)]\varepsilon_i(n-1)$. We thus get the following stochastic gradient descent rule [43]:

$$\alpha_k(n+1)=\alpha_k(n)+\mu_s\phi'_k(n)\left[\sum_{i\in\mathcal{N}_k}c_{ik}(n)\varepsilon_i^2(n)-\beta\bar{s}_k(n)\right], \tag{9}$$

where $\mu_s > 0$ is a step size and [47]

$$\phi'_k(n)=\frac{d\phi[\alpha_k(n)]}{d\alpha_k(n)}=\frac{\mathrm{sgm}[\alpha_k(n)]\{1-\mathrm{sgm}[\alpha_k(n)]\}}{\mathrm{sgm}[\alpha^+]-\mathrm{sgm}[-\alpha^+]}. \tag{10}$$

Eqs. (2) and (9) are the core of AS-dNLMS. It preserves the convergence rate of dNLMS but features a much lower computational cost in steady state, although there is a slight increase during transient [43]. The algorithm requires that every sampled node i 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. In the censoring version of the algorithm, named adaptive-sampling-and-censoring diffusion NLMS (ASC-dNLMS), the adaptation step (2a) is simply skipped when the node is not sampled. Thus, neither $\boldsymbol{\psi}_k$ nor ε_k changes. Assuming the nodes can store past data from their neighbors, this allows us to cut the number of transmissions.

The parameter β plays a crucial role in the behavior of AS-dNLMS, strongly influencing the expected number of sampled nodes. For the sampling of node k to cease, α_k must decrease over the iterations until it becomes negative. Thus, the term between brackets in (9) must be negative in steady state when $\bar{s}_k(n)=1$. Assuming the statistical independence between this term and $\phi'_k(n)$, and that the combination weights are static and deterministic, we conclude that we must have

$$\beta > \mathrm{E}\left\{\sum_{i\in\mathcal{N}_k}c_{ik}\varepsilon_i^2(n)\right\}. \tag{11}$$

Assuming that

$$\mathrm{E}\{\varepsilon_i^2(n)\}\approx\sigma_{v_i}^2 \tag{12}$$

for $i=1,\ \cdots,\ V$ in steady state, and considering the worst-case scenario, we get that

$$\beta > \sigma_{\max}^2 \triangleq \max_i \sigma_{v_i}^2 \tag{13}$$

is a sufficient (but not necessary) condition to reduce the number of sampled nodes in steady state [43].

Even with a proper choice for β, it is beneficial to select μ_s adequately, since it affects the rate at which the nodes cease to be sampled. Adopting a linear model for $\phi'_k(n)$ for $\alpha_k \in [0,\ \alpha^+]$, it is possible to show

that, if we wish that the sampling of the nodes ceases in at most Δn iterations after AS-dNLMS achieves steady-state, we must choose [43]

$$\mu_s > \frac{\alpha^+}{(\beta - \sigma^2_{\max})(\phi'_0 - \phi'_{\alpha^+})} \left[\left(\frac{\phi'_o}{\phi'_{\alpha^+}} \right)^{\frac{1}{\Delta n}} - 1 \right], \tag{14}$$

where ϕ'_0 and ϕ'_{α^+} are constants that respectively denote the value of ϕ' evaluated at $\alpha_k = 0$ and $\alpha_k = \alpha^+$.

From (13) we see that the proper selection of β depends on the knowledge of the largest noise variance in the network, $\sigma^2_{\max}$. Thus, the choice of this parameter becomes difficult if $\sigma^2_{\max}$ is not known beforehand. This is aggravated by the fact that large values for β can harm the tracking capability of AS-dNLMS, as shown in Fig. 1 and other simulation results of [43]. Moreover, if the largest noise variance in the network is much greater than the average noise variance, i.e., one of the nodes is much noisier than the others, the condition established by (13) can lead to inappropriately high values for β, as seen in the simulations of Fig. 2. Thus, it may be desirable to adopt different β_k and μ_k for each node k, depending on its own characteristics, which is done in the sequel.

3. 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 (11) and maintaining the assumption that $\mathrm{E}\{\varepsilon_i^2(n)\} \approx \sigma^2_{v_i}$ 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

$$\beta_k(n) = \gamma \sum_{i \in \mathcal{N}_k} c_{ik} \widehat{\sigma}^2_{v_i}(n) \triangleq \gamma \widehat{\sigma}^2_{\mathcal{N}_k}(n), \tag{15}$$

where $\gamma > 1$ is a parameter that the designer must choose. Thus, (9) can be recast as

$$\alpha_k(n+1) = \alpha_k(n) + \mu_s \phi'_k(n) \times \left[\sum_{i \in \mathcal{N}_k} c_{ik}(n) \varepsilon_i^2(n) - \gamma \widehat{\sigma}^2_{\mathcal{N}_k}(n) \bar{s}_k(n) \right]. \tag{16}$$

In this paper we consider the algorithm proposed in [48] for adaptive noise power estimation. It presents a faster convergence rate than other methods, and can cope with changes in the environment, so long as $v_k(n)$ is wide-sense stationary for $k = 1, \cdots, V$ [48]. 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 [48] 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 ζ_f, ζ_m and ζ_f are employed to calculate our estimate:

$$\theta_{f_k}^2(n) = \zeta_f \theta_{f_k}^2(n-1) + (1-\zeta_f) e_k^2(n), \tag{17}$$

$$\theta_{m_k}^2(n) = \zeta_m \theta_{m_k}^2(n-1) + (1-\zeta_m) e_k^2(n), \tag{18}$$

$$\theta_{s_k}^2(n) = \zeta_s \theta_{s_k}^2(n-1) + (1-\zeta_s) e_k^2(n). \tag{19}$$

On the other hand, when node k is not sampled, $\theta_{f_k}^2$, $\theta_{m_k}^2$ and $\theta_{s_k}^2$ are kept fixed. In [48], the following choices are suggested for the forgetting factors: $\zeta_f = 1 - \frac{1}{5M}$, $\zeta_m = 1 - \frac{1}{15M}$, and $\zeta_s = 1 - \frac{1}{45M}$. Since $\zeta_f > \zeta_m > \zeta_s$, $\theta_{f_k}^2$ 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, $\theta_{s_k}^2$ 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, $\theta_{m_k}^2$ shows an intermediate behavior [48]. If no change in the environment has been detected recently, i.e., the algorithm is in "normal mode", an intermediate estimate $\theta_{v_k}^2$ is calculated as [48]

$$\theta_{v_k}^2(n) = \zeta_f \theta_{v_k}^2(n-1) + (1-\zeta_f) \theta_{\min_k}^2, \tag{20}$$

where

$$\theta_{\min_k}^2 \triangleq \min\{\theta_{f_k}^2(n),\, \theta_{m_k}^2(n),\, \theta_{s_k}^2(n)\}. \tag{21}$$

Regardless of the current mode, the consolidated estimate $\widehat{\sigma}_{v_k}^2(n)$ is obtained by [48]

$$\widehat{\sigma}_{v_k}^2(n) = \min\{\theta_{v_k}^2(n),\, \theta_{f_k}^2(n)\}. \tag{22}$$

The mechanism of [48] enters "change mode", i.e., it considers that a change in the environment has been detected whenever

$$\theta_{f_k}^2(n) > \theta_{s_k}^2(n), \tag{23}$$

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 $\theta_{f_k}^2(n) < \theta_{m_k}^2(n)$, $\theta_{v_k}^2$ is updated as

$$\theta_{v_k}^2(n) = \zeta_m \theta_{v_k}^2(n-1) + (1-\zeta_m) \theta_{\min_k}^2(n). \tag{24}$$

Otherwise, it is kept fixed until $\theta_{m_k}^2(n) < \theta_{s_k}^2(n)$, upon which the algorithm returns to normal mode [48]. A summary of the algorithm is provided in Table 1 for clarity. The estimate $\widehat{\sigma}_{v_k}^2(n)$ can be calculated locally, since it only uses information available at node k. However, in order to calculate (15), each sampled node i must send $\widehat{\sigma}_{v_i}^2(n)$ to its neighbors. Nonetheless, if this information is sent bundled with $\varepsilon_i^2(n)$ and $\boldsymbol{\psi}_i(n)$, no extra transmissions are required.

Table 1: Summary of the noise power estimation algorithm proposed in [48].

% Initialization – for each node $i=1,\cdots,V$, *set* $\theta^2_{f_i}(-1)\leftarrow 0$, $\theta^2_{m_i}(-1)\leftarrow 0$, $\theta^2_{s_i}(-1)\leftarrow 0, \theta^2_{v_i}(-1)\leftarrow 0$, $\text{flag}_i \leftarrow$ false, $\text{mode}_i \leftarrow$ "normal"

% Then, repeat for every iteration $n\geqslant 0$ *and every node* k:

$\theta^2_{f_k}(n) \leftarrow \zeta_f \theta^2_{f_k}(n-1) + (1-\zeta_f)e^2_k(n)$

$\theta^2_{m_k}(n) \leftarrow \zeta_m \theta^2_{m_k}(n-1) + (1-\zeta_m)e^2_k(n)$

$\theta^2_{s_k}(n) \leftarrow \zeta_s \theta^2_{s_k}(n-1) + (1-\zeta_s)e^2_k(n)$

If mode = "normal", **do**:

  $\theta^2_{v_k}(n) \leftarrow \zeta_m \theta^2_{v_k}(n-1) + (1-\zeta_m)\theta^2_{\min_k}(n)$

  If $\theta^2_{f_k} > \theta^2_{s_k}$, **do**:

    mode $\leftarrow$ "change"

    $\text{flag}_i \leftarrow$ true

  End

Else, do:

  If $\theta^2_{m_k} < \theta^2_{s_k}$, **do**:

    mode $\leftarrow$ "normal"

  Else, do:

    If $\theta^2_{f_k} < \theta^2_{m_k}$ **or** $\text{flag}_i =$ false, **do**:

      $\theta^2_{v_k}(n) \leftarrow \zeta_m \theta^2_{v_k}(n-1) + (1-\zeta_m)\theta^2_{\min_k}(n)$

    End

  End

End

$\widehat{\sigma}^2_{v_k}(n) \leftarrow \min\{\theta^2_{v_k}(n), \theta^2_{f_k}(n)\}$

Since we now have a different $\beta_k(n)$ for each node instead of a global parameter β, we make this replacement in (14), allowing the nodes to have distinct step sizes $\mu_{s_k}(n)$. 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 (14). Using (15), we finally conclude after some algebraic manipulations that we must choose

$$\mu_{s_k}(n) > \frac{1}{\widehat{\sigma}^2_{\mathcal{N}_k}(n)} \left\{ \frac{\alpha^+}{(\gamma-1)(\phi'_0 - \phi'_{\alpha^+})} \left[\left(\frac{\phi'_o}{\phi'_{\alpha^+}} \right)^{\frac{1}{\Delta n}} - 1 \right] \right\} \tag{25}$$

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 mean-squared error (MSE). The term between braces in the right-hand side of (25) is a constant once the filter designer chooses the values for Δn and γ. Thus, the tuning of μ_{s_k} only requires one

extra division per iteration at each node and one extra sum if a regularization δ_c term is added to $\widehat{\sigma}^2_{\mathcal{N}_k}(n)$ as in (4).

Incorporating the algorithm of Table 1 as well as (15) and (25) into AS-dNLMS, we obtain an algorithm where each node k dynamically tunes its own parameters $\beta_k(n)$ and $\mu_{s_k}(n)$. We name the resulting algorithm as Dynamic-Tuning AS-dNLMS (DTAS-dNLMS). For convenience, a pseudocode is presented in Table 2. 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 Table 2. Particularly, if (4) and (5) are considered in conjunction with DTAS-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 $\widehat{\sigma}^2_{kk}$ could tend towards zero in (5) due to $\bar{s}_k$ being equal to zero in (2a). To avoid this, for $j=k$, we replace $\boldsymbol{\psi}_j(n+1)$ in (5) by $\bar{\boldsymbol{\psi}}_k(n+1) \triangleq \bar{s}_k(n)\boldsymbol{\psi}_k(n+1) + [1-\bar{s}_k(n)]\bar{\boldsymbol{\psi}}_k(n)$.

Table 2: Summary of the DTAS-dNLMS algorithm.

	% Initialization – for each node $i=1,\cdots,V$, *set* $\alpha_i(0)\leftarrow\alpha^+$, $\bar{s}_i(0)\leftarrow 1$, $\varepsilon_i(n)\leftarrow 0$, $\mathbf{x}_i(0)=\mathbf{0}$, $\boldsymbol{\psi}_i(0)\leftarrow \mathbf{0}$, $\mathbf{w}_i(0)\leftarrow\mathbf{0}$, $\theta^2_{f_i}(-1)\leftarrow 0$, $\theta^2_{m_i}(-1)\leftarrow 0$, $\theta^2_{s_i}(-1)\leftarrow 0$, $\theta^2_{v_i}(-1)\leftarrow 0$, $\text{flag}_i \leftarrow$ false, $\text{mode}_i \leftarrow$ "normal"
	% Then, repeat for every iteration $n\geqslant 0$ *and every node* k:
	% Adaptation Step
1	**If** $\alpha_k(n)\geqslant 0$, **do** :
2	$\bar{s}_k(n)\leftarrow 1$
3	Update $\mathbf{u}_k(n)$ and $\|\mathbf{u}_k(n)\|^2$
4	$\mu_k(n) \leftarrow \tilde{\mu}_k/[\delta + \|\mathbf{u}_k(n)\|^2]$
5	$e_k(n) \leftarrow d_k(n) - \mathbf{u}_k^{\mathrm{T}}(n)\mathbf{w}_k(n)$
6	$\varepsilon_k(n)\leftarrow e_k(n)$
7	$\boldsymbol{\psi}_k(n+1) \leftarrow \mathbf{w}_k(n) + \bar{s}_k(n)\mu_k(n)\mathbf{u}_k(n)e_k(n)$
8	Run lines 1-19 of the algorithm depicted in Table 1, thus obtaining $\widehat{\sigma}^2_{v_k}(n)$
9	**Else, do**:
10	$\bar{s}_k(n) \leftarrow 0$
11	$\varepsilon_k(n) \leftarrow \varepsilon_k(n-1)$
12	$\widehat{\sigma}^2_{v_k}(n) \leftarrow \widehat{\sigma}^2_{v_k}(n-1)$
13	$\boldsymbol{\psi}_k(n+1) \leftarrow \mathbf{w}_k(n)$
14	**End**
	% Transmission – send $\boldsymbol{\psi}_k$, ε_k^2 *and* $\widehat{\sigma}^2_{v_k}$ *to every node* $\in\mathcal{N}_k$
	% Combination Step
15	$\widehat{\sigma}^2_{\mathcal{N}_k}(n) = \sum_{i\in\mathcal{N}_k} c_{ik}(n)\widehat{\sigma}^2_{v_i}(n)$
16	$\beta_k(n) = \gamma\widehat{\sigma}^2_{\mathcal{N}_k}(n)$
17	$\mu_{s_k}(n) = \kappa/[\delta_c + \widehat{\sigma}^2_{\mathcal{N}_k}(n)]$
18	$\alpha_k(n+1) \leftarrow \alpha_k(n) + \mu_{s_k}(n)\phi'_k(n) \times [\sum_{i\in\mathcal{N}_k} c_{ik}(n)\varepsilon_i^2(n) - \beta_k(n)\bar{s}_k(n)]$
19	$\mathbf{w}_k(n+1)\leftarrow\sum_{j\in\mathcal{N}_k} c_{ik}(n)\boldsymbol{\psi}_i(n+1)$

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. 4 we resume the simulation of Fig. 2, considering Scenario 2 of Sec. 8. 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 $\mu_s = 0.0045$, which were obtained following the same rules that were used in the simulations of Figs. 1 and 2. On the other hand, the curves with star-shaped (★) markers show the results obtained with $\beta = 0.7\sigma^2_{\max}$ and $\mu_s = 0.0025$, which were chosen in order to obtain roughly the same number of nodes sampled per iteration as observed in Fig. 1 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_s is complicated, since the rule for its selection proposed in [43] 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 [43]. For DTAS-dNLMS, we consider $\gamma = 9$ and $\Delta n = 7 \cdot 10^4$. From Fig. 4(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. 4(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. 4(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. 5 we incorporate a reset tool for the sampling mechanism in DTAS-dNLMS, which addresses this issue. Before that, however, we present in Sec. 4 an analysis on the effects of the parameter γ, thus aiding the filter designer in its selection.

4. Selection of the parameter γ

In the simulations of Fig. 4, we adopted $\gamma = 9$ in order to achieve an average of two nodes sampled per iteration in 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 (8). Since β was replaced by $\beta_k(n) = \gamma\hat{\sigma}^2(n)$ in (16), 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

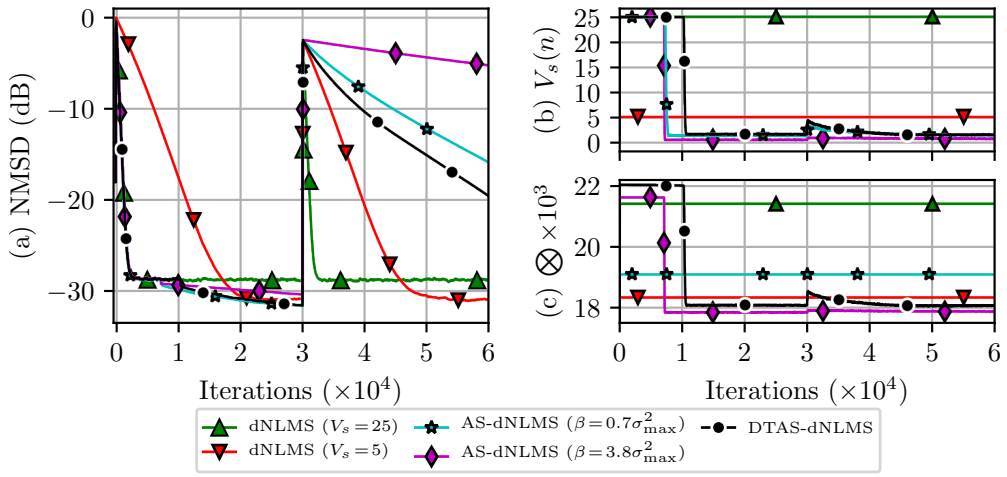

Figure 4: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS and DTAS-dNLMS in Scenario 2 described in Sec. 8, in which one of the nodes is much noisier than the others. (a) NMSD curves, and (b) number of nodes sampled per iteration.

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

$$\mathrm{E}\{V_s\} = \sum_{k=1}^{V} p_{s_k}. \tag{26}$$

Thus, we now seek to obtain an estimate $\widehat{p}_{s_k}$ for p_{s_k}, $k=1,\ \cdots,\ V$. At this point, it is useful to note that the sampling mechanism should exhibit a cyclic behavior in steady state. Ideally, 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_{s_k} by estimating the maximum expected "duty cycle" of the mechanism, i.e.,

$$p_{s_k} \leqslant \widehat{p}_{\max_k} \triangleq \frac{\xi_k}{\xi_k + \overline{\xi}_k}, \tag{27}$$

where ξ_k denotes the maximum expected number of iterations per cycle in which node k is sampled and $\overline{\xi}_k$ 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 ξ_k and $\overline{\xi}_k$. However, the detailed derivation is provided in Appendix A. 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 ξ_k by

$$\xi_k = \xi = \left\lceil \frac{1}{\gamma - 1} \right\rceil, \tag{28}$$

for $k = 1, \cdots, V$, where $\lceil\cdot\rceil$ denotes the ceiling function. We should notice that (28) is inherently greater than one for $\gamma > 1$. Analogously, for $\overline{\xi}_k$, we obtain

$$\overline{\xi}_k = \overline{\xi} = \max\{1, \lfloor \gamma - 1 \rfloor\}, \tag{29}$$

where $\lfloor\cdot\rfloor$ denotes the floor function. It is worth noting that ξ_k and $\overline{\xi}_k$ 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 (28) and (29) in (27) and using (26), we finally obtain the following upper bound:

$$\mathrm{E}\{V_s\} \leqslant V \cdot \frac{\left\lceil \frac{1}{\gamma - 1} \right\rceil}{\left\lceil \frac{1}{\gamma - 1} \right\rceil + \max\{1, \lfloor \gamma - 1 \rfloor\}}. \tag{30}$$

Analyzing (30), 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 expected number of nodes sampled 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 does not depend on the filter length M, the noise variance $\sigma^2_{v_k}$ or the step sizes $\widetilde{\mu}_k$ and μ_{s_k} at any node k, and so on. We should notice that (30) 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 in 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 take a long time. Nonetheless, a certain amount of good sense is usually sufficient to achieve satisfactory results.

5. Resetting the Sampling of the Nodes

The incorporation of the algorithm of [48] in the sampling mechanism in Sec. 3 provides a "reset tool" in the sampling mechanism through Criterion (23). 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 (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 $\widehat{\sigma}_{v_k}^2(n)$, 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

$$\theta_{f_k}^2(n) > \lambda \widehat{\sigma}_{v_k}^2(n) \tag{31}$$

holds for more than M consecutive iterations, where $\lambda > 1$ 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 $\theta_{f_k}^2(n) \leqslant \lambda \widehat{\sigma}_{v_k}^2(n)$ 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 shown in Table 3. The pseudocode presented should be inserted in the combination step of DTAS-dNLMS, between the lines 14 and 15 of Table 2. 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.

Table 3: Summary of the sampling reset mechanism of DTRAS-dNLMS.

	% Initialization – for each node $i = 1, \cdots, V$*, set* $\text{counter}_i \leftarrow 0$
	% Then, repeat for every iteration $n \geqslant 0$ *and every node* k:
1	**If** $\theta_{f_k}^2(n) > \lambda \widehat{\sigma}_{v_k}^2(n)$, **do**:
2	$\text{counter}_k \leftarrow \text{counter}_k + 1$
3	**Else, do**:
4	$\text{counter}_k \leftarrow 0$
5	**If** $\text{counter}_k > M$, **do**:
6	$\alpha_k(n+1) \leftarrow \alpha^+$
7	Go to line 19 of the algorithm in Table 2
8	**Else, do**:
9	Go to line 15 of the algorithm in Table 2

Ideally, λ 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 λ. 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 (31) can be recast as

$$X \triangleq \frac{\theta_{f_k}^2(n)}{\widehat{\sigma}_{v_k}^2(n)} > \lambda, \tag{32}$$

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 λ for which the probability of the Criterion (31) 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. 8, and gradually implemented changes in order to analyze different conditions. The resulting histograms for some of scenarios tested are presented in Fig. 5. In Fig. 5(a), we consider Scenario 1 of Sec. 8 and show the results obtained for node 1. In Fig. 5(b), we also consider Scenario 1, except that the optimal system $\mathbf{w}^{\mathrm{o}}$ is comprised of $M = 10$ coefficients instead of $M = 50$. In Fig. 5(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. 5(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 $\sigma^2_{v_k}$.

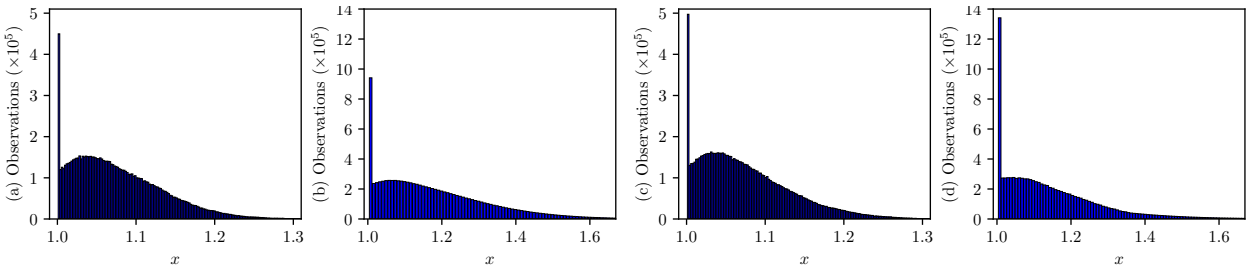

Figure 5: Histograms for X obtained from 100 realizations with $2 \cdot 10^5$ iterations each. Measurements taken in node 1 of the network depicted in Fig. 3. (a) Scenario 1 described in Sec. 8. (b) $M = 10$, noise variance $\sigma^2_{v_k}$ and step sizes $\widetilde{\mu}_k$ as depicted in Fig. 8. (c) $M = 50$, noise variance $\sigma^2_{v_k}$ and step sizes $\widetilde{\mu}_k$ as depicted in Fig. 8 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. 5(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 (22), which imposes that $\theta^2_{f_k}(n) \geqslant \widehat{\sigma}^2_{v_k}(n)$, i.e., $X \geqslant 1$, and enables $\theta^2_{f_k}(n) = \widehat{\sigma}^2_{v_k}(n)$ whenever $\theta^2_{f_k}(n) \leqslant \theta^2_{v_k}(n)$.

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

$$f_X(x) \approx a_3 \delta(x-1) + \frac{a_4}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(x-a_1)^2}{2a_2^2}\right] H(x-1), \tag{33}$$

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 λ such that $\Pr[X>\lambda]<p_\lambda$, with $0<p_\lambda\ll 1$. Thus, we must have

$$1 - F_X(\lambda) < p_\lambda, \tag{34}$$

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

For the sake of brevity, in this section we omit the step-by-step resolution of (34) and skip to the final solution of this inequality. However, a thorough demonstration for this result is provided in Appendix B. It can be shown that (34) is satisfied if we choose

$$\lambda > a_1 + a_2\sqrt{2}\cdot \text{erf}^{-1}\left[\frac{p_\lambda}{1-a_3}\cdot \text{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) + \frac{1-p_\lambda-a_3}{1-a_3}\right], \tag{35}$$

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

Let us examine (35) for two special cases: $p_\lambda = 0$ and $p_\lambda = 1$. The former case corresponds to $\Pr[X>\lambda]<0$, i.e., we should choose a value for λ that X can never surpass. Making the replacement $p_\lambda = 0$ in (35), we obtain $\lambda \to \infty$, which is in accordance with our expectations. On the other hand, the case $p_\lambda = 1$ corresponds to a situation where $\Pr[X>\lambda]<1$, which should hold for any finite value of λ, since X is a continuous random variable. Replacing $p_\lambda = 1$ in (35), we obtain

$$\lambda > a_1 + a_2\sqrt{2}\cdot \text{erf}^{-1}\left\{\frac{1}{1-a_3}\left[\text{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) - a_3\right]\right\}. \tag{36}$$

For the sake of simplicity, let us initially consider the special case $a_3 = 0$ in (36). This corresponds to the case where there is no Dirac delta in the expression for $f_X(x)$ in (33). In this case, (36) yields $\lambda > 1$. This is reasonable, since $X > 1$ always holds. Moreover, for $0 < a_3 \leqslant 1$, (36) yields even lower values for λ, which further supports the validity of the obtained expression. Finally, simulation results suggest that, in stationary environments and in the absence of impulsive noise, $p_\lambda = 5\cdot 10^{-4}$ leads to good results.

In order to successfully apply (35), 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. 6 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 the Model (33) for each value of M using the Nonlinear Least Squares method.

Using the values depicted in Fig. 6 and considering (35) with an equality sign and $p_\lambda = 5\cdot 10^{-4}$, it is possible to plot λ as a function of M, as depicted in Fig. 7. Furthermore, for the sake of simplicity, one could seek to approximate $\lambda(M)$ from the experimental data as an exponential function. Using once again the Nonlinear Least Squares method, the resulting approximation is given by

$$\lambda(M) \approx 1.2326 + 0.8603\exp(-0.0547\cdot M), \tag{37}$$

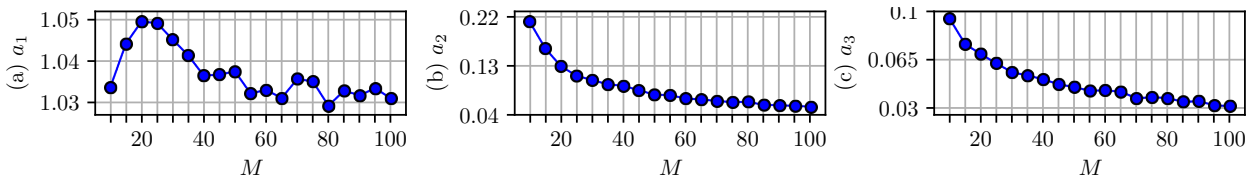

Figure 6: Values fit from the experimental data for (a) a_1, (b) a_2, and (c) a_3 for each filter length $10 \leqslant M \leqslant 100$ considering Model (33) and the Nonlinear Least Squares method.

which is also depicted in Fig. 7. As can be seen from the plot, (37) provides a reasonable approximation for $\lambda(M)$, greatly facilitating the selection of this parameter after the filter length is set. Although we only present the results for $10 \leqslant M \leqslant 100$, (37) holds as an approximation for M outside of this range as well.

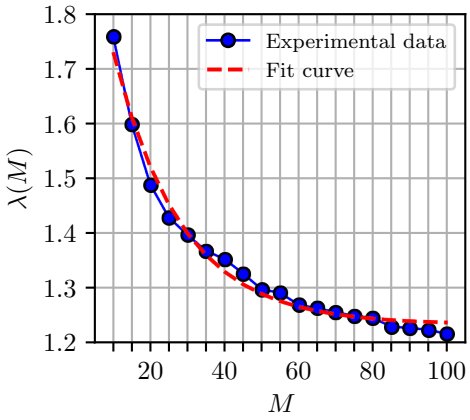

Figure 7: Comparison between the values obtained for λ using (35) with a_1, a_2 and a_3 as depicted in Fig 6 and those yielded by the Approximation (37).

A few remarks should be made about the scenario considered in Fig. 5(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. 6, which were obtained considering Scenario 1 of Sec. 8 with different values for M and white noise as input. However, replacing these values for a_1, a_2 and a_3 in (34), we get $\lambda = 1.622$. This represents an 8.4% error in comparison with the results depicted in Fig. 7, and a 6.7% error in comparison with the value yielded by (37). Thus, despite all the differences between the scenario of Fig. 5(d) and those of Figs. 5(a), (b) and (c), the final value obtained for λ by the method 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 the algorithms depicted in Tables 1, 2, and 3. It enables each node to have its own local parameters β_k and μ_{s_k}, which are tuned at each iteration

according to (15) and (25). 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.

6. 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 Tabs. 1 and 3, 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 [43]. For this purpose, we examine the increase in cost that the modifications proposed in Secs. 3–5 produce. From Tabs. 1 and 2, we can see that the noise estimation algorithm of [48] is only run when node k is sampled, and its cost can be represented by $8\bar{s}_k(n)$ multiplications, $4\bar{s}_k(n)$ sums, and $3\bar{s}_k(n)$ comparisons per node at iteration n. Moreover, from the lines 15 and 16 of Tab. 2, we observe that the computation of $\beta_k(n)$ requires $\bar{s}_k(n)\cdot(|\mathcal{N}_k|+1)$ multiplications and $\bar{s}_k(n)\cdot(|\mathcal{N}_k|-1)$ 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 $\mu_{s_k}(n)$ demands one extra sum and one extra division. Finally, line 1 of Tab. 3 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 $\bar{s}_k(n)\cdot(|\mathcal{N}_k|+9)+1$ more multiplications, $\bar{s}_k(n)\cdot(|\mathcal{N}_k|+3)+2$ more sums, one extra division, and $2+3\bar{s}_k(n)$ more comparisons at each node k and time instant n in the worst-case scenario. These results are summarized in Tab. 4, 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. 4.

Table 4: 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.	$M(3{+}2\lvert\mathcal{N}_k\rvert){+}\lvert\mathcal{N}_k\rvert{+}1$	$\bar{s}_k(n)(3M{+}4){+}2M\lvert\mathcal{N}_k\rvert{+}3\lvert\mathcal{N}_k\rvert{+}1$	$\bar{s}_k(n)(3M{+}13{+}\lvert\mathcal{N}_k\rvert){+}2M\lvert\mathcal{N}_k\rvert{+}3\lvert\mathcal{N}_k\rvert$
Sums	$M(2{+}3\lvert\mathcal{N}_k\rvert){+}2\lvert\mathcal{N}_k\rvert{-}1$	$\bar{s}_k(n)(5M{-}1){+}(\lvert\mathcal{N}_k\rvert{-}1)(3M{-}1)$	$\bar{s}_k(n)(5M{+}2{+}\lvert\mathcal{N}_k\rvert){+}(\lvert\mathcal{N}_k\rvert{-}1)(3M{-}1){+}2$
Div.	$2\lvert\mathcal{N}_k\rvert+1$	$2\lvert\mathcal{N}_k\rvert{+}\bar{s}_k(n)$	$2\lvert\mathcal{N}_k\rvert{+}\bar{s}_k(n)+1$
Compar.	0	3	$5+3\bar{s}_k(n)$

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., $\overline{s}_k(n) = 1$, 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. $\overline{s}_k(n) = 0$, 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. 4 at several iterations. For example, if the reset mechanism is activated, i.e., the condition of line 5 of Tab. 3 holds, $\alpha_k(n+1)$ is set to α^+ and the lines 15–18 of Tab. 2 do not have to be run, which saves some computation.

In comparison with dNLMS, we observe from Tab. 4 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 [17, 18, 49–51].

The expressions depicted in Tab. 4 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 $\overline{s}_k(n)$ with its expected value $\mathrm{E}\{\overline{s}_k\} = p_{sk}$, for which we estimated an upper bound in Sec. 4. 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. 4 that, in steady state,

$$\mathrm{E}\{\Delta\otimes_k\} = 3M-2|\mathcal{N}_k|+1 - p_{s_k}(3M+13+|\mathcal{N}_k|). \tag{38}$$

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 (38) that lower sampling probabilities p_{s_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_{s_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

$$\mathrm{E}\{\Delta\otimes_{\mathrm{global}}\} = V(3M+1) - \sum_{k=1}^{V}\left[2|\mathcal{N}_k| + p_{s_k}(3M+13+|\mathcal{N}_k|)\right]. \tag{39}$$

Since $p_{s_k} \leqslant p_{\max_k} = p_{\max}$ for $k = 1, \cdots, V$, we conclude from (39) that, in order to ensure a reduction in

the number of multiplications in steady state in comparison with dNLMS, i.e., $\mathrm{E}\{\Delta\otimes_{\mathrm{global}}\} > 0$, we must have

$$p_{\max} = \frac{\left\lceil \frac{1}{\gamma - 1} \right\rceil}{\left\lceil \frac{1}{\gamma - 1} \right\rceil + \max\{1, \lfloor \gamma - 1 \rfloor\}} < \frac{V(3M+1) - 2\sum_{k=1}^{V} |\mathcal{N}_k|}{V(3M+13) + \sum_{k=1}^{V} |\mathcal{N}_k|}. \tag{40}$$

Hence, assuming that the network topology is known beforehand, which is a usual practice [1–7], we can use (40) 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

$$M < \frac{1}{3}\left(2 \cdot \frac{1}{V}\sum_{k=1}^{V} |\mathcal{N}_k| - 1\right), \tag{41}$$

DTRAS-dNLMS cannot save computation in comparison with dNLMS in the worst-case scenario. If (41) holds, we must have $p_{\max} < 0$ to ensure a reduction in the computational cost in (40), which is impossible. Nonetheless, we should notice that the condition imposed by (41) 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.

7. 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 ζ_f, ζ_m and ζ_f of the algorithm of [48], which are respectively given by $\zeta_f = 1 - \frac{1}{5M}$, $\zeta_m = 1 - \frac{1}{15M}$ and $\zeta_m = 1 - \frac{1}{45M}$, DTRAS-dNLMS has three other parameters: γ, Δn and λ. 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_s *a priori* according to (14), whereas in DTRAS-dNLMS it is used to adjust each local step size $\mu_{s_k}(n)$ in an online manner, as can be seen in (25). Thus, the only additional parameter that DTRAS-dNLMS has in comparison with AS-dNLMS is λ, which is responsible for tuning the sensitivity of the reset mechanism proposed in Sec. 5. This comparison is summarized in Tab. 5.

The only condition on the parameter γ to ensure that the nodes cease to be sampled is $\gamma > 1$. Furthermore, (30) 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

Table 5: Comparison between the parameters of DTRAS-dNLMS and AS-dNLMS.

Role	AS-dNLMS [43]	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	–	λ

undermines the benefits of the sampling mechanism. This can be attested from (40), which can be used to ensure a reduction in the computational cost of the algorithm. Due to (25) 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 [43] 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. 4, selecting γ such that $\mathrm{E}\{V_s\} \ll 1$ can be problematic. In this case, changes in the environment may go unnoticed for extensive 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 $\mu_{s_k}(n)$, 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

$$\Delta n \approx 2M^2 \cdot \frac{V}{\sum_{k=1}^{V} \widetilde{\mu}_k} \tag{42}$$

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, λ can be set according to (37). The model proposed in Sec. 5 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.

8. Simulation Results

In this section, we present simulation results to showcase the behavior of the proposed 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 $\nu_k = 0.2$ for $k = 1, \cdots, V$ [45], and we use $\delta = 10^{-5}$ and $\delta_c = 10^{-8}$ as regularization factors. Although the results obtained with DTAS-dNLMS in the simulations of Fig. 4 are poor, we still include it in the simulations of this section for the sake of comparison. As a performance indicator, we adopt the NMSD, given by

$$\text{NMSD}(n) = \frac{1}{V}\sum_{k=1}^{V} \text{E}\{\|\mathbf{w}^{\text{o}}(n) - \mathbf{w}_k(n)\|^2\}. \tag{43}$$

8.1. Scenario 1 – Base Scenario

We consider the network of Fig. 3 in the simulations. Furthermore, each node k is subject to a different noise variance $\sigma_{v_k}^2$, as shown in Fig. 8(a), and we consider $\widetilde{\mu}_k \in \{0.1, 1\}$ for each node k, as depicted in Fig. 8(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.

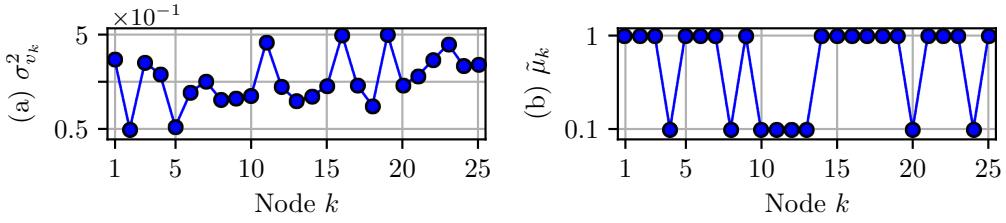

Figure 8: (a) Noise variance $\sigma_{v_k}^2$, and (b) normalized step size $\widetilde{\mu}_k$ for $k = 1, \cdots, V$ considered in the simulations.

Firstly, we resume the simulation of Fig. 1 in order to compare DTRAS-dNLMS with the algorithms previously considered. Its parameter λ was set to 1.298 using the results of Sec. 5 with $p_\lambda = 5 \cdot 10^{-4}$. 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 required for DTRAS-dNLMS. This difference is due to spurious resets of the sampling system, which still occasionally occur in DTRAS-dNLMS even with $p_\lambda = 5 \cdot 10^{-4}$. This will be illustrated in more detail in the sequel. Lastly, the value of μ_s for AS-dNLMS was chosen using (14) with an equality sign, whereas DTAS-dNLMS and DTRAS-dNLMS tune $\mu_{s_k}(n)$ at each iteration using (25). In all cases, $\Delta n = 7000$ was adopted.

In Fig. 9(a) we present the NMSD curves, in Fig. 9(b) the number of nodes sampled and in Fig. 9(c) the number of multiplications per iteration, respectively. As seen in Fig. 1, 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 Sec. 1, the sampling of less nodes leads to a slight reduction in steady-state NMSD. The dNLMS algorithm with $V_s = 5$ 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 nodes sampled, as well as a faster convergence rate. One possible interpretation for this is that although the adaptation step is important for the convergence during the transient and for detecting changes in the environment, in steady state it introduces noise into the network, which the combination step tends to remove [43, 44]. Thus, by reducing the sampling rate during steady state, there may be a slight reduction in NMSD. However, this reduction can be considered marginal in the case of Fig. 9(a). From Figs. 9(b) and 9(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. 6. In comparison with the worst-case scenario depicted in Tab. 4, 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.

In order to verify (30), 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_{s_k} was adopted for $k = 1, \cdots, V$ in order to avoid division by zero in (25). Furthermore, $\lambda = 1.298$ 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. 10. Along with the experimental data, we also present the result yielded by (30) 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 (30). 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 (30) 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 (30) are so

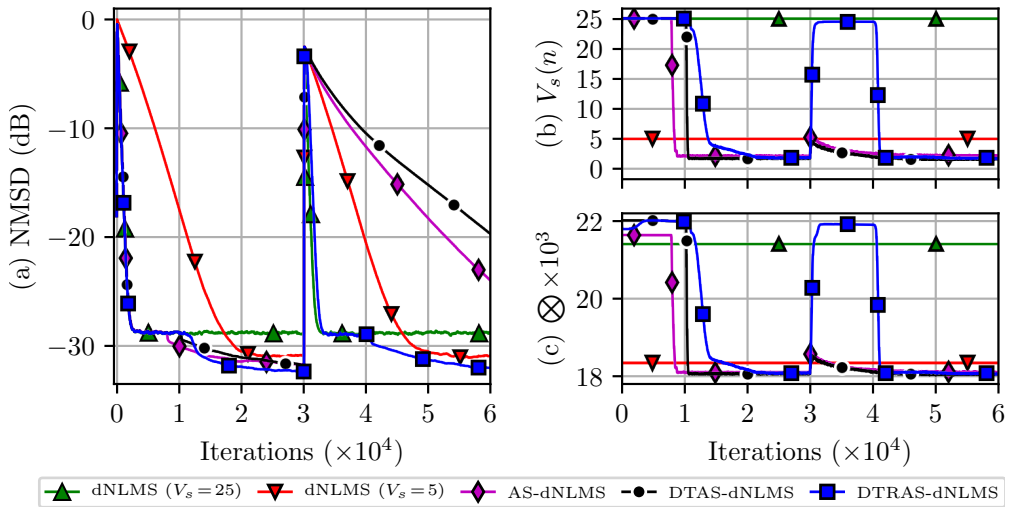

Figure 9: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 1.9$, $\mu_s = 0.0045$), DTAS-dNLMS ($\gamma = 9$), and DTRAS-dNLMS ($\gamma = 11$, $\lambda = 1.298$). For DTAS-dNLMS and DTRAS-dNLMS, $\mu_{s_k}(n)$ was set using (25) with $\Delta n = 7000$. (a) NMSD curves, (b) number of nodes sampled and (c) multiplications per iteration.

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. 10, we can see that (30) 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.

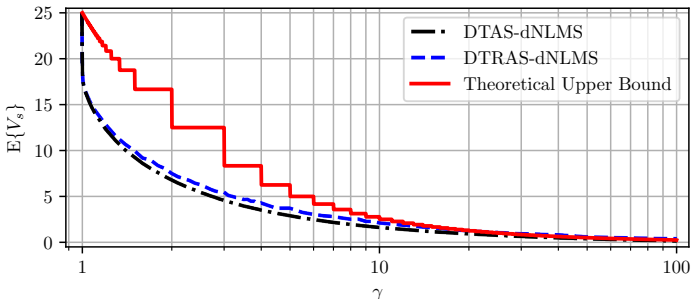

Figure 10: Theoretical results yielded by (30) and average number of nodes sampled by DTAS-dNLMS and DTRAS-dNLMS ($\lambda = 1.298$) as a function of $\gamma \geqslant 1$.

8.2. 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. 8(a) by tenfold.

Hence, in Fig. 11, we resume the simulations of Figs. 2 and 4, with the addition of the DTRAS-dNLMS

algorithm. In Fig. 11(a) we present the NMSD curves, in Fig. 11(b) the number of nodes sampled and in Fig. 11(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. 2 and 4, whereas for DTRAS-dNLMS we consider $\gamma = 11$, $\Delta n = 7 \cdot 10^4$, and $\lambda = 1.298$. 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. 11(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. 11(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.

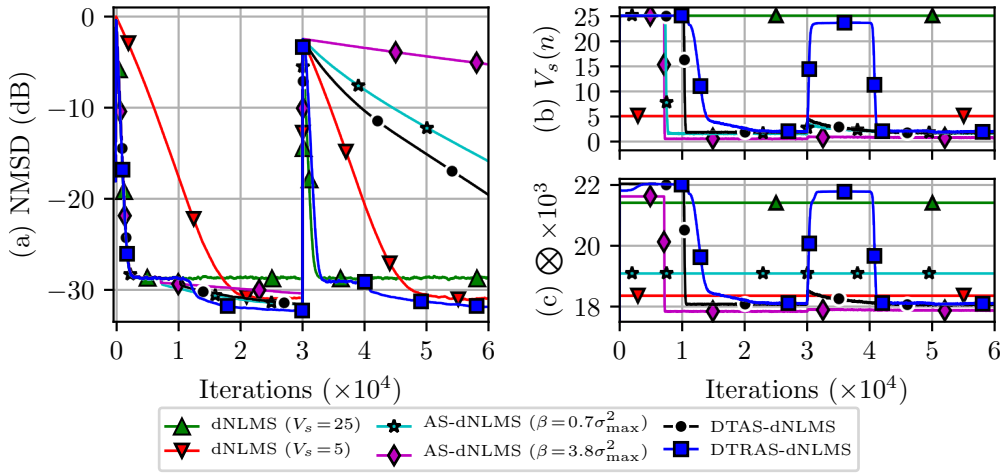

Figure 11: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 19 = 3.8\sigma^2_{\max}$, $\mu_s = 0.0045$, and $\beta = 3.5 = 0.7\sigma^2_{\max}$, $\mu_s = 0.0025$), DTAS-dNLMS ($\gamma = 9$, $\Delta n = 7000$), and DTRAS-dNLMS ($\gamma = 11$, $\lambda = 1.298$, $\Delta n = 7000$) in a scenario where $\sigma^2_{\max}$ is increased by tenfold in comparison with Fig. 8(b). (a) NMSD curves, and (b) number of nodes sampled per iteration.

8.3. Scenario 3 – Random-Walk Tracking

In this section, we investigate the behavior of the proposed algorithms in nonstationary environments following a random-walk model. Starting from Scenario 1, we consider that the optimal solution $\mathbf{w}^{\mathrm{o}}(n)$ varies according to

$$\mathbf{w}^{\mathrm{o}}(n) = \mathbf{w}^{\mathrm{o}}(n-1) + \mathbf{q}(n), \tag{44}$$

where $\mathbf{q}(n)$ is a zero-mean iid $M \times 1$ vector with autocovariance matrix $\mathbf{Q} = \mathrm{E}\{\mathbf{q}(n)\mathbf{q}^{\mathrm{T}}(n)\}$ independent from any other signal [52]. 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. 12, 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 λ: 1.3 and 1.2. The other parameters of the aforementioned algorithms were maintained from the simulations of Fig. 9. In Fig. 12(a), we present the steady-state levels of NMSD, and in Fig. 12(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. 12(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 $\mathrm{Tr}[\mathbf{Q}]$ varies from one solution to another. The dNLMS algorithm with $V_s = 25$ presents a slightly higher steady-state NMSD for $\mathrm{Tr}[\mathbf{Q}] = 10^{-8}$, but a better performance for $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-6}$ 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. 9. However, as $\mathrm{Tr}[\mathbf{Q}]$ 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. 12(a) and 12(b), we observe that the more nodes are sampled, the better the performances of the algorithms for $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-6}$. Furthermore, dNLMS with $V_s = 2$ and DTAS-dNLMS present similar results and the highest NMSD for higher values of $\mathrm{Tr}[\mathbf{Q}]$. As for AS-dNLMS and DTRAS-dNLMS with $\lambda = 1.3$, we observe that their performances are similar to those of DTAS-dNLMS for $\mathrm{Tr}[\mathbf{Q}] \leqslant 10^{-6}$, but are superior for $\mathrm{Tr}[\mathbf{Q}] = 10^{-5}$ and $\mathrm{Tr}[\mathbf{Q}] = 10^{-4}$. For $\mathrm{Tr}[\mathbf{Q}] = 10^{-4}$, DTRAS-dNLMS with $\lambda = 1.3$ performs noticeably better than AS-dNLMS. By changing the value of λ, we can control the behavior of the proposed algorithm, since for $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-7}$ DTAS-dNLMS with $\lambda = 1.2$ performs better than AS-dNLMS and DTAS-dNLMS, as well as dNLMS with $V_s = 2$ and DTRAS-dNLMS. From Fig. 12(b) we observe that the DTRAS-dNLMS with $\lambda = 1.2$ 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.

The sampling mechanism of DTRAS-dNLMS algorithm presents a cyclic behavior in the case of $\mathrm{Tr}[\mathbf{Q}] = 10^{-4}$. In Figs. 13(a) and 13(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 $\lambda = 1.3$ 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 $\mathrm{Tr}[\mathbf{Q}] = 10^{-4}$, 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

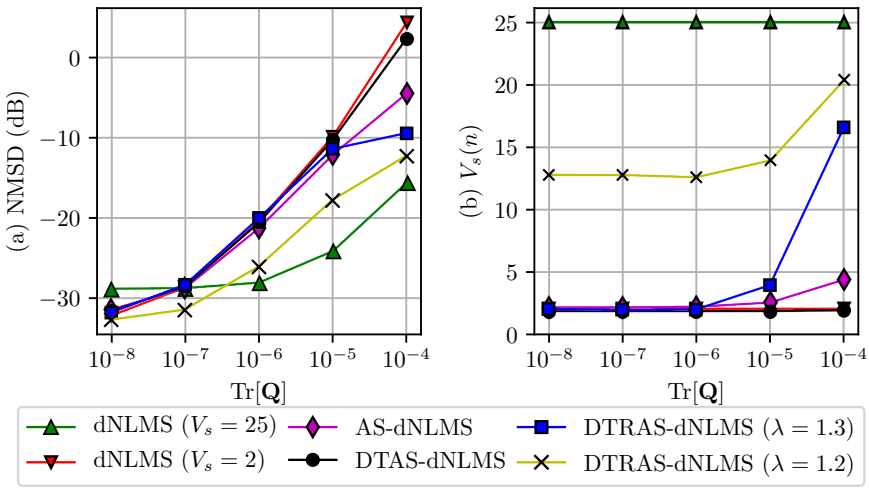

Figure 12: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 1.9$, $\mu_s = 0.0045$), DTAS-dNLMS ($\gamma = 9$, $\Delta n = 7000$), and DTRAS-dNLMS ($\gamma = 11$, $\Delta n = 7000$, and different values for λ) in a nonstationary environment following Model (44). (a) steady-state NMSD, and (b) average number of nodes sampled per iteration.

to a reduction in the number of nodes sampled once again due to (9). Hence, the oscillations in the number of sampled nodes arise. Over time, they stabilize, but do not die out. From Figs. 13(a) and 13(b), we can see that the DTRAS-dNLMS algorithm with $\lambda = 1.2$ also presents fluctuations, but these are much slighter in comparison. The adoption of a lower value for λ makes the reset system activate much more easily, which reduces the impact of oscillations in the error magnitude on the number of nodes sampled.

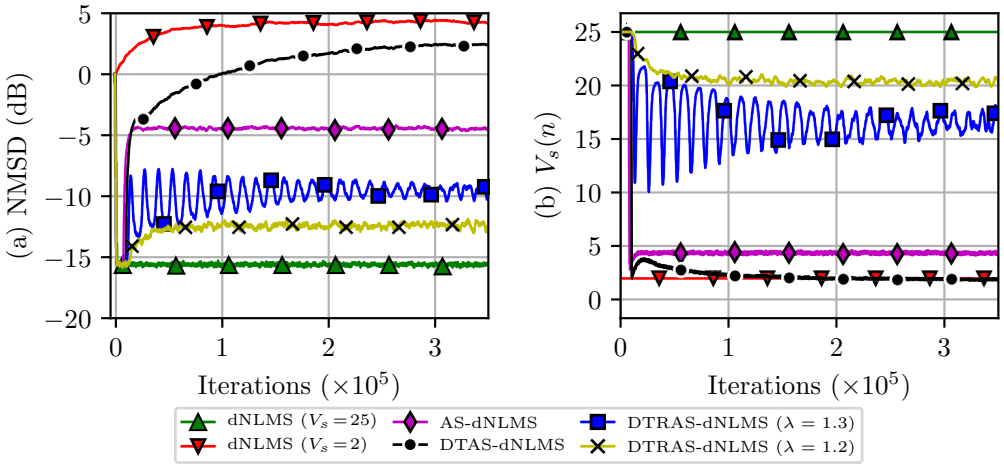

Figure 13: Comparison between dNLMS with V_s nodes randomly sampled per iteration, AS-dNLMS ($\beta = 1.9$, $\mu_s = 0.0045$), DTAS-dNLMS ($\gamma = 9$, $\Delta n = 7000$), and DTRAS-dNLMS ($\gamma = 11$, $\lambda = 1.298$, $\Delta n = 7000$) in a scenario with random-walk tracking as in (44) with $\mathrm{Tr}[\mathbf{Q}] = 10^{-4}$. (a) NMSD curves, and (b) number of nodes sampled per iteration.

Finally, in the simulations of Fig. 14, we repeat the experiments of Fig. 12 considering only the DTRAS

algorithm with $1.15 \leqslant \lambda \leqslant 1.45$. The lower the value of λ, the more nodes are sampled for all values of $\mathrm{Tr}[\mathbf{Q}]$, as expected. By selecting $\lambda = 1.15$, the reset mechanism maintained the sampling of all nodes. Moreover, for $\lambda \geqslant 1.3$, 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 $\lambda \geqslant 1.3$ 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 λ 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 $1.2 \leqslant \lambda \leqslant 1.25$, the reset mechanism activates for all values of $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-8}$. This is especially noticeable for $\lambda = 1.2$, and, similarly to what was observed in Fig. 12, leads to more nodes sampled per iteration and a lower steady-state NMSD for $\mathrm{Tr}[\mathbf{Q}] \geqslant 10^{-8}$.

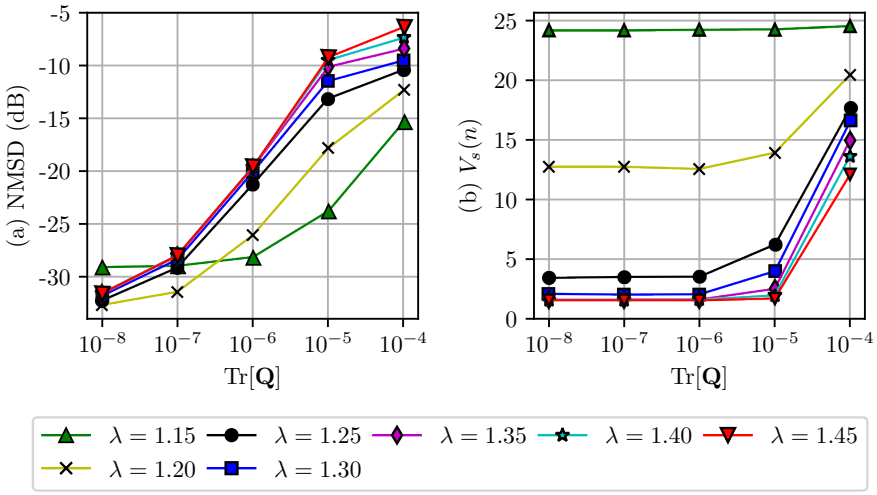

Figure 14: Simulation results obtained in a nonstationary environment following Model (44) with DTRAS-dNLMS ($\gamma = 11$, $\Delta n = 7000$ and different values for λ). (a) Steady-state NMSD, and (b) Number of nodes sampled per iteration.

In the presence of impulsive noise, higher values for λ 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 λ grants the DTRAS-dNLMS algorithm a great degree of flexibility, which makes it suitable for different scenarios and applications.

8.4. Scenario 4 – Colored Input and diffusion Affine Projection Algorithm

In the simulations of this section, we consider a colored input signal $u_k(n)$ at each node k, given by

$$u_k(n) = r_k(n) - 0.8u_k(n-1), \tag{45}$$

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. 8.1, the coefficients of the optimal system $\mathbf{w}_o$ were randomly generated following a uniform distribution in the range $[-1, 1]$, and then normalized to ensure that $\mathbf{w}^o$ has unit norm. In the middle of each realization, we multiply $\mathbf{w}^o$ by 0.25 to simulate an abrupt change in the environment. The noise variances and step sizes are the same as in Fig. 8.

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 the diffusion Affine Projections Algorithm (dAPA) [7]. Its adaptation step is given by

$$\boldsymbol{\psi}_k(n+1) = \mathbf{w}_k(n) + \widetilde{\mu}_k \mathbf{U}_k(n)[\delta \mathbf{I} + \mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)]^{-1}\mathbf{e}_k(n), \tag{46}$$

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 [7, 52]. The adoption of greater values for L usually increases the convergence speed, but also deteriorates its steady-state performance [7]. If $L = 1$ is chosen, the algorithm coincides with dNLMS. Finally, the combination step of dAPA is also given by (2b) [7].

To incorporate the sampling mechanisms into dAPA, we introduce the binary sampling variable $\bar{s}_k(n)$ in the correction term of (46) analogously to (2a). Hence, if $\bar{s}_k(n) = 0$, $\mathbf{U}_k^{\mathrm{T}}(n)\mathbf{w}_k(n)$, $\mathbf{e}_k(n)$, $\mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)$, $[\delta \mathbf{I} + \mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)]^{-1}$, and $\widetilde{\mu}_k \mathbf{U}_k(n)[\delta \mathbf{I} + \mathbf{U}_k^{\mathrm{T}}(n)\mathbf{U}_k(n)]^{-1}\mathbf{e}_k(n)$ do not have to be calculated. If $\bar{s}_k(n) = 1$, (46) is computed as usual. Unlike in the dNLMS adaptation, we consider that the desired signal $d_k(n)$ is sampled even if $\bar{s}_k(n) = 0$, 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 (3) for the adaptation of the sampling mechanisms, rather than the error vector $\mathbf{e}_k(n)$.

In Fig. 15 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. 15 (a) we show the NMSD curves, in Fig. 15(b) the number of nodes sampled and in Fig. 15(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. 15 (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. 9. We adopted $\beta = 4\sigma_{\max}^2 = 2$ and $\mu_s = 0.0098$ for AS-dAPA, $\gamma = 9.5$ and $\Delta n = 10^3$ for DTAS-dAPA, and $\gamma = 9.5$, $\Delta n = 500$, and $\lambda = 1.2328$ for DTAS-dAPA. The value for λ was obtained by replacing $M = 150$ in (37). 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. 15 (a) that DTRAS-dAPA converges to a steady-state NMSD of -30dB 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. 15 (a) and (b). On the other hand, it resumes to an NMSD level of -30dB 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. 15 (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. 9 (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 $L > 1$.

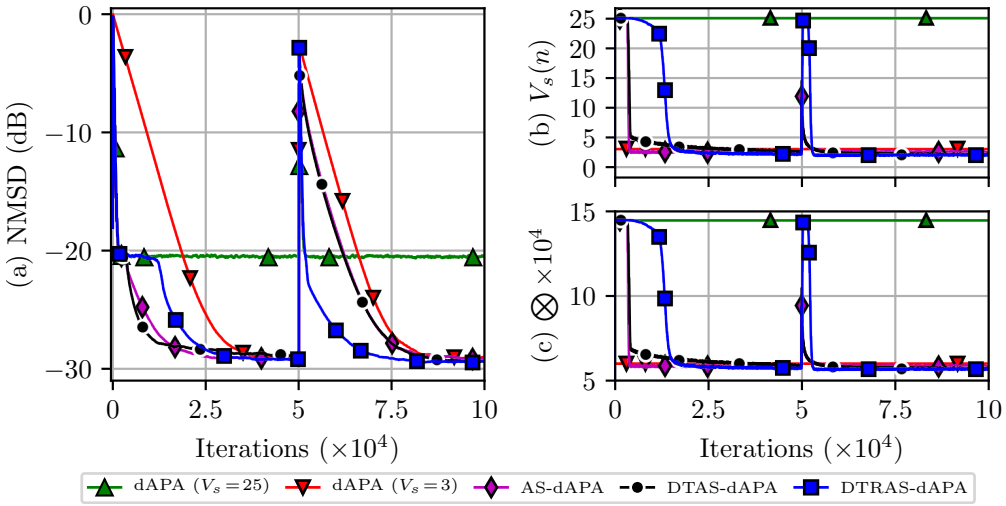

Figure 15: Comparison between dAPA with V_s nodes randomly sampled per iteration, AS-dAPA ($\beta = 2$, $\mu_s = 0.0098$), DTAS-dAPA ($\gamma = 9.5$), and DTRAS-dAPA ($\gamma = 9.5$, $\lambda = 1.2328$). For DTAS-dAPA and DTRAS-dAPA, $\mu_{s_k}(n)$ was set using (25) with $\Delta n = 1000$ and $\Delta n = 500$, respectively. (a) NMSD curves, (b) number of nodes sampled and (c) multiplications per iteration.

9. Conclusions

In this paper, building from our previous works, we proposed an algorithm for adaptive sampling over diffusion networks. The DTRAS-dNLMS algorithm addresses the main limitations of the previously proposed AS-dNLMS. It eliminates the need for *a priori* knowledge of the maximum noise variance in the network, features increased robustness to the presence of noisy nodes, and presents better tracking capabilities than AS-dNLMS, which was one of the main weakness of the latter [43]. In addition to the simulations results, we

also derived analytic expressions that aid in the selection of the parameters of DTRAS-dNLMS, and which were validated by our experiments.

For future work, we intend to obtain more theoretical results for DTRAS-dNLMS, e.g., by presenting convergence and steady-state NMSD analysis, and to test it in other scenarios, possibly using real-world data. Furthermore, we intend to investigate its behavior in environments with impulsive noise and analyze how this would affect the choice of the parameter λ. It should be noted that the AS-dNLMS algorithm also has a counterpart that can be used for censoring over diffusion networks, named ASC-dNLMS algorithm [43]. Although in this paper we focused on the role of DTRAS-dNLMS algorithm for sampling, it could be straightforwardly extended to operate as a censoring mechanism as well, which is yet another topic of interest for future research. Finally, we also intend to run computer simulations comparing the proposed algorithm to other state-of-the-art solutions for sampling and censoring in the future.

Acknowledgment

This work was supported in part by CAPES under Grant 88887.512247/2020-00 and Finance Code 001 and in part by the São Paulo Research Foundation (FAPESP) under Grant 2021/02063-6.

Appendix A. Deriving (28) and (29)

In order to estimate ξ_k and $\overline{\xi}_k$, 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 (16). Simulation results suggest that this approximation is reasonable. Since $\alpha_k \geqslant 0$ and $\bar{s}_k = 1$ when node k is sampled, taking expectations from both sides of (16) in this case yields

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)\geqslant 0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_s \mathrm{E}\{\phi'_k(n)\} \times \left[\sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{\varepsilon_i^2(n)\} - \gamma \mathrm{E}\{\widehat{\sigma}^2_{\mathcal{N}_k}(n)\}\right]. \tag{A.1}$$

In contrast, since $\alpha_k < 0$ and $\bar{s}_k = 0$ when node k is not sampled, we conclude from (16) that, in this case,

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)<0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_s \mathrm{E}\{\phi'_k(n)\} \times \sum_{i\in\mathcal{N}_k} c_{ik}\mathrm{E}\{\varepsilon_i^2(n)\}. \tag{A.2}$$

Since α_k keeps oscillating around the point $\alpha_k = 0$ in steady state, we replace ϕ'_k in (A.1) and (A.2) with its first-order Taylor expansion around $\alpha_k = 0$, which is equal to the constant ϕ'_0. Using (12) and making these replacements in (A.1) and (A.2), we then obtain

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)\geqslant 0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_s \phi'_0 \sigma^2_{\mathcal{N}_k}(1-\gamma) \tag{A.3}$$

and

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n)<0\} = \mathrm{E}\{\alpha_k(n)\} + \mu_s \phi'_0 \sigma^2_{\mathcal{N}_k}. \tag{A.4}$$

Defining $\Delta\alpha_k(n) = \alpha_k(n+1) - \alpha_k(n)$, we get

$$\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n) \geqslant 0\} = -\mu_s\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1), \tag{A.5}$$

where we have rearranged the expression since $1-\gamma < 0$, and

$$\mathrm{E}\{\Delta\alpha_k(n)|\alpha_k(n) < 0\} = \mu_s\phi_0'\sigma_{\mathcal{N}_k}^2. \tag{A.6}$$

Analyzing (A.3) to (A.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 (A.3) and (A.5) that

$$\mathrm{E}\{\alpha_k(n+1)|\alpha_k(n) = 0_+\} = -\mu_s\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1). \tag{A.7}$$

Thus, we observe that $\alpha_k(n+1) < 0$. On the other hand, from (A.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, (A.7) provides the minimum value that α_k can achieve in the mean during steady state, i.e. $\mathrm{E}\{\alpha_{k_{\min}}^{\text{s.s.}}\} = -\mu_s\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1)$.

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 (A.4) and (A.6) that

$$\mathrm{E}\{\alpha_{k_{\max}}^{\text{s.s.}}\} = \mathrm{E}\{\alpha_k(n+1)|\alpha_k(n) = 0_-\} = \mu_s\phi_0'\sigma_{\mathcal{N}_k}^2 \tag{A.8}$$

is the maximum value α_k can assume in the mean during steady state.

Hence, in order to estimate the expected number $\bar{\xi}_k$ 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 (A.5), we thus obtain

$$\xi_k = \frac{\mu_s\phi_0'\sigma_{\mathcal{N}_k}^2}{\mu_s\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1)} = \frac{1}{\gamma-1}. \tag{A.9}$$

In order to obtain $\bar{\xi}_k$, we follow an analogous procedure for $\mathrm{E}\{\alpha_{k_{\min}}^{\text{s.s.}}\}$. Taking (A.6) into account, we arrive at

$$\bar{\xi}_k = \frac{\mu_s\phi_0'\sigma_{\mathcal{N}_k}^2(\gamma-1)}{\mu_s\phi_0'\sigma_{\mathcal{N}_k}^2} = \gamma - 1. \tag{A.10}$$

However, taking into account the fact that ξ_k and $\bar{\xi}_k$ represent a certain number of iterations, we should expect them to be natural numbers. Since we are interested in the maximum value that ξ_k can assume, we replace it by its ceiling. Analogously, since we seek the minimum value that $\bar{\xi}_k$ can assume, we replace it by its floor. Moreover, taking into account that ξ_k and $\bar{\xi}_k$ should be greater than or equal to one, we arrive at (28) and (29).

Appendix B. Obtaining λ from p_λ

Analyzing (33) and using the definition of cumulative distribution function, we conclude that, for $x > 1$,

$$F_X(x) = a_3 + \int_1^x \frac{a_4}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(\rho - a_1)^2}{2a_2^2}\right] d\rho. \tag{B.1}$$

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.,

$$g_X(x) = \frac{1}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(x - a_1)^2}{2a_2^2}\right], \tag{B.2}$$

and its cdf by

$$G_X(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi a_2^2}} \exp\left[\frac{-(\rho - a_1)^2}{2a_2^2}\right] d\rho, \tag{B.3}$$

we may recast (B.1) as

$$F_X(x) = a_3 + a_4 \left[G_X(x) - G_X(1)\right]. \tag{B.4}$$

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 (B.4) for $x \to \infty$, we get $1 = a_3 + a_4 \left[1 - G_X(1)\right]$, from which we conclude that

$$a_4 = \frac{1 - a_3}{1 - G_X(1)}. \tag{B.5}$$

Thus, (B.1) can be recast as

$$F_X(x) = a_3 + \frac{a_3 - 1}{2\left[1 - G_X(1)\right]} \left[G_X(x) - G_X(1)\right]. \tag{B.6}$$

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 (B.6) as

$$F_X(x) = \; a_3 + \frac{(1 - a_3)}{1 - \text{erf}\left(\frac{1-a_1}{a_2\sqrt{2}}\right)} \left[\text{erf}\left(\frac{x - a_1}{a_2\sqrt{2}}\right) - \text{erf}\left(\frac{1 - a_1}{a_2\sqrt{2}}\right)\right]. \tag{B.7}$$

Replacing (B.7) in (34) and making $x = \lambda$, we finally get

$$\text{erf}\left(\frac{\lambda - a_1}{a_2\sqrt{2}}\right) > \frac{1 - p_\lambda - a_3}{1 - a_3} \left[1 - \text{erf}\left(\frac{1 - a_1}{a_2\sqrt{2}}\right)\right] + \text{erf}\left(\frac{1 - a_1}{a_2\sqrt{2}}\right) = \frac{p_\lambda}{1 - a_3} \cdot \text{erf}\left(\frac{1 - a_1}{a_2\sqrt{2}}\right) + \frac{1 - p_\lambda - a_3}{1 - a_3}. \tag{B.8}$$

It should be noted that if

$$\frac{p_\lambda}{1 - a_3} \cdot \text{erf}\left(\frac{1 - a_1}{a_2\sqrt{2}}\right) + \frac{1 - p_\lambda - a_3}{1 - a_3} = \frac{1 - a_3 - p_\lambda \text{erfc}\left(\frac{1-a_1}{a_2\sqrt{2}}\right)}{1 - a_3} \leqslant -1, \tag{B.9}$$

where erfc denotes the complementary error function, any value of λ satisfies (B.8). Moreover, there is always a solution to (B.8). The only case in which this would not happen is if

$$\frac{1-a_3-p_\lambda \text{erfc}\left(\frac{1-a_1}{a_2\sqrt{2}}\right)}{1-a_3} > 1, \tag{B.10}$$

i.e.,

$$p_\lambda \text{erfc}\left(\frac{1-a_1}{a_2\sqrt{2}}\right) < 0, \tag{B.11}$$

which is impossible since $p_\lambda \geqslant 0$ and $\text{erfc}(x) > 0,\ \forall x$. Thus, assuming that (B.9) does not hold, (35) can be straightforwardly obtained from (B.8). If (B.9) does hold, then any choice for λ is equally fitting.

References

[1] A. H. Sayed, *Adaptation, Learning, and Optimization over Networks*, vol. 7, Foundations and Trends in Machine Learning, now Publishers Inc., Hanover, MA, 2014.

[2] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, Jul. 2008.

[3] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, Mar. 2009.

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

[5] Z. Zheng, Z. Liu, and M. Huang, "Diffusion least mean square/fourth algorithm for distributed estimation," *Signal Processing*, vol. 134, pp. 268–274, May 2017.

[6] L. Lu and H. Zhao, "Diffusion leaky LMS algorithm: Analysis and implementation," *Signal processing*, vol. 140, pp. 77–86, Nov. 2017.

[7] M. S. E. Abadi and M. S. Shafiee, "Distributed estimation over an adaptive diffusion network based on the family of affine projection algorithms," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 2, pp. 234–247, Jun. 2019.

[8] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4129–4144, Aug. 2014.

[9] P. Di Lorenzo, S. Barbarossa, and A. H. Sayed, "Distributed spectrum estimation for small cell networks based on sparse diffusion adaptation," *IEEE Signal Processing Letters*, vol. 20, no. 12, pp. 1261–1265, Dec. 2013.

[10] A. Hajihoseini and S. A. Ghorashi, "Distributed spectrum sensing for cognitive radio sensor networks using diffusion adaptation," *IEEE Sensors Letters*, vol. 1, no. 5, pp. 1–4, Oct. 2017.

[11] W. Xia, G, Xia, and J, Li, "Collaborative beamforming via diffusion adaptation based on tensor over array networks," *Digital Signal Processing*, vol. 106, pp. 102825, Nov. 2020.

[12] F. S. Cattivelli and A. H. Sayed, "Modeling bird flight formations using diffusion adaptation," *IEEE Transactions on Signal Processing*, vol. 59, pp. 2038–2051, May 2011.

[13] F. Chen, L. Hu, P. Liu, and M. Feng, "A robust diffusion estimation algorithm for asynchronous networks in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 9103–9115, Sept. 2020.

[14] J. Fernandez-Bes, , J. Arenas-García, M. T. M. Silva, and L. A. Azpicueta-Ruiz, "Adaptive diffusion schemes for heterogeneous networks," *IEEE Transactions on Signal Processing*, vol. 65, pp. 5661–5674, Nov. 2017.

[15] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4129–4133.

[16] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, "A preconditioned graph diffusion LMS for adaptive graph signal processing," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 111–115.

[17] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, "Adaptive graph signal processing: Algorithms and optimal sampling strategies," *IEEE Transactions on Signal Processing*, vol. 66, no. 13, pp. 3584–3598, Jul. 2018.

[18] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, "Distributed adaptive learning of graph signals," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4193–4208, Aug. 2017.

[19] N. Takahashi and I. Yamada, "Link probability control for probabilistic diffusion least-mean squares over resource-constrained networks," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2010, pp. 3518–3521.

[20] R. Arroyo-Valles, S. Maleki, and G. Leus, "A censoring strategy for decentralized estimation in energy-constrained adaptive diffusion networks," in *2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2013, pp. 155–159.

[21] J. Fernandez-Bes, R. Arroyo-Valles, J. Arenas-García, and J. Cid-Sueiro, "Censoring diffusion for harvesting WSNs," in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2015, pp. 237–240.

[22] R. Arablouei, S. Werner, Y.-F. Huang, and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Transactions on Signal Processing*, vol. 62, no. 2, pp. 472–484, Jan. 2014.

[23] S. Xu, R. C. De Lamare, and H. V. Poor, "Dynamic topology adaptation for distributed estimation in smart grids," in *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2013, pp. 420–423.

[24] S. Chouvardas, K. Slavakis, and S. Theodoridis, "Trading off complexity with communication costs in distributed adaptive learning via krylov subspaces for dimensionality reduction," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 257–273, Apr. 2013.

[25] M. O. Sayin and S. S. Kozat, "Compressive diffusion strategies over distributed networks for reduced communication load," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5308–5323, Oct. 2014.

[26] S. Xu, R. C. De Lamare, and H. V. Poor, "Distributed compressed estimation based on compressive sensing," *IEEE Signal Processing Letters*, vol. 22, no. 9, pp. 1311–1315, Sept. 2015.

[27] S. Gupta, A. K. Sahoo, and U. K. Sahoo, "Partial diffusion over distributed networks to reduce inter-node communication," in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2017, pp. 1–6.

[28] I. E. K. Harrane, R. Flamary, and C. Richard, "On reducing the communication cost of the diffusion LMS algorithm," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 100–112, Mar. 2018.

[29] C. G. Lopes and A. H. Sayed, "Diffusion adaptive networks with changing topologies," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2008, pp. 3285–3288.

[30] S. Werner, Y.-F. Huang, M. L. R. De Campos, and V. Koivunen, "Distributed parameter estimation with selective cooperation," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2009, pp. 2849–2852.

[31] X. Zhao and A. H. Sayed, "Single-link diffusion strategies over adaptive networks," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 3749–3752.

[32] S. Xu, R. C. de Lamare, and H. V. Poor, "Adaptive link selection algorithms for distributed estimation," *EURASIP Journal on Advances in Signal Processing*, vol. 2015, no. 1, pp. 86, 2015.

[33] R. Arablouei, S. Werner, K. Doğançay, and Y.-F. Huang, "Analysis of a reduced-communication diffusion LMS algorithm," *Signal Processing*, vol. 117, pp. 355–361, Dec. 2015.

[34] F. Chen and X. Shao, "Broken-motifs diffusion LMS algorithm for reducing communication load," *Signal Processing*, vol. 133, pp. 213–218, Apr. 2017.

[35] A. Rastegarnia, "Reduced-communication diffusion RLS for distributed estimation over multi-agent networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 177–181, Jan. 2020.

[36] O. N. Gharehshiran, V. Krishnamurthy, and G. Yin, "Distributed energy-aware diffusion least mean squares: Game-theoretic learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 5, pp. 821–836, Oct. 2013.

[37] D. K. Berberidis, V. Kekatos, G. Wang, and G. B. Giannakis, "Adaptive censoring for large-scale regressions," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5475–5479.

[38] C.-K. Yu, M. Van Der Schaar, and A. H. Sayed, "Information-sharing over adaptive networks with self-interested agents," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 1, pp. 2–19, Mar. 2015.

[39] Z. Wang, Z. Yu, Q. Ling, D. Berberidis, and G. B. Giannakis, "Distributed recursive least-squares with data-adaptive censoring," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5860–5864.

[40] Z. Wang, Z. Yu, Q. Ling, D. Berberidis, and G. B. Giannakis, "Decentralized RLS with data-adaptive censoring for regressions over large-scale networks," *IEEE Transactions on Signal Processing*, vol. 66, no. 6, pp. 1634–1648, Mar. 2018.

[41] L. Yang, H. Zhu, K. Kang, X. Luo, H. Qian, and Y. Yang, "Distributed censoring with energy constraint in wireless sensor networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6428–6432.

[42] L. Yang, H. Zhu, H. Wang, K. Kang, and H. Qian, "Data censoring with network lifetime constraint in wireless sensor networks," *Digital Signal Processing*, vol. 92, pp. 73–81, Sept. 2019.

[43] 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.

[44] J.-W. Lee, J.-T. Kong, W.-J. Song, and S.-E. Kim, "Data-reserved periodic diffusion LMS with low communication cost over networks," *IEEE Access*, vol. 6, pp. 54636–54650, Sep. 2018.

[45] S.-Y. Tu and A. H. Sayed, "Optimal combination rules for adaptation and learning over networks," in *2011 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2011, pp. 317–320.

[46] J. Arenas-Garcia, L. A. Azpicueta-Ruiz, M. T. M. Silva, V. H. Nascimento, and A. H. Sayed, "Combinations of adaptive filters: Performance and convergence properties," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 120–140, Jan. 2016.

[47] M. Lázaro-Gredilla, L. A. Azpicueta-Ruiz, A. R. Figueiras-Vidal, and J. Arenas-Garcia, "Adaptively biasing the weights of adaptive filters," *IEEE Trans. Signal Process.*, vol. 58, no. 7, pp. 3890–3895, Jul. 2010.

[48] T. Strutz, "Estimation of measurement-noise variance for variable-step-size NLMS filters," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 675–679.

[49] S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic, "Discrete signal processing on graphs: Sampling theory," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510–6523, Dec. 2015.

[50] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, Jul. 2016.

[51] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Transactions on Signal Processing*, vol. 64, no. 18, pp. 4845–4860, May 2016.

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

Daniel G. Tiglea received the B.S. degree in 2018 and the M.S. degree in 2020, both in Electrical

Engineering from Escola Politécnica, Universidade de São Paulo, Brazil. Since August 2020, he has been pursuing his PhD at the same institution. His research interests include linear and nonlinear adaptive filtering, and distributed signal processing.

Renato Candido received the B.S. degree in 2006 from Universidade Presbiteriana Mackenzie, São Paulo, Brazil and the M.S. and Ph.D. degrees in 2009 and 2014 from Escola Politécnica, Universidade de São Paulo, Brazil, all in Electrical Engineering. From 2015 to 2017, he worked as a Postdoctoral Researcher at the Department of Electronic Systems Engineering, Escola Politéncia, Universidade de São Paulo and currently he collaborates as a researcher at the same university. His research interests include signal processing, adaptive filtering, and machine learning.

Magno T. M. Silva (M'05) received the B.S. degree in 1999, the M.S. degree in 2001, and the Ph.D. degree in 2005, all in Electrical Engineering from Escola Politécnica, Universidade de São Paulo, Brazil. Since August 2006, he has been with the Department of Electronic Systems Engineering at Escola Politécnica, Universidade de São Paulo, where he is currently an Associate Professor. From January to July 2012, he worked as a Postdoctoral Researcher at Universidad Carlos III de Madrid, Leganés, Spain. From 2015 to 2018, he served as Associate Editor for the IEEE Signal Processing Letters. His research interests include linear and nonlinear adaptive filtering, distributed estimation, and machine learning for signal processing.