RESEARCH

Adaptive Link Selection Algorithms for Distributed Estimation

Songcen Xu[1*], Rodrigo C. de Lamare[1,2] and H. Vincent Poor[3]

[*]Correspondence:
songcen.xu@york.ac.uk
[1]Department of Electronics, University of York, YO10 5DD York, UK
Full list of author information is available at the end of the article

Abstract

This paper presents adaptive link selection algorithms for distributed estimation and considers their application to wireless sensor networks and smart grids. In particular, exhaustive search–based least–mean–squares (LMS) / recursive least squares (RLS) link selection algorithms and sparsity–inspired LMS / RLS link selection algorithms that can exploit the topology of networks with poor–quality links are considered. The proposed link selection algorithms are then analyzed in terms of their stability, steady–state and tracking performance, and computational complexity. In comparison with existing centralized or distributed estimation strategies, key features of the proposed algorithms are: 1) more accurate estimates and faster convergence speed can be obtained; and 2) the network is equipped with the ability of link selection that can circumvent link failures and improve the estimation performance. The performance of the proposed algorithms for distributed estimation is illustrated via simulations in applications of wireless sensor networks and smart grids.

Keywords: Adaptive link selection; distributed estimation; wireless sensor networks; smart grids

Introduction

Distributed signal processing algorithms have become a key approach for statistical inference in wireless networks and applications such as wireless sensor networks and smart grids [1, 2, 3, 4, 5]. It is well known that distributed processing techniques deal with the extraction of information from data collected at nodes that are distributed over a geographic area [1]. In this context, for each specific node, a set of neighbor nodes collect their local information and transmit the estimates to a specific node. Then, each specific node combines the collected information together with its local estimate to generate an improved estimate.

Prior and Related Work

Several works in the literature have proposed strategies for distributed processing which include incremental [1, 6, 7, 8], diffusion [2, 9], sparsity–aware [3, 10] and consensus–based strategies [11]. With the incremental strategy, the processing follows a Hamiltonian cycle, i.e., the information flows through these nodes in one direction, which means each node passes the information to its adjacent node in a uniform direction. However, in order to determine an optimum cyclic path that covers all nodes (considering the noise, interference, path loss and channels between neighbor nodes), this method needs to solve an NP–hard

problem. In addition, when any of the nodes fails, data communication through the cycle is interrupted and the distributed processing breaks down [1].

In distributed diffusion strategies [2, 10], the neighbors for each node are fixed and the combining coefficients are calculated after the network topology is deployed and starts its operation. One potential risk of this approach is that the estimation procedure may be affected by poorly performing links. More specifically, the fixed neighbors and the pre–calculated combining coefficients may not provide an optimized estimation performance for each specified node because there are links that are more severely affected by noise or fading. Moreover, when the number of neighbor nodes is large, each node requires a large bandwidth and transmit power. In [12, 13], the idea of partial diffusion was introduced for reducing communications between neighbor nodes. Prior work on topology design and adjustment techniques includes the studies in [14, 15, 16] and [17], which are not dynamic in the sense that they cannot track changes in the network and mitigate the effects of poor links.

Contributions

The adaptive link selection algorithms for distributed estimation problems are proposed and studied in this chapter. Specifically, we develop adaptive link selection algorithms that can exploit the knowledge of poor links by selecting a subset of data from neighbor nodes. The first approach consists of exhaustive search–based LMS/RLS link selection (ES–LMS/ES–RLS) algorithms, whereas the second technique is based on sparsity–inspired LMS/RLS link selection (SI–LMS/SI–RLS) algorithms. With both approaches, distributed processing can be divided into two steps. The first step is called the adaptation step, in which each node employs LMS or RLS to perform the adaptation through its local information. Following the adaptation step, each node will combine its collected estimates from its neighbors and local estimate, through the proposed adaptive link selection algorithms. The proposed algorithms result in improved estimation performance in terms of the mean–square error (MSE) associated with the estimates. In contrast to previously reported techniques, a key feature of the proposed algorithms is that the combination step involves only a subset of the data associated with the best performing links.

In the ES–LMS and ES–RLS algorithms, we consider all possible combinations for each node with its neighbors and choose the combination associated with the smallest MSE value. In the SI–LMS and SI–RLS algorithms, we incorporate a reweighted zero attraction (RZA) strategy into the adaptive link selection algorithms. The RZA approach is often employed in applications dealing with sparse systems in such a way that it shrinks the small values in the parameter vector to zero, which results in better convergence and steady–state performance. Unlike prior work with sparsity–aware algorithms [3, 18, 19, 20, 21, 22, 23], the proposed SI–LMS and SI–RLS algorithms exploit the possible sparsity of the MSE values associated with each of the links in a different way. In contrast to existing methods that shrink the signal samples to zero, SI–LMS and SI–RLS shrink to zero the links that have poor performance or high MSE values. By using the SI–LMS and SI–RLS algorithms, data associated with unsatisfactory performance will be discarded, which means the effective network topology used in the estimation procedure will change as well. Although the physical topology is not changed by the proposed algorithms, the choice of the data coming from the neighbor nodes for each node is dynamic, leads to the change of combination weights and results in improved performance. We also remark that the topology could be

altered with the aid of the proposed algorithms and a feedback channel which could inform the nodes whether they should be switched off or not. The proposed algorithms are considered for wireless sensor networks and also as a tool for distributed state estimation that could be used in smart grids.

In summary, the main contributions of this chapter are:

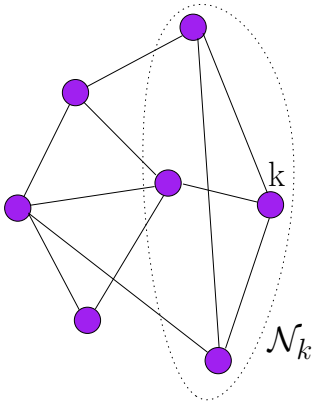
- We present adaptive link selection algorithms for distributed estimation that are able to achieve significantly better performance than existing algorithms.
- We devise distributed LMS and RLS algorithms with link selection capabilities to perform distributed estimation.
- We analyze the MSE convergence and tracking performance of the proposed algorithms and their computational complexities and we derive analytical formulas to predict their MSE performance.
- A simulation study of the proposed and existing distributed estimation algorithms is conducted along with applications in wireless sensor networks and smart grids.

This paper is organized as follows. Section II describes the system model and the problem statement. In Section III, the proposed link selection algorithms are introduced. We analyze the proposed algorithms in terms of their stability, steady–state and tracking performance, and computational complexity in Section IV. The numerical simulation results are provided in Section V. Finally, we conclude the paper in Section VI.

Notation: We use boldface upper case letters to denote matrices and boldface lower case letters to denote vectors. We use $(\cdot)^T$ and $(\cdot)^{-1}$ denote the transpose and inverse operators respectively, $(\cdot)^H$ for conjugate transposition and $(\cdot)^*$ for complex conjugate.

System Model and Problem Statement

Figure 1 Network topology with N nodes



We consider a set of N nodes, which have limited processing capabilities, distributed over a given geographical area as depicted in Fig. 1. The nodes are connected and form a network, which is assumed to be partially connected because nodes can exchange information only with neighbors determined by the connectivity topology. We call a network with this property a partially connected network whereas a fully connected network means that data broadcast by a node can be captured by all other nodes in the network in one hop [24]. We can think of this network as a wireless network, but our analysis also applies to wired

networks such as power grids. In our work, in order to perform link selection strategies, we assume that each node has at least two neighbors.

The aim of the network is to estimate an unknown parameter vector $\boldsymbol{\omega}_0$, which has length M. At every time instant i, each node k takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad i = 1, 2, \ldots, I, \tag{1}$$

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ random regression input signal vector and $n_k(i)$ denotes the Gaussian noise at each node with zero mean and variance $\sigma_{n,k}^2$. This linear model is able to capture or approximate well many input-output relations for estimation purposes [25] and we assume $I > M$. To compute an estimate of $\boldsymbol{\omega_0}$ in a distributed fashion, we need each node to minimize the MSE cost function [2]

$$J_k\big(\boldsymbol{\omega}_k(i)\big) = \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big|^2, \tag{2}$$

where $\mathbb{E}$ denotes expectation and $\boldsymbol{\omega}_k(i)$ is the estimated vector generated by node k at time instant i. Equation (3) is also the definition of the MSE and the global network cost function could be described as

$$J(\boldsymbol{\omega}) = \sum_{k=1}^{N} \mathbb{E}|d_k(i) - \boldsymbol{\omega}^H \boldsymbol{x}_k(i)|^2. \tag{3}$$

To solve this problem, diffusion strategies have been proposed in [2, 9] and [26]. In these strategies, the estimate for each node is generated through a fixed combination strategy given by

$$\boldsymbol{\omega}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\psi}_l(i), \tag{4}$$

where $\mathcal{N}_k$ denotes the set of neighbors of node k including node k itself, $c_{kl} \geq 0$ is the combining coefficient and $\boldsymbol{\psi}_l(i)$ is the local estimate generated by node l through its local information.

There are many ways to calculate the combining coefficient c_{kl} which include the Hastings [27], the Metropolis [28], the Laplacian [29] and the nearest neighbor [30] rules. In this work, due to its simplicity and good performance we adopt the Metropolis rule [28] given by

$$c_{kl} = \begin{cases} \frac{1}{max\{|\mathcal{N}_k|, |\mathcal{N}_l|\}}, & \text{if } k \neq l \text{ are linked} \\ 1 - \sum\limits_{l \in \mathcal{N}_k / k} c_{kl}, & \text{for } k = l. \end{cases} \tag{5}$$

where $|\mathcal{N}_k|$ denotes the cardinality of $\mathcal{N}_k$. The set of coefficients c_{kl} should satisfy [2]

$$\sum_{l \in \mathcal{N}_k \ \forall k} c_{kl} = 1. \tag{6}$$

For the combination strategy mentioned in (4), the choice of neighbors for each node is fixed, which results in some problems and limitations, namely:

- Some nodes may face high levels of noise or interference, which may lead to inaccurate estimates.
- When the number of neighbors for each node is high, large communication bandwidth and high transmit power are required.
- Some nodes may shut down or collapse due to network problems. As a result, local estimates to their neighbors may be affected.

Under such circumstances, a performance degradation is likely to occur when the network cannot discard the contribution of poorly performing links and their associated data in the estimation procedure. In the next section, the proposed adaptive link selection algorithms are presented, which equip a network with the ability to improve the estimation procedure. In the proposed scheme, each node is able to dynamically select the data coming from its neighbors in order to optimize the performance of distributed estimation techniques.

Proposed Adaptive Link Selection Algorithms

In this section, we present the proposed adaptive link selection algorithms. The goal of the proposed algorithms is to optimize the distributed estimation and improve the performance of the network by dynamically changing the topology. These algorithmic strategies give the nodes the ability to choose their neighbors based on their MSE performance. We develop two categories of adaptive link selection algorithms; the first one is based on an exhaustive search, while the second is based on a sparsity–inspired relaxation. The details will be illustrated in the following subsections.

Exhaustive Search–Based LMS/RLS Link Selection

The proposed ES–LMS and ES–RLS algorithms employ an exhaustive search to select the links that yield the best performance in terms of MSE. First, we describe how we define the adaptation step for these two strategies. In the ES–LMS algorithm, we employ the adaptation strategy given by

$$\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \mu_k \boldsymbol{x}_k(i)\big[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big]^*, \tag{7}$$

where μ_k is the step size for each node. In the ES–RLS algorithm, we employ the following steps for the adaptation:

$$\boldsymbol{\Phi}_k^{-1}(i) = \lambda^{-1}\boldsymbol{\Phi}_k^{-1}(i-1) - \frac{\lambda^{-2}\boldsymbol{\Phi}_k^{-1}(i-1)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}_k^{-1}(i-1)}{1+\lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}_k^{-1}(i-1)\boldsymbol{x}_k(i)}, \tag{8}$$

where λ is the forgetting factor. Then, we let

$$\boldsymbol{P}_k(i) = \boldsymbol{\Phi}_k^{-1}(i) \tag{9}$$

and

$$\boldsymbol{k}_k(i) = \frac{\lambda^{-1}\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}{1+\lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}. \tag{10}$$

$$\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \boldsymbol{k}(i)\big[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big]^*, \tag{11}$$

$$\boldsymbol{P}_k(i+1) = \lambda^{-1}\boldsymbol{P}_k(i) - \lambda^{-1}\boldsymbol{k}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i). \tag{12}$$

Following the adaptation step, we introduce the combination step for both ES–LMS and ES–RLS algorithms, based on an exhaustive search strategy. At first, we introduce a tentative set Ω_k using a combinatorial approach described by

$$\Omega_k \in 2^{|\mathcal{N}_k|} \backslash \varnothing, \tag{13}$$

where the set Ω_k is a nonempty set with $2^{|\mathcal{N}_k|}$ elements. After the tentative set Ω_k is defined, we write the cost function (2) for each node as

$$J_k\big(\boldsymbol{\psi}(i)\big) \triangleq \mathbb{E}\big|d_k(i) - \boldsymbol{\psi}^H(i)\boldsymbol{x}_k(i)\big|^2, \tag{14}$$

where

$$\boldsymbol{\psi}(i) \triangleq \sum_{l\in\Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i) \tag{15}$$

is the local estimator and $\boldsymbol{\psi}_l(i)$ is calculated through (7) or (11), depending on the algorithm, i.e., ES–LMS or ES–RLS. With different choices of the set Ω_k, the combining coefficients c_{kl} will be re–calculated through (5), to ensure condition (6) is satisfied.

Then, we introduce the error pattern for each node, which is defined as

$$e_{\Omega_k}(i) \triangleq d_k(i) - \left[\sum_{l\in\Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i)\right]^H \boldsymbol{x}_k(i). \tag{16}$$

For each node k, the strategy that finds the best set $\Omega_k(i)$ must solve the following optimization problem:

$$\hat{\Omega}_k(i) = \arg\min_{\Omega_k\in 2^{N_k}\backslash\varnothing} |e_{\Omega_k}(i)|. \tag{17}$$

After all steps have been completed, the combination step in (4) is performed as described by

$$\boldsymbol{\omega}_k(i+1) = \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i). \tag{18}$$

At this stage, the main steps of the ES–LMS and ES–RLS algorithms have been completed. The proposed ES–LMS and ES–RLS algorithms find the set $\widehat{\Omega}_k(i)$ that minimizes the error pattern in (16) and (17) and then use this set of nodes to obtain $\boldsymbol{\omega}_k(i)$ through (18). The ES–LMS/ES–RLS algorithms are briefly summarized as follows:

Step 1 Each node performs the adaptation through its local information based on the LMS or RLS algorithm.

Step 2 Each node finds the best set $\Omega_k(i)$, which satisfies (17).

Step 3 Each node combines the information obtained from its best set of neighbors through (18).

Table 1 The ES-LMS Algorithm

Initialize: $\boldsymbol{\omega}_k(1)$=0, for $k = 1, 2, \ldots, N$
For each time instant $i = 1, 2, ..., I$
  For each node $k = 1, 2, \ldots, N$
    $\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \mu_k \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)]^*$
  end
  For each node $k = 1, 2, \ldots, N$
    find all possible sets of Ω_k
    $e_{\Omega_k}(i) = d_k(i) - [\sum_{l \in \Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i)]^H \boldsymbol{x}_k(i)$
    $\widehat{\Omega}_k(i) = \arg\min_{\Omega_k} |e_{\Omega_k}(i)|$
    $\boldsymbol{\omega}_k(i+1) = \sum_{l \in \widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i)$
  end
end

Table 2 The ES-RLS Algorithm

Initialize: $\boldsymbol{\omega}_k(1)$=0, for $k = 1, 2, \ldots, N$
    $\boldsymbol{\Phi}_k^{-1}(0) = \delta^{-1}\boldsymbol{I}$, δ = small positive constant
For each time instant $i = 1, 2, ..., I$
  For each node $k = 1, 2, \ldots, N$
    $\boldsymbol{\Phi}_k^{-1}(i) = \lambda^{-1}\boldsymbol{\Phi}_k^{-1}(i-1) - \dfrac{\lambda^{-2}\boldsymbol{\Phi}_k^{-1}(i-1)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}^{-1}(i-1)}{1 + \lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}^{-1}(i-1)\boldsymbol{x}_k(i)}$
    $\boldsymbol{P}_k(i) = \boldsymbol{\Phi}_k^{-1}(i)$
    $\boldsymbol{k}_k(i) = \dfrac{\lambda^{-1}\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}{1 + \lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}$
    $\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \boldsymbol{k}(i)[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)]^*$
    $\boldsymbol{P}_k(i+1) = \lambda^{-1}\boldsymbol{P}_k(i) - \lambda^{-1}\boldsymbol{k}(i)\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i)$
  end
  For each node $k = 1, 2, \ldots, N$
    find all possible sets of Ω_k
    $e_{\Omega_k}(i) = d_k(i) - [\sum_{l \in \Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i)]^H \boldsymbol{x}_k(i)$
    $\widehat{\Omega}_k(i) = \arg\min_{\Omega_k} |e_{\Omega_k}(i)|$
    $\boldsymbol{\omega}_k(i+1) = \sum_{l \in \widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i)$
  end
end

The details of the proposed ES–LMS and ES–RLS algorithms are shown in Tables 1 and 2. When the ES–LMS and ES–RLS algorithms are implemented in networks with a large number of small and low–power sensors, the computational complexity cost may become high, as the algorithm in (17) requires an exhaustive search and needs more computations to examine all the possible sets $\Omega_k(i)$ at each time instant.

Sparsity–Inspired LMS/RLS Link Selection

The ES–LMS/ES–RLS algorithms previously outlined need to examine all possible sets to find a solution at each time instant, which might result in high computational complexity for large networks operating in time–varying scenarios. To solve the combinatorial problem with reduced complexity, we propose sparsity-inspired based SI–LMS and SI–RLS algorithms, which are as simple as standard diffusion LMS or RLS algorithms and are suitable for adaptive implementations and scenarios where the parameters to be estimated are slowly time–varying. The zero–attracting strategy (ZA), reweighted zero–attracting strategy (RZA) and zero–forcing (ZF) are reported in [3] and [31] as for sparsity aware techniques. These approaches are usually employed in applications dealing with sparse systems in scenarios where they shrink the small values in the parameter vector to zero, which results in better convergence rate and steady–state performance. Unlike existing methods that

shrink the signal samples to zero, the proposed SI–LMS and SI–RLS algorithms shrink to zero the links that have poor performance or high MSE values. To detail the novelty of the proposed sparsity–inspired LMS/RLS link selection algorithms, we illustrate the processing in Fig.2.

Figure 2 Sparsity aware signal processing strategies

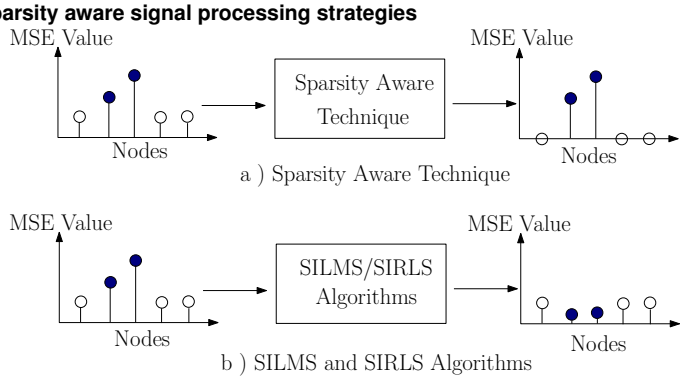


Fig. 2 (a) shows a standard type of sparsity–aware processing. We can see that, after being processed by a sparsity–aware algorithm, the nodes with small MSE values will be shrunk to zero. In contrast, the proposed SI–LMS and SI–RLS algorithms will keep the nodes with lower MSE values and reduce the combining weight of the nodes with large MSE values as illustrated in Fig. 2 (b). When compared with ES–type algorithms, the SI–LMS/RLS algorithms do not need to consider all possible combinations of nodes, which means the SI–LMS/RLS algorithms have lower complexity. In the following, we will show how the proposed SI–LMS/SI–RLS algorithms are employed to realize the link selection strategy automatically.

In the adaptation step, we follow the same procedure in (7)–(11) as that of the ES–LMS and ES–RLS algorithms for the SI–LMS and SI–RLS algorithms, respectively. Then we reformulate the combination step. First, we introduce the log–sum penalty into the combination step in (4). Different penalty terms have been considered for this task. We have adopted a heuristic approach [3, 32] known as reweighted zero–attracting strategy into the combination step in (4) because this strategy has shown an excellent performance and is simple to implement. The log–sum penalty is defined as:

$$f_1(e_k(i)) = \sum_{l \in \mathcal{N}_k} \log\left(1 + \varepsilon |e_{kl}(i)|\right), \tag{19}$$

where the error $e_{kl}(i)(l \in \mathcal{N}_k)$, which stands for the neighbor node l of node k including node k itself, is defined as

$$e_{kl}(i) \triangleq d_k(i) - \boldsymbol{\psi}_l^H(i)\boldsymbol{x}_k(i) \tag{20}$$

and ε is the shrinkage magnitude. Then, we introduce the vector and matrix quantities required to describe the combination step. We first define a vector $\boldsymbol{c}_k$ that contains the combining coefficients for each neighbor of node k including node k itself as described by

$$\boldsymbol{c}_k \triangleq [c_{kl}], \quad l \in \mathcal{N}_k. \tag{21}$$

Then, we define a matrix $\boldsymbol{\Psi}_k$ that includes all the estimated vectors, which are generated after the adaptation step of SI–LMS and of SI–RLS for each neighbor of node k including node k itself as given by

$$\boldsymbol{\Psi}_k \triangleq [\boldsymbol{\psi}_l(i)], \quad l \in \mathcal{N}_k. \tag{22}$$

Note that the adaptation steps of SI–LMS and SI–RLS are identical to (7) and (11), respectively. An error vector $\hat{\boldsymbol{e}}_k$ that contains all error values calculated through (20) for each neighbor of node k including node k itself is expressed by

$$\hat{\boldsymbol{e}}_k \triangleq [e_{kl}(i)], \quad l \in \mathcal{N}_k. \tag{23}$$

To devise the sparsity–inspired approach, we have modified the vector $\hat{\boldsymbol{e}}_k$ in the following way:

1 The element with largest absolute value $|e_{kl}(i)|$ in $\hat{\boldsymbol{e}}_k$ will be kept as $|e_{kl}(i)|$.
2 The element with smallest absolute value will be set to $-|e_{kl}(i)|$. This process will ensure the node with smallest error pattern has a reward on its combining coefficient.
3 The remaining entries will be set to zero.

At this point, the combination step can be defined as [32]

$$\boldsymbol{\omega}_k(i) = \sum_{j=1}^{|\mathcal{N}_k|} \left[c_{k,j} - \rho \frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}} \right] \boldsymbol{\psi}_{k,j}, \tag{24}$$

where $c_{k,j}, \hat{e}_{k,j}$ stand for the jth element in the $\boldsymbol{c}_k, \hat{\boldsymbol{e}}_k$ and $\boldsymbol{\psi}_{k,j}$ stands for the jth column in $\boldsymbol{\Psi}_k$. The parameter ρ is used to control the algorithm's shrinkage intensity. We then calculate the partial derivative of $\hat{\boldsymbol{e}}_k[j]$:

$$\begin{aligned} \frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}} &= \frac{\partial\big(\log(1+\varepsilon|e_{kl}(i)|)\big)}{\partial\big(e_{kl}(i)\big)} \\ &= \varepsilon \frac{\operatorname{sign}(e_{kl}(i))}{1+\varepsilon|e_{kl}(i)|} \quad l \in \mathcal{N}_k \\ &= \varepsilon \frac{\operatorname{sign}(\hat{e}_{k,j})}{1+\varepsilon|\hat{e}_{k,j}|}. \end{aligned} \tag{25}$$

To ensure that $\sum_{j=1}^{|\mathcal{N}_k|} \left(c_{k,j} - \rho \frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}} \right) = 1$, we replace $\hat{e}_{k,j}$ with $\xi_{\min}$ in the denominator of (25), where the parameter $\xi_{\min}$ stands for the minimum absolute value of $e_{kl}(i)$ in $\hat{\boldsymbol{e}}_k$. Then, (25) can be rewritten as

$$\frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}} \approx \varepsilon \frac{\operatorname{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}. \tag{26}$$

At this stage, the log-sum penalty performs shrinkage and selects the set of estimates from the neighbor nodes with the best performance, at the combination step. The function $\operatorname{sign}(a)$ is defined as

$$\operatorname{sign}(a) = \begin{cases} a/|a| & a \neq 0 \\ 0 & a = 0. \end{cases} \tag{27}$$

Then, by inserting (26) into (24), the proposed combination step is given by

$$\boldsymbol{\omega}_k(i) = \sum_{j=1}^{|\mathcal{N}_k|} \left[c_{k,j} - \rho\varepsilon \frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|} \right] \boldsymbol{\psi}_{k,j}. \tag{28}$$

Note that the condition $c_{k,j} - \rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|} \geq 0$ is enforced in (28). When $c_{k,j} - \rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|} = 0$, it means that the corresponding node has been discarded from the combination step. In the following time instant, if this node still has the largest error, there will be no changes in the combining coefficients for this set of nodes.

To guarantee the stability, the parameter ρ is assumed to be sufficiently small and the penalty takes effect only on the element in $\hat{\boldsymbol{e}}_k$ for which the magnitude is comparable to $1/\varepsilon$ [3]. Moreover, there is little shrinkage exerted on the element in $\hat{\boldsymbol{e}}_k$ whose $|\hat{\boldsymbol{e}}_k[j]| \ll 1/\varepsilon$. The SI–LMS and SI–RLS algorithms perform link selection by the adjustment of the combining coefficients through (28). At this point, it should be emphasized that:

- The process in (28) satisfies condition (6), as the penalty and reward amounts of the combining coefficients are the same for the nodes with maximum and minimum error, respectively, and there are no changes for the rest nodes in the set.
- When computing (28), there are no matrix–vector multiplications. Therefore, no additional complexity is introduced. As described in (24), only the jth element of $\boldsymbol{c}_k$, $\hat{\boldsymbol{e}}_k$ and jth column of $\boldsymbol{\Psi}_k$ are used for calculation.

For the neighbor node with the largest MSE value, after the modifications of $\hat{\boldsymbol{e}}_k$, its $e_{kl}(i)$ value in $\hat{\boldsymbol{e}}_k$ will be a positive number which will lead to the term $\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}$ in (28) being positive too. This means that the combining coefficient for this node will be shrunk and the weight for this node to build $\boldsymbol{\omega}_k(i)$ will be shrunk too. In other words, when a node encounters high noise or interference levels, the corresponding MSE value might be large. As a result, we need to reduce the contribution of that node.

In contrast, for the neighbor node with the smallest MSE, as its $e_{kl}(i)$ value in $\hat{\boldsymbol{e}}_k$ will be a negative number, the term $\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}$ in (28) will be negative too. As a result, the weight for this node associated with the smallest MSE to build $\boldsymbol{\omega}_k(i)$ will be increased. For the remaining neighbor nodes, the entry $e_{kl}(i)$ in $\hat{\boldsymbol{e}}_k$ is zero, which means the term $\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}$ in (28) is zero and there is no change for the weights to build $\boldsymbol{\omega}_k(i)$. The main steps for the proposed SI–LMS and SI–RLS algorithms are listed as follows:

Step 1 Each node carries out the adaptation through its local information based on the LMS or RLS algorithm.

Step 2 Each node calculates the error pattern through (20).

Step 3 Each node modifies the error vector $\hat{\boldsymbol{e}}_k$.

Step 4 Each node combines the information obtained from its selected neighbors through (28).

The SI–LMS and SI–RLS algorithms are detailed in Table 3. For the ES–LMS/ES–RLS and SI–LMS/SI–RLS algorithms, we design different combination steps and employ the same adaptation procedure, which means the proposed algorithms have the ability to equip any diffusion–type wireless networks operating with other than the LMS and RLS algorithms. This includes, for example, the diffusion conjugate gradient strategy [33]. Apart from using weights related to the node degree, other signal dependent approaches may also be considered, e.g., the parameter vectors could be weighted according to the signal-to-noise ratio (SNR) (or the noise variance) at each node within the neighborhood.

Table 3 The SI-LMS and SI-RLS Algorithms

Initialize: $\boldsymbol{\omega}_k(-1)$=0, $k = 1, 2, \ldots, N$
$\boldsymbol{P}(0) = \delta^{-1}\boldsymbol{I}, \delta =$ small positive constant
For each time instant $i = 1, 2, ..., I$
For each node $k = 1, 2, \ldots, N$
The adaptation step for computing $\boldsymbol{\psi}_k(i)$
is exactly the same as the ES-LMS and ES-RLS
for the SI-LMS and SI-RLS algorithms respectively
end
For each node $k = 1, 2, \ldots, N$
$e_{kl}(i) = d_k(i) - \boldsymbol{\psi}_l^H(i)\boldsymbol{x}_k(i) \quad l \in \mathcal{N}_k$
$\boldsymbol{c}_k = [c_{kl}] \quad l \in \mathcal{N}_k$
$\boldsymbol{\Psi}_k = [\boldsymbol{\psi}_l(i)] \quad l \in \mathcal{N}_k$
$\hat{\boldsymbol{e}}_k = [e_{kl}(i)] \quad l \in \mathcal{N}_k$
Find the maximum and minimum absolute terms in $\boldsymbol{e}_k$
Modified $\hat{\boldsymbol{e}}_k$ as $\hat{\boldsymbol{e}}_k$=[0$\cdots$0,$\underbrace{|e_{kl}(i)|}_{\max}$,0$\cdots$0,$\underbrace{-|e_{kl}(i)|}_{\min}$,0$\cdots$0]
$\xi_{\min} = \min\left(|e_{kl}(i)|\right)$
$\boldsymbol{\omega}_k(i) = \sum_{j=1}^{|\mathcal{N}_k|}\left[c_{k,j} - \rho\varepsilon\frac{\mathrm{sign}(e_{k,j})}{1+\varepsilon|\xi_{\min}|}\right]\boldsymbol{\psi}_{k,j}$
end
end

Analysis of the proposed algorithms

In this section, a statistical analysis of the proposed algorithms is developed, including a stability analysis and an MSE analysis of the steady–state and tracking performance. In addition, the computational complexity of the proposed algorithms is also detailed. Before we start the analysis, we make some assumptions that are common in the literature [25].

Assumption I: The weight-error vector $\boldsymbol{\varepsilon}_k(i)$ and the input signal vector $\boldsymbol{x}_k(i)$ are statistically independent, and the weight–error vector for node k is defined as

$$\boldsymbol{\varepsilon}_k(i) \triangleq \boldsymbol{\omega}_k(i) - \boldsymbol{\omega}_0, \tag{29}$$

where $\boldsymbol{\omega}_0$ denotes the optimum Wiener solution of the actual parameter vector to be estimated, and $\boldsymbol{\omega}_k(i)$ is the estimate produced by a proposed algorithm at time instant i.

Assumption II: The input signal vector $x_l(i)$ is drawn from a stochastic process, which is ergodic in the autocorrelation function [25].

Assumption III: The $M \times 1$ vector $\boldsymbol{q}(i)$ represents a stationary sequence of independent zero–mean vectors and positive definite autocorrelation matrix $\boldsymbol{Q} = \mathbb{E}[\boldsymbol{q}(i)\boldsymbol{q}^H(i)]$, which is independent of $\boldsymbol{x}_k(i)$, $\boldsymbol{n}_k(i)$ and $\boldsymbol{\varepsilon}_l(i)$.

Assumption IV (Independence): All regressor input signals $\boldsymbol{x}_k(i)$ are spatially and temporally independent. This assumption allows us to consider the input signal $\boldsymbol{x}_k(i)$ independent of $\boldsymbol{\omega}_l(i), l \in \mathcal{N}_k$.

Stability Analysis

In general, to ensure that a partially-connected network performance can converge to the global network performance, the estimates should be propagated across the network [34]. The work in [14] shows that it is central to the performance that each node should be able to reach the other nodes through one or multiple hops [34].

To discuss the stability analysis of the proposed ES–LMS and SI–LMS algorithms, we first substitute (7) into (18) and obtain

$$\begin{aligned}\boldsymbol{\omega}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i+1)\\ &= \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\omega}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i)\\ &= \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\omega}_0+\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i)\\ &= \sum_{l\in\widehat{\Omega}_k(i)} \boldsymbol{\omega}_0 c_{kl} + \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i)\\ &\text{subject to } \sum_l c_{kl}(i)=1\\ &= \boldsymbol{\omega}_0 + \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i). \end{aligned} \tag{30}$$

Then, we have

$$\boldsymbol{\varepsilon}_k(i+1) = \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i). \tag{31}$$

By employing *Assumption IV*, we start with (31) for the ES–LMS algorithm and define the global vectors and matrices:

$$\boldsymbol{\varepsilon}(i+1) \triangleq [\boldsymbol{\varepsilon}_1(i+1),\cdots,\boldsymbol{\varepsilon}_N(i+1)]^T \tag{32}$$

$$\boldsymbol{M} \triangleq \text{diag}\{\mu_1\boldsymbol{I}_M,...,\mu_N\boldsymbol{I}_M\} \tag{33}$$

$$\boldsymbol{D}(i+1) \triangleq \text{diag}\{\boldsymbol{x}_1(i+1)\boldsymbol{x}_1^H(i+1),...,\boldsymbol{x}_N(i+1)\boldsymbol{x}_N^H(i+1)\} \tag{34}$$

and the $NM\times 1$ vector

$$\boldsymbol{g}(i+1) = [\boldsymbol{x}_1^T(i+1)n_1(i+1),\cdots,\boldsymbol{x}_N^T(i+1)n_N(i+1)]^T. \tag{35}$$

We also define an $N\times N$ matrix $\boldsymbol{C}$ where the combining coefficients $\{c_{kl}\}$ correspond to the $\{l,k\}$ entries of the matrix $\boldsymbol{C}$ and the $NM\times NM$ matrix $\boldsymbol{C}_G$ with a Kronecker structure:

$$\boldsymbol{C}_G = \boldsymbol{C}\otimes\boldsymbol{I}_M \tag{36}$$

where $\otimes$ denotes the Kronecker product.

By inserting $e_l(i+1) = e_{0-l}(i+1) - \boldsymbol{\varepsilon}_l^H(i)\boldsymbol{x}_l(i+1)$ into (31), the global version of (31) can then be written as

$$\boldsymbol{\varepsilon}(i+1) = \boldsymbol{C}_G^T\big[\boldsymbol{I}-\boldsymbol{M}\boldsymbol{D}(i+1)\big]\boldsymbol{\varepsilon}(i) + \boldsymbol{C}_G^T\boldsymbol{M}\boldsymbol{g}(i+1), \tag{37}$$

where $e_{0-l}(i+1)$ is the estimation error produced by the Wiener filter for node l as described by

$$e_{0-l}(i+1) = d_l(i) - \boldsymbol{\omega}_0^H \boldsymbol{x}_l(i). \tag{38}$$

If we define

$$\begin{aligned} \mathcal{D} &\triangleq \mathbb{E}[\boldsymbol{D}(i+1)] \\ &= \text{diag}\{\boldsymbol{R}_1, ..., \boldsymbol{R}_N\} \end{aligned} \tag{39}$$

and take the expectation of (37), we arrive at

$$\mathbb{E}\{\boldsymbol{\varepsilon}(i+1)\} = \boldsymbol{C}_G^T\big[\boldsymbol{I} - \boldsymbol{M}\mathcal{D}\big]\mathbb{E}\{\boldsymbol{\varepsilon}(i)\}. \tag{40}$$

Before we proceed, let us define $\boldsymbol{X} = \boldsymbol{I} - \boldsymbol{M}\mathcal{D}$. We say that a square matrix $\boldsymbol{X}$ is stable if it satisfies $\boldsymbol{X}^i \to 0$ as $i \to \infty$. A known result in linear algebra states that a matrix is stable if, and only if, all its eigenvalues lie inside the unit circle. We need the following lemma to proceed [9].

Lemma 1: Let $\boldsymbol{C}_G$ and $\boldsymbol{X}$ denote arbitrary $NM \times NM$ matrices, where $\boldsymbol{C}_G$ has real, non-negative entries, with columns adding up to one. Then, the matrix $\boldsymbol{Y} = \boldsymbol{C}_G^T\boldsymbol{X}$ is stable for any choice of $\boldsymbol{C}_G$ if, and only if, $\boldsymbol{X}$ is stable.

Proof: Assume that $\boldsymbol{X}$ is stable, it is true that for every square matrix $\boldsymbol{X}$ and every $\alpha > 0$, there exists a submultiplicative matrix norm $||\cdot||_\tau$ that satisfies $||\boldsymbol{X}||_\tau \leq \tau(\boldsymbol{X}) + \alpha$, where the submultiplicative matrix norm $||\boldsymbol{AB}|| \leq ||\boldsymbol{A}|| \cdot ||\boldsymbol{B}||$ holds and $\tau(\boldsymbol{X})$ is the spectral radius of $\boldsymbol{X}$ [35, 36]. Since $\boldsymbol{X}$ is stable, $\tau(\boldsymbol{X}) < 1$, and we can choose $\alpha > 0$ such that $\tau(\boldsymbol{X}) + \alpha = v < 1$ and $||\boldsymbol{X}||_\tau \leq v < 1$. Then we obtain [9]

$$\begin{aligned} ||\boldsymbol{Y}^i||_\tau &= ||(\boldsymbol{C}_G^T\boldsymbol{X})^i||_\tau \\ &\leq ||(\boldsymbol{C}_G^T)^i||_\tau \cdot ||\boldsymbol{X}^i||_\tau \\ &\leq v^i||(\boldsymbol{C}_G^T)^i||_\tau. \end{aligned} \tag{41}$$

Since $\boldsymbol{C}_G^T$ has non–negative entries with columns that add up to one, it is element–wise bounded by unity. This means its Frobenius norm is bounded as well and by the equivalence of norms, so is any norm, in particular $||(\boldsymbol{C}_G^T)^i||_\tau$. As a result, we have

$$\lim_{i\to\infty} ||\boldsymbol{Y}^i||_\tau = \boldsymbol{0}, \tag{42}$$

so $\boldsymbol{Y}^i$ converges to the zero matrix for large i. Therefor, $\boldsymbol{Y}$ is stable.

In view of *Lemma 1* and (82), we need the matrix $\boldsymbol{I} - \boldsymbol{M}\mathcal{D}$ to be stable. As a result, it requires $\boldsymbol{I} - \mu_k\boldsymbol{R}_k$ to be stable for all k, which holds if the following condition is satisfied:

$$0 < \mu_k < \frac{2}{\lambda_{max}(\boldsymbol{R}_k)} \tag{43}$$

where $\lambda_{max}(\boldsymbol{R}_k)$ is the largest eigenvalue of the correlation matrix $\boldsymbol{R}_k$. The difference between the ES–LMS and SI–LMS algorithms is the strategy to calculate the matrix $\boldsymbol{C}$.

Lemma 1 indicates that for any choice of $\boldsymbol{C}$, only $\boldsymbol{X}$ needs to be stable. As a result, SI–LMS has the same convergence condition as in (43). Given the convergence conditions, the proposed ES–LMS/ES–RLS and SI–LMS/SI–RLS algorithms will adapt according to the network connectivity by choosing the group of nodes with the best available performance to construct their estimates.

MSE Steady–State Analysis

In this part of the analysis, we devise formulas to predict the excess MSE (EMSE) of the proposed algorithms. The error signal at node k can be expressed as

$$\begin{aligned} e_k(i) &= d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i) \\ &= d_k(i) - [\boldsymbol{\omega}_0 - \boldsymbol{\varepsilon}_k(i)]^H\boldsymbol{x}_k(i) \\ &= d_k(i) - \boldsymbol{\omega}_0^H\boldsymbol{x}_k(i) + \boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i) \\ &= e_{0-k} + \boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i). \end{aligned} \tag{44}$$

With *Assumption I*, the MSE expression can be derived as

$$\begin{aligned} \mathcal{J}_{mse-k}(i) &= \mathbb{E}[|e_k(i)|^2] \\ &= \mathbb{E}\Big[\big(e_{0-k} + \boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i)\big)\big(e_0^* + \boldsymbol{x}_k^H(i)\boldsymbol{\varepsilon}_k(i)\big)\Big] \\ &= \mathcal{J}_{min-k} + \mathbb{E}[\boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{\varepsilon}_k(i)] \\ &= \mathcal{J}_{min-k} + \text{tr}\{\mathbb{E}[\boldsymbol{\varepsilon}_k(i)\boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]\} \\ &= \mathcal{J}_{min-k} + \text{tr}\{\mathbb{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]\mathbb{E}[\boldsymbol{\varepsilon}_k(i)\boldsymbol{\varepsilon}_k^H(i)]\} \\ &= \mathcal{J}_{min-k} + \text{tr}\{\boldsymbol{R}_k(i)\boldsymbol{K}_k(i)\}, \end{aligned} \tag{45}$$

where $\text{tr}(\cdot)$ denotes the trace of a matrix and $\mathcal{J}_{min-k}$ is the minimum mean–square error (MMSE) for node k [25]:

$$\mathcal{J}_{min-k} = \sigma_{d,k}^2 - \boldsymbol{p}_k^H(i)\boldsymbol{R}_k^{-1}(i)\boldsymbol{p}_k(i), \tag{46}$$

$\boldsymbol{R}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]$ is the correlation matrix of the inputs for node k, $\boldsymbol{p}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)d_k^*(i)]$ is the cross–correlation vector between the inputs and the measurement $d_k(i)$, and $\boldsymbol{K}_k(i) = \mathbb{E}[\boldsymbol{\varepsilon}_k(i)\boldsymbol{\varepsilon}_k^H(i)]$ is the weight–error correlation matrix. From [25], the EMSE is defined as the difference between the mean–square error at time instant i and the minimum mean–square error. Then, we can write

$$\begin{aligned} \mathcal{J}_{ex-k}(i) &= \mathcal{J}_{mse-k}(i) - \mathcal{J}_{min-k} \\ &= \text{tr}\{\boldsymbol{R}_k(i)\boldsymbol{K}_k(i)\}. \end{aligned} \tag{47}$$

For the proposed adaptive link selection algorithms, we will derive the EMSE formulas separately based on (47) and *we assume that the input signal is modeled as a stationary process*.

ES–LMS

To update the estimate $\boldsymbol{\omega}_k(i)$, we employ

$$\begin{aligned}\boldsymbol{\omega}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i+1)\\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_l^*(i+1)]\\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i)+\mu_l\boldsymbol{x}_l(i+1)(d_l(i+1)-\boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))].\end{aligned} \tag{48}$$

Then, subtracting $\boldsymbol{\omega}_0$ from both sides of (48), we arrive at

$$\begin{aligned}\boldsymbol{\varepsilon}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i)+\mu_l\boldsymbol{x}_l(i+1)(d_l(i+1)-\boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))]\\ &\quad-\sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\omega}_0\\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)\big(d_l(i+1)-\boldsymbol{x}_l^H(i+1)(\boldsymbol{\varepsilon}_l(i)+\boldsymbol{\omega}_0)\big)\Big]\\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)\big(d_l(i+1)-\boldsymbol{x}_l^H(i+1)\boldsymbol{\varepsilon}_l(i)-\boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_0\big)\Big]\\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i)-\mu_l\boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big]\\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\big(\boldsymbol{I}-\mu_l\boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big].\end{aligned} \tag{49}$$

Let us introduce the random variables $\alpha_{kl}(i)$:

$$\alpha_{kl}(i)=\begin{cases}1, & \text{if } l\in\widehat{\Omega}_k(i)\\ 0, & \text{otherwise.}\end{cases} \tag{50}$$

At each time instant, each node will generate data associated with network covariance matrices $\boldsymbol{A}_k$ with size $N\times N$ which reflect the network topology, according to the exhaustive search strategy. In the network covariance matrices $\boldsymbol{A}_k$, a value equal to 1 means nodes k and l are linked and a value 0 means nodes k and l are not linked.

For example, suppose a network has 5 nodes. For node 3, there are two neighbor nodes, namely, nodes 2 and 5. Through Eq. (13), the possible configurations of set Ω_3 are $\{3,2\}$, $\{3,5\}$ and $\{3,2,5\}$. Evaluating all the possible sets for Ω_3, the relevant covariance matrices $\boldsymbol{A}_3$ with size 5×5 at time instant i are described in Fig. 3.

Then, the coefficients α_{kl} are obtained according to the covariance matrices $\boldsymbol{A}_k$. In this example, the three sets of α_{kl} are respectively shown in Table 4.

The parameters c_{kl} will then be calculated through Eq. (5) for different choices of matrices $\boldsymbol{A}_k$ and coefficients α_{kl}. After α_{kl} and c_{kl} are calculated, the error pattern for each

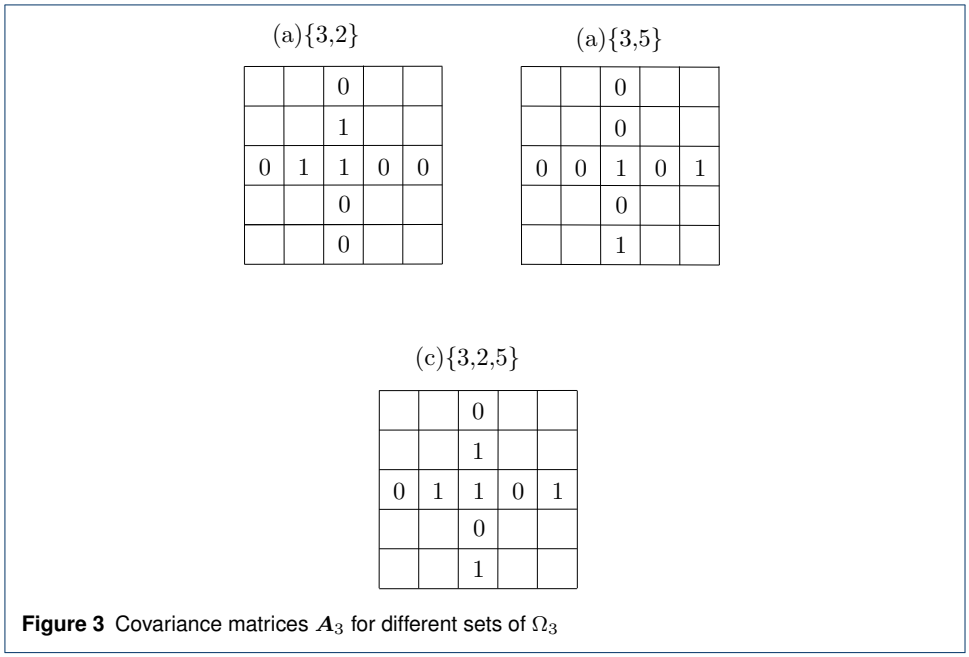


Figure 3 Covariance matrices $\boldsymbol{A}_3$ for different sets of Ω_3

Table 4 Coefficients α_{kl} for different sets of Ω_3

$$\{2,3\}\begin{cases}\alpha_{31}=0\\ \alpha_{32}=1\\ \alpha_{33}=1\\ \alpha_{34}=0\\ \alpha_{35}=0\end{cases}\qquad \{3,5\}\begin{cases}\alpha_{31}=0\\ \alpha_{32}=0\\ \alpha_{33}=1\\ \alpha_{34}=0\\ \alpha_{35}=1\end{cases}\qquad \{2,3,5\}\begin{cases}\alpha_{31}=0\\ \alpha_{32}=1\\ \alpha_{33}=1\\ \alpha_{34}=0\\ \alpha_{35}=1\end{cases}$$

possible Ω_k will be calculated through (16) and the set with the smallest error will be selected according to (17).

With the newly defined α_{kl}, (49) can be rewritten as

$$\boldsymbol{\varepsilon}_k(i+1)=\sum_{l\in\mathcal{N}_k}\alpha_{kl}(i)c_{kl}(i)\Big[\big(\boldsymbol{I}-\mu_l\boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big]. \tag{51}$$

Starting from (47), we then focus on $\boldsymbol{K}_k(i+1)$.

$$\boldsymbol{K}_k(i+1)=\mathbb{E}[\boldsymbol{\varepsilon}_k(i+1)\boldsymbol{\varepsilon}_k^H(i+1)]. \tag{52}$$

In (51), the term $\alpha_{kl}(i)$ is determined through the network topology for each subset, while the term $c_{kl}(i)$ is calculated through the Metropolis rule. We assume that $\alpha_{kl}(i)$ and $c_{kl}(i)$ are statistically independent from the other terms in (51). Upon convergence, the parameters $\alpha_{kl}(i)$ and $c_{kl}(i)$ do not vary because at steady state the choice of the subset $\widehat{\Omega}_k(i)$ for each node k will be fixed. Then, under these assumptions, substituting (51) into (52) we

arrive at:

$$\begin{aligned}\boldsymbol{K}_k(i+1) = \sum_{l\in\mathcal{N}_k}\mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Big(\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\boldsymbol{K}_l(i)\times\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\\ +\mu_l^2 e_{0-l}(i+1)e_{0-l}^*(i+1)\times\boldsymbol{R}_l(i+1)\Big)+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ \times\Big(\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\boldsymbol{K}_{l,q}(i)\big(\boldsymbol{I}-\mu_q\boldsymbol{R}_l(i+1)\big)\\ +\mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\boldsymbol{R}_{l,q}(i+1)\Big)+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ \times\Big(\big(\boldsymbol{I}-\mu_q\boldsymbol{R}_q(i+1)\big)\boldsymbol{K}_{l,q}^H(i)\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\\ +\mu_l\mu_q e_{0-q}(i+1)e_{0-l}^*(i+1)\boldsymbol{R}_{l,q}^H(i+1)\Big)\end{aligned}\tag{53}$$

where $\boldsymbol{R}_{l,q}(i+1)=\mathbb{E}[\boldsymbol{x}_l(i+1)\boldsymbol{x}_q^H(i+1)]$ and $\boldsymbol{K}_{l,q}(i)=\mathbb{E}[\boldsymbol{\varepsilon}_l(i)\boldsymbol{\varepsilon}_q^H(i)]$. To further simplify the analysis, we assume that the samples of the input signal $\boldsymbol{x}_k(i)$ are uncorrelated, i.e., $\boldsymbol{R}_k=\sigma_{x,k}^2\boldsymbol{I}$ with $\sigma_{x,k}^2$ being the variance. Using the diagonal matrices $\boldsymbol{R}_k=\boldsymbol{\Lambda}_k=\sigma_{x,k}^2\boldsymbol{I}$ and $\boldsymbol{R}_{l,q}=\boldsymbol{\Lambda}_{l,q}=\sigma_{x,l}\sigma_{x,q}\boldsymbol{I}$ we can write

$$\begin{aligned}\boldsymbol{K}_k(i+1) = \sum_{l\in\mathcal{N}_k}\mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Big(\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)\boldsymbol{K}_l(i)\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)\\ +\mu_l^2 e_{0-l}(i+1)e_{0-l}^*(i+1)\boldsymbol{\Lambda}_l\Big)\\ +\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\times\Big(\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)\boldsymbol{K}_{l,q}(i)\big(\boldsymbol{I}-\mu_q\boldsymbol{\Lambda}_q\big)\\ +\mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\boldsymbol{\Lambda}_{l,q}\Big)+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ \times\Big(\big(\boldsymbol{I}-\mu_q\boldsymbol{\Lambda}_q\big)\boldsymbol{K}_{l,q}^H(i)\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)+\mu_l\mu_q e_{0-q}(i+1)e_{0-l}^*(i+1)\boldsymbol{\Lambda}_{l,q}^H\Big).\end{aligned}\tag{54}$$

Due to the structure of the above equations, the approximations and the quantities involved, we can decouple (54) into

$$\begin{aligned}K_k^n(i+1) = &\sum_{l\in\mathcal{N}_k}\mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Big(\big(1-\mu_l\lambda_l^n\big)K_l^n(i)\big(1-\mu_l\lambda_l^n\big)\\&+\mu_l^2 e_{0-l}(i+1)e_{0-l}^*(i+1)\lambda_l^n\Big)\\&+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\Big(\big(1-\mu_l\lambda_l^n\big)K_{l,q}^n(i)\big(1-\mu_q\lambda_q^n\big)\\&+\mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\lambda_{l,q}^n\Big)+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\&\times\Big(\big(1-\mu_q\lambda_q^n\big)(K_{l,q}^n(i))^H\big(1-\mu_l\lambda_l^n\big)+\mu_l\mu_q e_{0-q}(i+1)e_{0-l}^*(i+1)\lambda_{l,q}^n\Big),\end{aligned}\tag{55}$$

where $K_k^n(i+1)$ is the nth element of the main diagonal of $\boldsymbol{K}_k(i+1)$. With the assumption that $\alpha_{kl}(i)$ and $c_{kl}(i)$ are statistically independent from the other terms in (51), we can rewrite (55) as

$$\begin{aligned}K_k^n(i+1) = &\sum_{l\in\mathcal{N}_k}\mathbb{E}\Big[\alpha_{kl}^2(i)\Big]\mathbb{E}\Big[c_{kl}^2(i)\Big]\Big(\big(1-\mu_l\lambda_l^n\big)^2K_l^n(i)+\mu_l^2e_{0-l}(i+1)e_{0-l}^*(i+1)\lambda_l^n\Big)\\&+2\times\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)\Big]\mathbb{E}\Big[c_{kl}(i)c_{kq}(i)\Big]\Big(\big(1-\mu_l\lambda_l^n\big)\big(1-\mu_q\lambda_q^n\big)K_{l,q}^n(i)\\&+\mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\lambda_{l,q}^n\Big).\end{aligned}\tag{56}$$

By taking $i\to\infty$, we can obtain (57).

$$K_k^n(\text{ES-LMS}) = \frac{\sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2\mu_l^2\mathcal{J}_{min-l}\lambda_l^n+2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}\mu_l\mu_q e_{0-l}e_{0-q}^*\lambda_{l,q}^n}{1-\sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2(1-\mu_l\lambda_l^n)^2-2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}(1-\mu_l\lambda_l^n)(1-\mu_q\lambda_q^n)}.\tag{57}$$

We assume that the choice of covariance matrix $\boldsymbol{A}_k$ for node k is fixed upon the proposed algorithms convergence, as a result, the covariance matrix $\boldsymbol{A}_k$ is deterministic and does not vary. In the above example, we assume the choice of $\boldsymbol{A}_3$ is fixed as shown in Fig. 4.

Then the coefficients α_{kl} will also be fixed and given by

$$\begin{cases}\alpha_{31}=0\\ \alpha_{32}=1\\ \alpha_{33}=1\\ \alpha_{34}=0\\ \alpha_{35}=1\end{cases}$$

		0		
		1		
0	1	1	0	1
		0		
		1		

Figure 4 Covariance matrix $\boldsymbol{A}_3$ upon convergence

as well as the parameters c_{kl} that are computed using the Metropolis combining rule. As a result, the coefficients α_{kl} and the coefficients c_{kl} are deterministic and can be taken out from the expectation. The MSE is then given by

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(\text{ES-LMS}). \tag{58}$$

SI–LMS

For the SI–LMS algorithm, we do not need to consider all possible combinations. This algorithm simply adjusts the combining coefficients for each node with its neighbors in order to select the neighbor nodes that yield the smallest MSE values. Thus, we redefine the combining coefficients through (28)

$$c_{kl-new} = c_{kl} - \rho\varepsilon \frac{\text{sign}(|e_{kl}|)}{1+\varepsilon|\xi_{\min}|} \ \ (l \in \mathcal{N}_k). \tag{59}$$

For each node k, at time instant i, after it receives the estimates from all its neighbors, it calculates the error pattern $e_{kl}(i)$ for every estimate received through Eq. (20) and finds the nodes with the largest and smallest errors. An error vector $\hat{\boldsymbol{e}}_k$ is then defined through (23), which contains all error patterns $e_{kl}(i)$ for node k.

Then a procedure which is detailed after Eq. (23) is carried out and modifies the error vector $\hat{\boldsymbol{e}}_k$. For example, suppose node 5 has three neighbor nodes, which are nodes $3, 6$ and 8. The error vector $\hat{\boldsymbol{e}}_5$ has the form described by $\hat{\boldsymbol{e}}_5 = [e_{53}, e_{55}, e_{56}, e_{58}] = [0.023, 0.052, -0.0004, -0.012]$. After the modification, the error vector $\hat{\boldsymbol{e}}_5$ will be edited as $\hat{\boldsymbol{e}}_5 = [0, 0.052, -0.0004, 0]$. The quantity h_{kl} is then defined as

$$h_{kl} = \rho\varepsilon \frac{\text{sign}(|e_{kl}|)}{1+\varepsilon|\xi_{\min}|} \ \ (l \in \mathcal{N}_k), \tag{60}$$

and the term 'error pattern' e_{kl} in (60) is from the modified error vector $\hat{\boldsymbol{e}}_k$.

From [32], we employ the relation $\mathbb{E}[\text{sign}(e_{kl})] \approx \text{sign}(e_{0-k})$. According to Eqs. (1) and (38), when the proposed algorithm converges at node k or the time instant i goes to infinity, we assume that the error e_{0-k} will be equal to the noise variance at node k. Then, the asymptotic value h_{kl} can be divided into three situations due to the rule of the SI–LMS algorithm:

$$h_{kl} = \begin{cases} \rho\varepsilon \frac{\text{sign}(|e_{0-k}|)}{1+\varepsilon|e_{0-k}|} & \text{for the node with the largest MSE} \\ \rho\varepsilon \frac{\text{sign}(-|e_{0-k}|)}{1+\varepsilon|e_{0-k}|} & \text{for the node with the smallest MSE} \\ 0 & \text{for all the remaining nodes.} \end{cases} \tag{61}$$

Under this situation, after the time instant i goes to infinity, the parameters h_{kl} for each neighbor node of node k can be obtained through (61) and the quantity h_{kl} will be deterministic and can be taken out from the expectation.

Finally, removing the random variables $\alpha_{kl}(i)$ and inserting (59), (60) into (57), the asymptotic values K_k^n for the SI-LMS algorithm are obtained as in (62).

$$K_k^n(\text{SI-LMS}) = \frac{\sum\limits_{l\in\mathcal{N}_k}(c_{kl}-h_{kl})^2\mu_l^2\mathcal{J}_{min-l}\lambda_l^n + 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}(c_{kl}-h_{kl})(c_{kq}-h_{kq})\mu_l\mu_q e_{0-l}e_{0-q}^*\lambda_{l,q}^n}{1-\sum\limits_{l\in\mathcal{N}_k}(c_{kl}-h_{kl})^2(1-\mu_l\lambda_l^n)^2 - 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}(c_{kl}-h_{kl})(c_{kq}-h_{kq})(1-\mu_l\lambda_l^n)(1-\mu_q\lambda_q^n)}. \tag{62}$$

At this point, the theoretical results are deterministic, and the MSE for SI–LMS algorithm is given by

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2\sum_{n=1}^{M}K_k^n(\text{SI-LMS}). \tag{63}$$

ES–RLS

For the proposed ES–RLS algorithm, we start from (11), after inserting (11) into (18), we have

$$\begin{aligned}\boldsymbol{\omega}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)}c_{kl}(i)\boldsymbol{\psi}_l(i+1)\\ &= \sum_{l\in\widehat{\Omega}_k(i)}c_{kl}(i)[\boldsymbol{\omega}_l(i)+\boldsymbol{k}_l(i+1)e_l^*(i+1)]\\ &= \sum_{l\in\widehat{\Omega}_k(i)}c_{kl}(i)[\boldsymbol{\omega}_l(i)+\boldsymbol{k}_l(i+1)(d_l(i+1)-\boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))].\end{aligned} \tag{64}$$

Then, subtracting the $\boldsymbol{\omega}_0$ from both sides of (48), we arrive at

$$\begin{aligned}\boldsymbol{\varepsilon}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)}c_{kl}(i)[\boldsymbol{\omega}_l(i)+\boldsymbol{k}_l(i+1)(d_l(i+1)-\boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))]\\ &\quad-\sum_{l\in\widehat{\Omega}_k(i)}c_{kl}(i)\boldsymbol{\omega}_0\\ &= \sum_{l\in\widehat{\Omega}_k(i)}c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i)+\boldsymbol{k}_l(i+1)\big(d_l(i+1)-\boldsymbol{x}_l^H(i+1)(\boldsymbol{\varepsilon}_l(i)+\boldsymbol{\omega}_0)\big)\Big]\\ &= \sum_{l\in\widehat{\Omega}_k(i)}c_{kl}(i)\Big[\big(\boldsymbol{I}-\boldsymbol{k}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i)+\boldsymbol{k}_l(i+1)e_{0-l}^*(i+1)\Big].\end{aligned} \tag{65}$$

Then, with the random variables $\alpha_{kl}(i)$, (65) can be rewritten as

$$\boldsymbol{\varepsilon}_k(i+1) = \sum_{l\in\mathcal{N}_k} \alpha_{kl}(i)c_{kl}(i)\Big[\big(\boldsymbol{I} - \boldsymbol{k}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i) + \boldsymbol{k}_l(i+1)e_{0-l}^*(i+1)\Big]. \tag{66}$$

Since $\boldsymbol{k}_l(i+1) = \boldsymbol{\Phi}_l^{-1}(i+1)\boldsymbol{x}_l(i+1)$ [25], we can modify the (66) as

$$\begin{aligned}\boldsymbol{\varepsilon}_k(i+1) = \sum_{l\in\mathcal{N}_k} \alpha_{kl}(i)c_{kl}(i)\Big[&\big(\boldsymbol{I} - \boldsymbol{\Phi}_l^{-1}(i+1)\boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i) \\ &+ \boldsymbol{\Phi}_l^{-1}(i+1)\boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big].\end{aligned} \tag{67}$$

At this point, if we compare (67) with (51), we can find that the difference between (67) and (51) is, the $\boldsymbol{\Phi}_l^{-1}(i+1)$ in (67) has replaced the μ_l in (51). From [25], we also have

$$\mathbb{E}[\boldsymbol{\Phi}_l^{-1}(i+1)] = \frac{1}{i-M}\boldsymbol{R}_l^{-1}(i+1) \quad \text{for } i > M+1. \tag{68}$$

As a result, we can arrive

$$\begin{aligned}\boldsymbol{K}_k(i+1) = &\sum_{l\in\mathcal{N}_k} \mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Big(\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l}{i-M}\big)\boldsymbol{K}_l(i)\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l\boldsymbol{\Lambda}_l^{-1}}{i-M}\big) \qquad (69)\\ &+ \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l\boldsymbol{\Lambda}_l^{-1}}{(i-M)^2}e_{0-l}(i+1)e_{0-l}^*(i+1)\Big) + \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ &\times\Big(\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l}{i-M}\big)\boldsymbol{K}_{l,q}(i)\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_q\boldsymbol{\Lambda}_q^{-1}}{i-M}\big) + \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_{l,q}\boldsymbol{\Lambda}_q^{-1}}{(i-M)^2}e_{0-l}(i+1)\\ &\times e_{0-q}^*(i+1)\Big) + \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ &\times\Big(\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_q\boldsymbol{\Lambda}_q^{-1}}{i-M}\big)\boldsymbol{K}_{l,q}^H(i)\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l}{i-M}\big) + \frac{\boldsymbol{\Lambda}_q^{-1}\boldsymbol{\Lambda}_{l,q}^H\boldsymbol{\Lambda}_l^{-1}}{(i-M)^2}e_{0-q}(i+1)e_{0-l}^*(i+1)\Big). \qquad (70)\end{aligned}$$

Due to the structure of the above equations, the approximations and the quantities involved, we can decouple (70) into

$$\begin{aligned}K_k^n(i+1) = &\sum_{l\in\mathcal{N}_k} \mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Big(\big(1 - \frac{1}{i-M}\big)^2K_l^n(i) + \frac{e_{0-l}(i+1)e_{0-l}^*(i+1)}{\lambda_l^n(i-M)^2}\Big)\\ &+ \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\Big(\big(1 - \frac{1}{i-M}\big)^2K_{l,q}^n(i)\\ &+ \frac{\lambda_{l,q}^n e_{0-l}(i+1)e_{0-q}^*(i+1)}{(i-M)^2\lambda_l^n\lambda_q^n}\Big) + \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ &\times\Big(\big(1 - \frac{1}{i-M}\big)^2(K_{l,q}^n(i))^H + \frac{\lambda_{l,q}^n e_{0-q}(i+1)e_{0-l}^*(i+1)}{(i-M)^2\lambda_q^n\lambda_l^n}\Big) \qquad (71)\end{aligned}$$

where $K_k^n(i+1)$ is the nth elements of the main diagonals of $\boldsymbol{K}_k(i+1)$. With the assumption that, upon convergence, α_{kl} and c_{kl} do not vary, because at steady state, the choice of subset $\widehat{\Omega}_k(i)$ for each node k will be fixed, we can rewrite (71) as (72). Then, the MSE is given by

$$K_k^n(\text{ES-RLS}) = \frac{\sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2\frac{\mathcal{J}_{min-l}}{\lambda_l^n(i-M)^2} + 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}\frac{\lambda_{l,q}^n e_{0-l}e_{0-q}^*}{(i-M)^2\lambda_l^n\lambda_q^n}}{1-\sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2\Big(1-\frac{1}{i-M}\Big)^2 - 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}\Big(1-\frac{1}{i-M}\Big)^2}. \tag{72}$$

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2\sum_{n=1}^{M}K_k^n(\text{ES-RLS}). \tag{73}$$

On the basis of (72), we have that when i tends to infinity, the MSE approaches the MMSE in theory [25].

SI–RLS

For the proposed SI–RLS algorithm, we insert (59) into (72), remove the random variables $\alpha_{kl}(i)$, and following the same procedure as for the SI–LMS algorithm, we can obtain (74), where h_{kl} and h_{kq} satisfy the rule in (61). Then, the MSE is given by

$$K_k^n(\text{SI-RLS}) = \frac{\sum\limits_{l\in\mathcal{N}_k}(c_{kl}-h_{kl})^2\frac{\mathcal{J}_{min-l}}{\lambda_l^n(i-M)^2} + 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}(c_{kl}-h_{kl})(c_{kq}-h_{kq})\frac{\lambda_{l,q}^n e_{0-l}e_{0-q}^*}{(i-M)^2\lambda_l^n\lambda_q^n}}{1-\sum\limits_{l\in\mathcal{N}_k}(c_{kl}-h_{kl})^2\Big(1-\frac{1}{i-M}\Big)^2 - 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}(c_{kl}-h_{kl})(c_{kq}-h_{kq})\Big(1-\frac{1}{i-M}\Big)^2}. \tag{74}$$

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2\sum_{n=1}^{M}K_k^n(\text{SI-RLS}). \tag{75}$$

In conclusion, according to (62) and (74), with the help of modified combining coefficients, for the proposed SI–type algorithms, the neighbor node with lowest MSE contributes the most to the combination, while the neighbor node with the highest MSE contributes the least. Therefore, the proposed SI–type algorithms perform better than the standard diffusion algorithms with fixed combining coefficients.

Tracking Analysis

In this subsection, we assess the proposed ES–LMS/RLS and SI–LMS/RLS algorithms in a non–stationary environment, in which the algorithms have to track the minimum point of the error–performance surface [37, 38]. In the time–varying scenarios of interest, the optimum estimate is assumed to vary according to the model $\boldsymbol{\omega}_0(i+1) = \beta\boldsymbol{\omega}_0(i) + \boldsymbol{q}(i)$, where $\boldsymbol{q}(i)$ denotes a random perturbation [35] and $\beta = 1$ in order to facilitate the analysis. This is typical in the context of tracking analysis of adaptive algorithms [35, 25, 39, 40].

ES–LMS

For the tracking analysis of the ES–LMS algorithm, we employ *Assumption III* and start from (48). After subtracting the $\boldsymbol{\omega}_0(i+1)$ from both sides of (48), we obtain

$$\begin{aligned}\boldsymbol{\varepsilon}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i)+\mu_l\boldsymbol{x}_l(i+1)(d_l(i+1)-\boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))] \\ &\quad - \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\omega}_0(i+1) \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i)+\mu_l\boldsymbol{x}_l(i+1)(d_l(i+1) \\ &\qquad - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))] - \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big(\boldsymbol{\omega}_0(i)+\boldsymbol{q}(i)\Big) \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)\big(d_l(i+1)-\boldsymbol{x}_l^H(i+1)(\boldsymbol{\varepsilon}_l(i)+\boldsymbol{\omega}_0)\big)\Big]-\boldsymbol{q}(i) \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\big(\boldsymbol{I}-\mu_l\boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big]-\boldsymbol{q}(i).\end{aligned} \tag{76}$$

Using *Assumption III*, we can arrive at

$$\mathcal{J}_{ex-k}(i+1) = \mathrm{tr}\{\boldsymbol{R}_k(i+1)\boldsymbol{K}_k(i+1)\} + \mathrm{tr}\{\boldsymbol{R}_k(i+1)\boldsymbol{Q}\}. \tag{77}$$

The first part on the right side of (77), has already been obtained in the MSE steady–state analysis part in Section IV B. The second part can be decomposed as

$$\begin{aligned}\mathrm{tr}\{\boldsymbol{R}_k(i+1)\boldsymbol{Q}\} &= \mathrm{tr}\Big\{\mathbb{E}\big[\boldsymbol{x}_k(i+1)\boldsymbol{x}_k^H(i+1)\big]\mathbb{E}\big[\boldsymbol{q}(i)\boldsymbol{q}^H(i)\big]\Big\} \\ &= M\sigma_{x,k}^2\mathrm{tr}\{\boldsymbol{Q}\}.\end{aligned} \tag{78}$$

The MSE is then obtained as

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2\sum_{n=1}^{M} K_k^n(\text{ES-LMS}) + M\sigma_{x,k}^2\mathrm{tr}\{\boldsymbol{Q}\}. \tag{79}$$

SI–LMS

For the SI–LMS recursions, we follow the same procedure as for the ES-LMS algorithm, and obtain

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2\sum_{n=1}^{M} K_k^n(\text{SI-LMS}) + M\sigma_{x,k}^2\mathrm{tr}\{\boldsymbol{Q}\}. \tag{80}$$

ES–RLS

For the SI–RLS algorithm, we follow the same procedure as for the ES–LMS algorithm and arrive at

$$\mathcal{J}_{mse-k}(i+1) = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(i+1)(\text{ES-RLS}) + M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}. \tag{81}$$

SI–RLS

We start from (75), and after a similar procedure to that of the SI–LMS algorithm, we have

$$\mathcal{J}_{mse-k}(i+1) = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(i+1)(\text{SI-RLS}) + M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}. \tag{82}$$

In conclusion, for time-varying scenarios there is only one additional term $M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}$ in the MSE expression for all algorithms, and this additional term has the same value for all algorithms. As a result, the proposed SI–type algorithms still perform better than the standard diffusion algorithms with fixed combining coefficients, according to the conclusion obtained in the previous subsection.

Computational Complexity

In the analysis of the computational cost of the algorithms studied, we assume complex-valued data and first analyze the adaptation step. For both ES–LMS/RLS and SI–LMS/RLS algorithms, the adaptation cost depends on the type of recursions (LMS or RLS) that each strategy employs. The details are shown in Table 5.

Table 5 Computational complexity for the adaptation step per node per time instant

Adaptation Method	Multiplications	Additions
LMS	$2M+1$	$2M$
RLS	$4M^2+16M+2$	$4M^2+12M-1$

In the combination step, we analyze the computational complexity in Table 6. The overall complexity for each algorithm is summarized in Table 7. In the above three tables, t is the number of nodes chosen from $|\mathcal{N}_k|$ and M is the length of the unknown vector $\boldsymbol{\omega}_0$. The proposed algorithms require extra computations as compared to the existing distributed LMS and RLS algorithms. This extra cost ranges from a small additional number of operations for the SI-LMS/RLS algorithms to a more significant extra cost that depends on $|\mathcal{N}_k|$ for the ES-LMS/RLS algorithms.

Table 6 Computational complexity for the combination step per node per time instant

Algorithms	Multiplications	Additions
ES–LMS/RLS	$M(t+1)\frac{\lvert\mathcal{N}_k\rvert!}{t!(\lvert\mathcal{N}_k\rvert-t)!}$	$Mt\frac{\lvert\mathcal{N}_k\rvert!}{t!(\lvert\mathcal{N}_k\rvert-t)!}$
SI–LMS/RLS	$(2M+4)\lvert\mathcal{N}_k\rvert$	$(M+2)\lvert\mathcal{N}_k\rvert$

Simulations

In this section, we investigate the performance of the proposed link selection strategies for distributed estimation in two scenarios: wireless sensor networks and smart grids. In

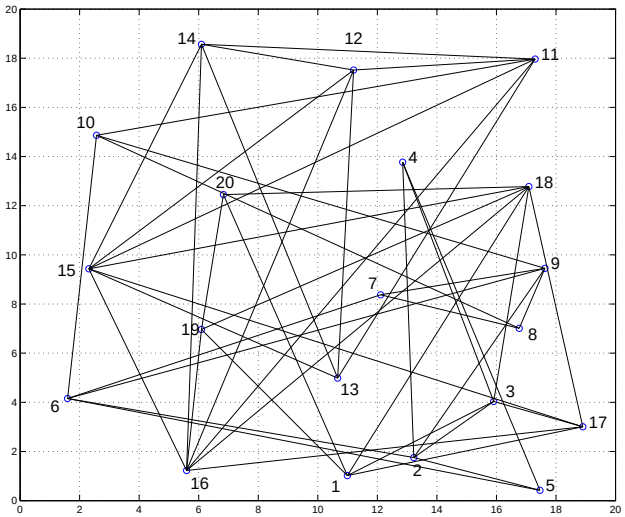
Table 7 Computational complexity per node per time instant

Algorithm	Multiplications	Additions
ES–LMS	$\left[\frac{(t+1)\lvert\mathcal{N}_k\rvert!}{t!(\lvert\mathcal{N}_k\rvert-t)!}+8\right]M+2$	$\left[\frac{\lvert\mathcal{N}_k\rvert!}{(t-1)!(\lvert\mathcal{N}_k\rvert-t)!}+8\right]M$
ES–RLS	$4M^2+\left[\frac{(t+1)\lvert\mathcal{N}_k\rvert!}{t!(\lvert\mathcal{N}_k\rvert-t)!}+16\right]M+2$	$4M^2+\left[\frac{\lvert\mathcal{N}_k\rvert!}{(t-1)!(\lvert\mathcal{N}_k\rvert-t)!}+12\right]M-1$
SI–LMS	$(8+2\lvert\mathcal{N}_k\rvert)M+4\lvert\mathcal{N}_k\rvert+2$	$(8+\lvert\mathcal{N}_k\rvert)M+2\lvert\mathcal{N}_k\rvert$
SI–RLS	$4M^2+(16+2\lvert\mathcal{N}_k\rvert)M+4\lvert\mathcal{N}_k\rvert+2$	$4M^2+(12+\lvert\mathcal{N}_k\rvert)M+2\lvert\mathcal{N}_k\rvert-1$

these applications, we simulate the proposed link selection strategies in both static and time–varying scenarios. We also show the analytical results for the MSE steady–state and tracking performances that we obtained in Section IV.

Diffusion Wireless Sensor Networks

Figure 5 Wireless sensor networks topology with 20 nodes

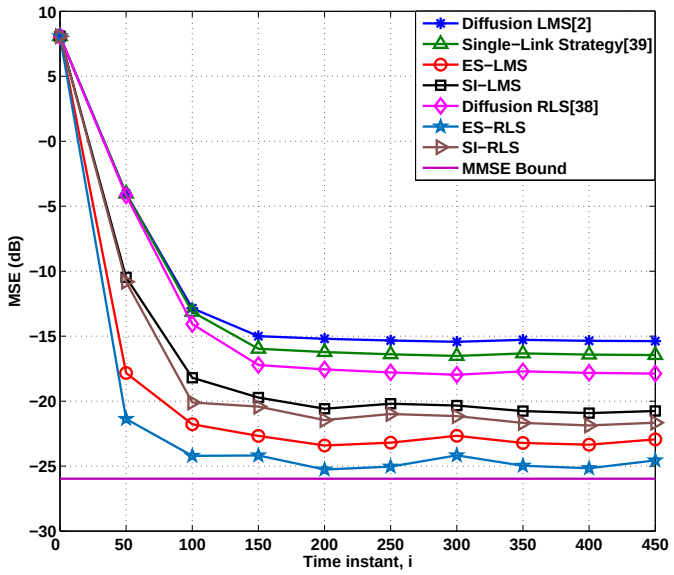


In this subsection, we compare the proposed ES–LMS/ES–RLS and SI–LMS/SI–RLS algorithms with the diffusion LMS algorithm [2], the diffusion RLS algorithm [41] and the single–link strategy [42] in terms of their MSE performance. A reduced–communication diffusion LMS algorithm with a performance comparable or worse to the standard diffusion LMS algorithm, which has been reported in [43] may also be considered if a designer needs to reduce the required bandwidth.

The network topology is illustrated in Fig. 5 and we employ $N = 20$ nodes in the simulations. The average node degree of the wireless sensor network is 5. The length of the unknown parameter vector $\boldsymbol{\omega}_0$ is $M = 10$ and it is generated as a complex random vector. The input signal is generated as $\boldsymbol{x}_k(i) = [x_k(i) \quad x_k(i-1) \quad \ldots \quad x_k(i-M+1)]$ and $x_k(i) = u_k(i) + \alpha_k x_k(i-1)$, where α_k is a correlation coefficient and $u_k(i)$ is a white noise process with variance $\sigma_{u,k}^2 = 1 - |\alpha_k|^2$, to ensure the variance of $\boldsymbol{x}_k(i)$ is $\sigma_{x,k}^2 = 1$. The $\boldsymbol{x}_k(0)$ is defined as a Gaussian randan number with zero mean and variance $\sigma_{x,k}^2$. The noise samples are modeled as circular Gaussian noise with zero mean and

variance $\sigma_{n,k}^2 \in [0.001, 0.01]$. The step size for the diffusion LMS ES–LMS and SI–LMS algorithms is $\mu = 0.2$. For the diffusion RLS algorithm, both ES–RLS and SI–RLS, the forgetting factor λ is set to 0.97 and δ is equal to 0.81. In the static scenario, the sparsity parameters of the SI–LMS/SI–RLS algorithms are set to $\rho = 4 \times 10^{-3}$ and $\varepsilon = 10$. The Metropolis rule is used to calculate the combining coefficients c_{kl}. The MSE and MMSE are defined as in (3) and (46), respectively. The results are averaged over 100 independent runs.

Figure 6 Network MSE curves in a static scenario



In Fig. 6, we can see that ES–RLS has the best performance for both steady-state MSE and convergence rate, and obtains a gain of about 8 dB over the standard diffusion RLS algorithm. SI–RLS is worse than the ES–RLS, but is still significantly better than the standard diffusion RLS algorithm by about 5 dB. Regarding the complexity and processing time, SI–RLS is as simple as the standard diffusion RLS algorithm, while ES–RLS is more complex. The proposed ES–LMS and SI–LMS algorithms are superior to the standard diffusion LMS algorithm.

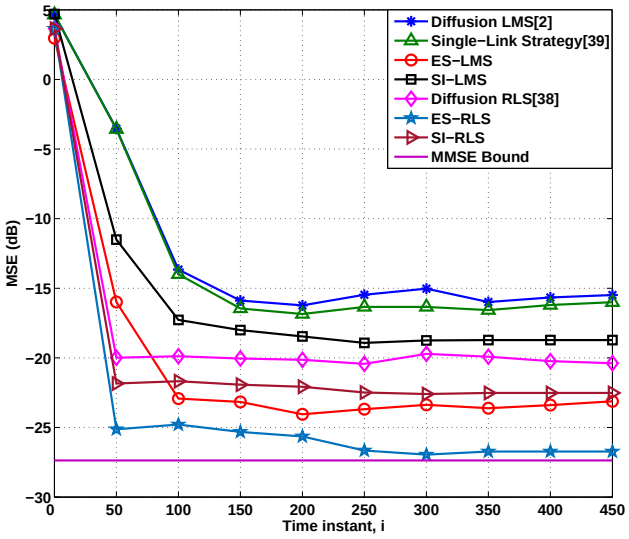
In the time–varying scenario, the sparsity parameters of the SI–LMS and SI–RLS algorithms are set to $\rho = 6 \times 10^{-3}$ and $\varepsilon = 10$. The unknown parameter vector $\boldsymbol{\omega}_0$ varies according to the first–order Markov vector process:

$$\boldsymbol{\omega}_0(i+1) = \beta \boldsymbol{\omega}_0(i) + \boldsymbol{q}(i), \tag{83}$$

where $\boldsymbol{q}(i)$ is an independent zero–mean Gaussian vector process with variance $\sigma_q^2 = 0.01$ and $\beta = 0.9998$.

Fig. 7 shows that, similarly to the static scenario, ES–RLS has the best performance, and obtains a 5 dB gain over the standard diffusion RLS algorithm. SI–RLS is slightly worse than the ES–RLS, but is still better than the standard diffusion RLS algorithm by about 3 dB. The proposed ES–LMS and SI–LMS algorithms have the same advantage

Figure 7 Network MSE curves in a time—varying scenario



over the standard diffusion LMS algorithm in the time-varying scenario. Notice that in the scenario with large $|\mathcal{N}_k|$, the proposed SI-type algorithms still have a better performance when compared with the standard techniques.

To illustrate the link selection for the ES–type algorithms, we provide Figs. 8 and 9. From these two figures, we can see that upon convergence the proposed algorithms converge to a fixed selected set of links $\widehat{\Omega}_k$.

MSE Analytical Results

The aim of this section is to validate the analytical results obtained in Section IV. First, we verify the MSE steady–state performance. Specifically, we compare the analytical results in (58), (63), (73) and (75) to the results obtained by simulations under different SNR values. The SNR indicates the input signal variance to noise variance ratio. We assess the MSE against the SNR, as show in Figs. 10 and 11. For ES–RLS and SI–RLS algorithms, we use (73) and (75) to compute the MSE after convergence. The results show that the analytical curves coincide with those obtained by simulations, which indicates the validity of the analysis. We have assessed the proposed algorithms with SNR equal to 0dB, 10dB, 20dB and 30dB, respectively, with 20 nodes in the network. For the other parameters, we follow the same definitions used to obtain the network MSE curves in a static scenario. The details have been shown on the top of each sub figure in Figs. 10 and 11. The theoretical curves for ES–LMS/RLS and SI–LMS/RLS are all close to the simulation curves.

The tracking analysis of the proposed algorithms in a time–varying scenario is discussed as follows. Here, we verify that the results (79), (80), (81) and (82) of the subsection on the tracking analysis can provide a means of estimating the MSE. We consider the same model as in (83), but with β is set to 1. In the next examples, we employ $N = 20$ nodes in the network and the same parameters used to obtain the network MSE curves in a time–varying scenario. A comparison of the curves obtained by simulations and by the analytical formulas is shown in Figs. 12 and 13. From these curves, we can verify that the gap between

Figure 8 Set of selected links for node 16 with ES–LMS in a static scenario

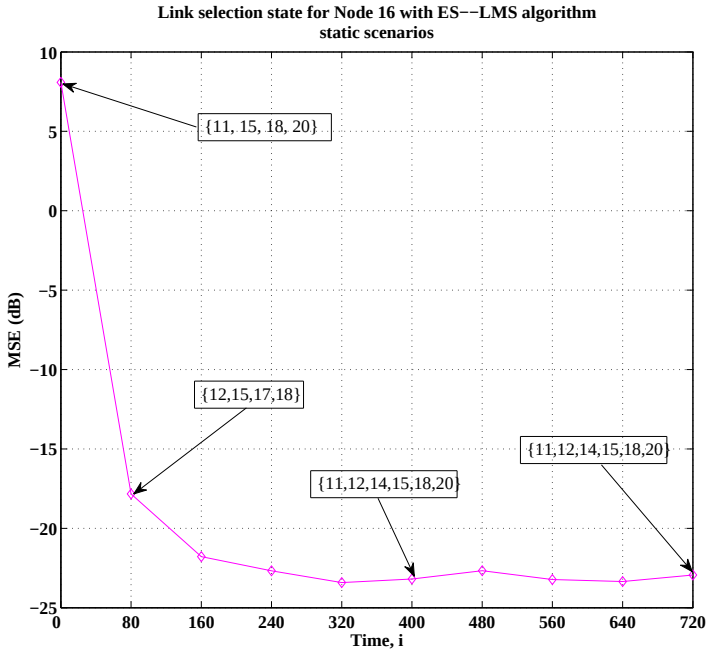
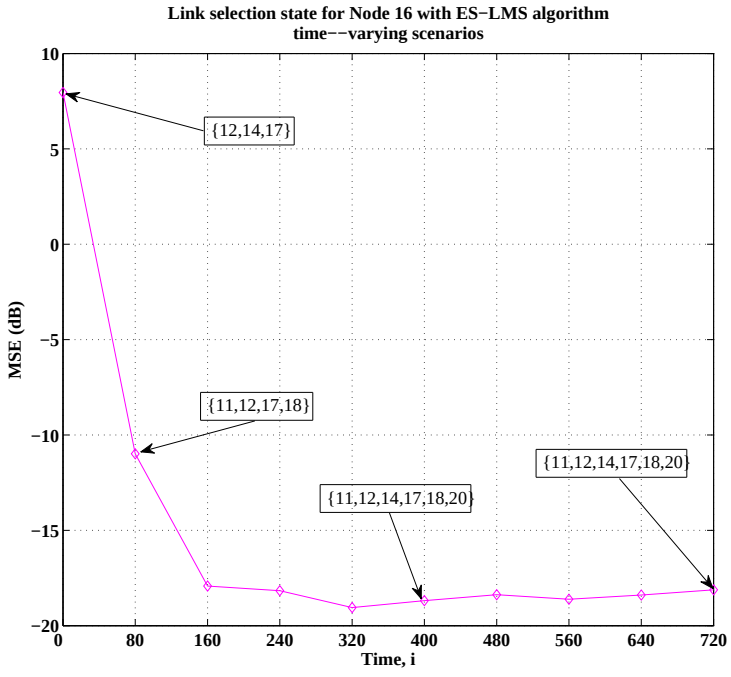


Figure 9 Link selection state for node 16 with ES–LMS in a time—varying scenario



the simulation and analysis results are extraordinary small under different SNR values. The details of the parameters are shown on the top of each sub figure in Figs. 12 and 13.

Figure 10 MSE performance against SNR for ES–LMS and SI–LMS

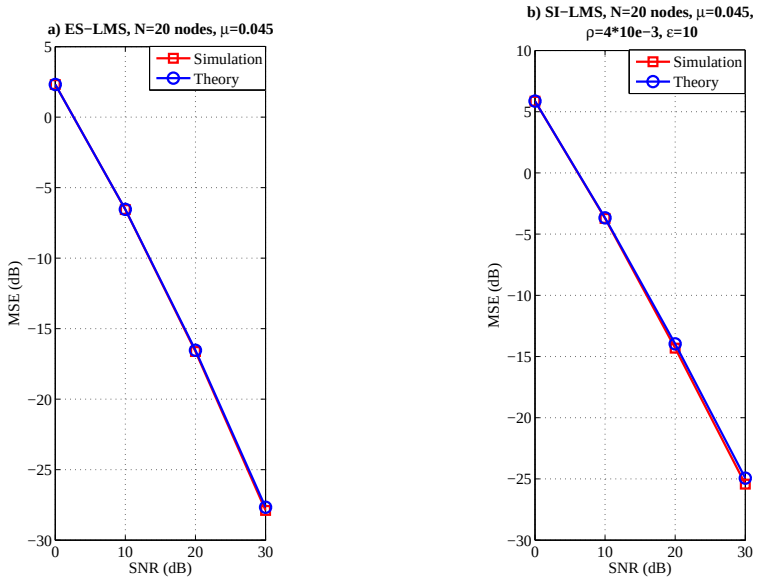
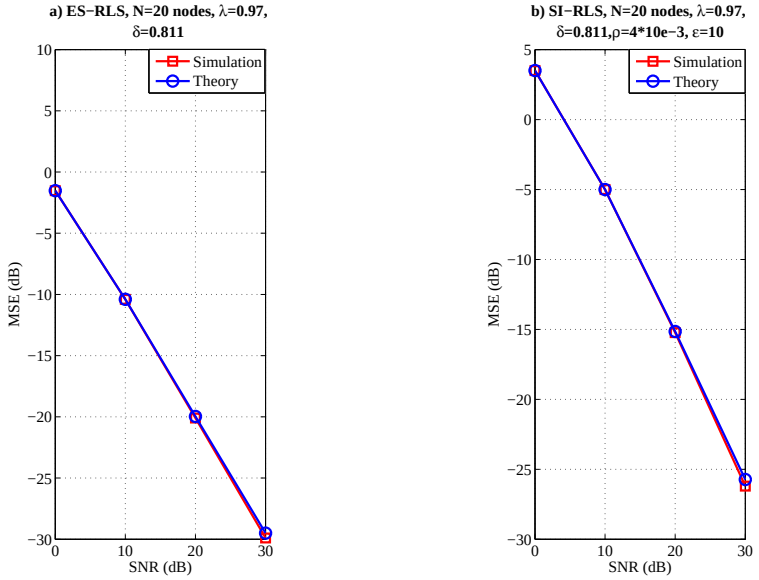


Figure 11 MSE performance against SNR for ES–RLS and SI–RLS



Smart Grids

The proposed algorithms provide a cost–effective tool that could be used for distributed state estimation in smart grid applications. In order to test the proposed algorithms in a possible smart grid scenario, we consider the IEEE 14–bus system [44], where 14 is the number of substations. At every time instant i, each bus $k, k = 1, 2, \ldots, 14$, takes a scalar

Figure 12 MSE performance against SNR for ES–LMS and SI–LMS in a time—varying scenario

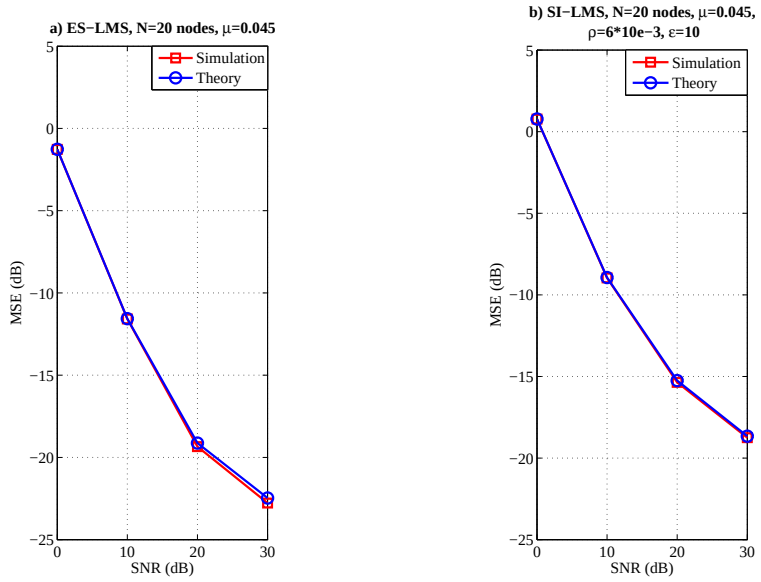
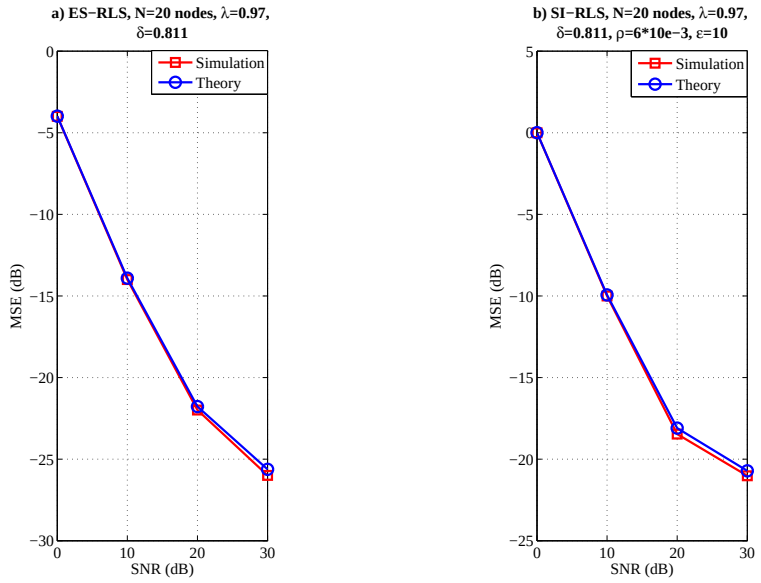


Figure 13 MSE performance against SNR for ES–RLS and SI–RLS in a time—varying scenario



measurement $d_k(i)$ according to

$$d_k(i) = X_k\big(\boldsymbol{\omega}_0(i)\big) + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{84}$$

where $\boldsymbol{\omega}_0(i)$ is the state vector of the entire interconnected system, and $X_k(\boldsymbol{\omega}_0(i))$ is a nonlinear measurement function of bus k. The quantity $n_k(i)$ is the measurement error with mean equal to zero and which corresponds to bus k.

Initially, we focus on the linearized DC state estimation problem. The state vector $\boldsymbol{\omega}_0(i)$ is taken as the voltage phase angle vector $\boldsymbol{\omega}_0$ for all buses. Therefore, the nonlinear measurement model for state estimation (84) is approximated by

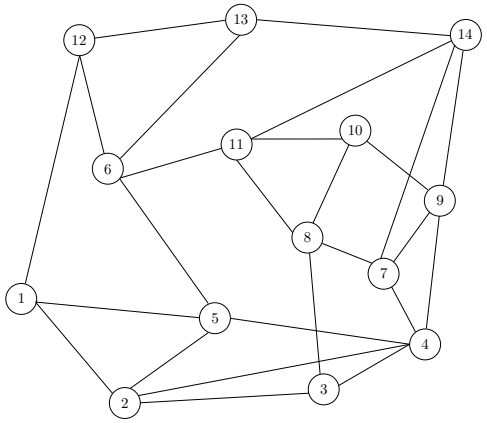
$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{85}$$

where $\boldsymbol{x}_k(i)$ is the measurement Jacobian vector for bus k. Then, the aim of the distributed estimation algorithm is to compute an estimate of $\boldsymbol{\omega}_0$, which can minimize the cost function given by

$$J_k(\boldsymbol{\omega}_k(i)) = \mathbb{E}|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)|^2. \tag{86}$$

We compare the proposed algorithms with the M–CSE algorithm [4], the single link strategy [42], the standard diffusion RLS algorithm [41] and the standard diffusion LMS algorithm [2] in terms of MSE performance. The MSE comparison is used to determine the accuracy of the algorithms, and compare the rate of convergence. We define the IEEE–14 bus system as in Fig. 14.

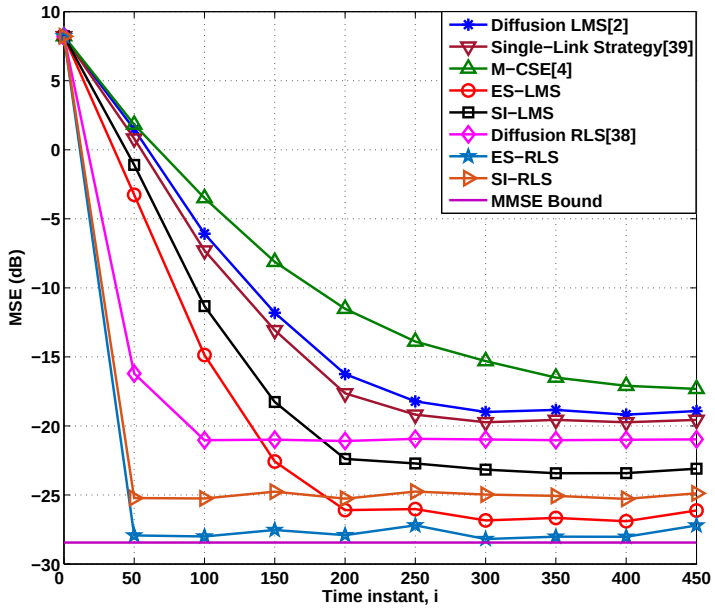
Figure 14 IEEE 14–bus system for simulation



All buses are corrupted by additive white Gaussian noise with variance $\sigma_{n,k}^2 \in [0.001, 0.01]$. The step size for the standard diffusion LMS [2], the proposed ES–LMS and SI–LMS algorithms is 0.15. The parameter vector $\boldsymbol{\omega}_0$ is set to an all–one vector. For the diffusion RLS, ES–RLS and SI–RLS algorithms the forgetting factor λ is set to 0.945 and δ is equal to 0.001. The sparsity parameters of the SI–LMS/RLS algorithms are set to $\rho = 0.07$ and $\varepsilon = 10$. The results are averaged over 100 independent runs. We simulate the proposed algorithms for smart grids under a static scenario.

From Fig. 15, it can be seen that ES–RLS has the best performance, and significantly outperforms the standard diffusion LMS [2] and the $\mathcal{M}$–$\mathcal{CSE}$ [4] algorithms. The ES–LMS is slightly worse than ES–RLS, which outperforms the remaining techniques. SI–RLS is worse than ES–LMS but is still better than SI–LMS, while SI–LMS remains better than the diffusion RLS, LMS, $\mathcal{M}$–$\mathcal{CSE}$ algorithms and the single link strategy.

Figure 15 MSE performance curves for smart grids



Conclusions

In this paper, we have proposed ES–LMS/RLS and SI–LMS/RLS algorithms for distributed estimation in applications such as wireless sensor networks and smart grids. We have compared the proposed algorithms with existing methods. We have also devised analytical expressions to predict their MSE steady–state performance and tracking behavior. Simulation experiments have been conducted to verify the analytical results and illustrate that the proposed algorithms significantly outperform the existing strategies, in both static and time–varying scenarios, in examples of wireless sensor networks and smart grids.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This research was supported in part by the U. S. National Science Foundation under Grants CCF-1420575, CNS-1456793 and DMS-1118605.
Part of this work has been presented at the 2013 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vancouver, Canada and 2013 IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, Saint Martin.
The authors wish to thank the anonymous reviewers, whose comments and suggestions have greatly improved the presentation of these results.

Author details
[1]Department of Electronics, University of York, YO10 5DD York, UK. [2]CETUC, PUC-Rio, Brazil. [3]Department of Electrical Engineering, Princeton University, 08544 Princeton NJ, USA.

References

1. Lopes, C.G., Sayed, A.H.: Incremental adaptive strategies over distributed networks. IEEE Trans. Signal Process. **48**(8), 223–229 (2007)
2. Lopes, C.G., Sayed, A.H.: Diffusion least–mean squares over adaptive networks: Formulation and performance analysis. IEEE Trans. Signal Process. **56**(7), 3122–3136 (2008)
3. Chen, Y., Gu, Y., Hero, A.O.: Sparse LMS for system identification. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3125–3128 (2009)
4. Xie, L., Choi, D.-H., Kar, S., Poor, H.V.: Fully distributed state estimation for wide-area monitoring systems. IEEE Trans. Smart Grid **3**(3), 1154–1169 (2012)

5. Huang, Y.-H., Werner, S., Huang, J., N. Kashyap, V.G.: State estimation in electric power grids: meeting new challenges presented by the requirements of the future grid. IEEE Signal Process. Mag. **29**(5), 33–43 (2012)
6. Bertsekas, D.: A new class of incremental gradient methods for least squares problems. SIAM J. Optim **7**(4), 913–926 (1997)
7. Nedic, A., Bertsekas, D.: Incremental subgradient methods for nondifferentiable optimization. SIAM J. Optim **12**(1), 109–138 (2001)
8. Rabbat, M.G., Nowak, R.D.: Quantized incremental algorithms for distributed optimization. IEEE J. Sel. Areas Commun **23**(4), 798–808 (2005)
9. Cattivelli, F.S., Sayed, A.H.: Diffusion LMS strategies for distributed estimation. IEEE Trans. Signal Process. **58**, 1035–1048 (2010)
10. Lorenzo, P.D., Barbarossa, S., Sayed, A.H.: Sparse diffusion LMS for distributed adaptive estimation. In: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 3281–3284 (2012)
11. Mateos, G., Schizas, I.D., Giannakis, G.B.: Distributed Recursive Least-Squares for Consensus-Based In-Network Adaptive Estimation. IEEE Trans. Signal Process. **57**(11), 4583–4588 (2009)
12. Arablouei, R., Doğançay, K., Werner, S., Huang, Y.-F.: Adaptive distributed estimation based on recursive least-squares and partial diffusion. IEEE Trans. Signal Process. **62**, 3510–3522 (2014)
13. Arablouei, R., Werner, S., Huang, Y.-F., Doğançay, K.: Distributed least mean–square estimation with partial diffusion. IEEE Trans. Signal Process. **62**, 472–484 (2014)
14. Lopes, C.G., Sayed, A.H.: Diffusion adaptive networks with changing topologies. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3285–3288 (Las Vegas, March 2008)
15. Fadlallah, B., Principe, J.: Diffusion Least-Mean squares over adaptive networks with dynamic topologies. In: Proc. IEEE International Joint Conference on Neural Networks (2013)
16. Tu, S.-Y., Sayed, A.H.: On the influence of informed agents on learning and adaptation over networks. IEEE Trans. Signal Process. **61**, 1339–1356 (2013)
17. Wimalajeewa, T., Jayaweera, S.K.: Distributed node selection for sequential estimation over noisy communication channels. IEEE Trans. Wirel. Commun. **9**(7), 2290–2301 (2010)
18. de Lamare, R.C., Sampaio-Neto, R.: Adaptive reduced-rank processing based on joint and iterative interpolation, decimation and filtering. IEEE Trans. Signal Process. **57**(7), 2503–2514 (2009)
19. de Lamare, R.C., Diniz, P.S.R.: Set-membership adaptive algorithms based on time-varying error bounds for CDMA interference suppression. IEEE Trans. Vehi. Techn. **58**(2), 644–654 (2009)
20. de Lamare, R.C., Sampaio-Neto, R., Haardt, M.: Blind adaptive constrained constant-modulus reduced-rank interference suppression algorithms based on interpolation and switched decimation. Signal Processing, IEEE Transactions on **59**(2), 681–695 (2011). doi:10.1109/TSP.2010.2091274
21. de Lamare, R.C., de Lamare, R.C.: Sparsity-aware adaptive algorithms based on alternating optimization and shrinkage. IEEE Signal Processing Letters **21**(2), 225–229 (2014)
22. Guo, L., Huang, Y.F.: Frequency-domain set-membership filtering and its applications. IEEE Trans. Signal Process. **55**(4), 1326–1338 (2007)
23. Xu, S., de Lamare, R.C., Poor, H.V.: Distributed compressed estimation based on compressive sensing. IEEE Signal Processing Letters **22**(9), 1311–1315 (2015)
24. Bertrand, A., Moonen, M.: Distributed adaptive node–specific signal estimation in fully connected sensor networks–part II: Simultaneous and asynchronous node updating. IEEE Trans. Signal Process. **58**(10), 5292–5306 (2010)
25. Haykin, S.: Adaptive Filter Theory, 4th edn. Prentice Hall, Upper Saddle River, NJ, USA (2002)
26. Li, L., Chambers, J.A.: Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology. In: Proc. IEEE/SP 15th Workshop on Statistical Signal Processing, pp. 757–760 (2009)
27. Zhao, X., Sayed, A.H.: Performance limits for distributed estimation over lms adaptive networks. IEEE Trans. Signal Process. **60**(10), 5107–5124 (2012)
28. Xiao, L., Boyd, S.: Fast linear iterations for distributed averaging. Syst. Control Lett. **53**(1), 65–78 (2004)
29. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Autom. Control. **49**, 1520–1533 (2004)
30. Jadbabaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. Autom. Control **48**(6), 988–1001 (2003)
31. Meng, R., de Lamare, R.C., Nascimento, V.H.: Sparsity-aware affine projection adaptive algorithms for system identification. In: Proc. Sensor Signal Processing for Defence (London, UK, September 2011)
32. Chen, Y., Gu, Y., Hero, A.: Regularized least-mean-square algorithms. Technical Report for AFOSR (2010)
33. Xu, S., de Lamare, R.C.: Distributed conjugate gradient strategies for distributed estimation over sensor networks. In: Proc. Sensor Signal Processing for Defence (London, UK, September 2012)
34. Cattivelli, F., Sayed, A.H.: Diffusion strategies for distributed kalman filtering and smoothing. IEEE Trans. Autom. Control **55**(9), 2069–2084 (2010)
35. Sayed, A.H.: Fundamentals of Adaptive Filtering. John Wiley&Sons, Hoboken, NJ, USA (2003)
36. Kailath, T., Sayed, A.H., Hassibi, B.: Linear Estimation. Prentice-Hall, Englewood Cliffs, NJ, USA (2000)
37. de Lamare, R.C., Diniz, P.S.R.: Blind adaptive interference suppression based on set-membership constrained constant-modulus algorithms with dynamic bounds. IEEE Trans. Signal Process. **61**(5), 1288–1301 (2013)
38. Cai, Y., de Lamare, R.C.: Low–complexity variable step-size mechanism for code-constrained constant modulus stochastic gradient algorithms applied to cdma interference suppression. IEEE Trans. Signal Process. **57**(1), 313–323 (2009)
39. Widrow, B., Stearns, S.D.: Adaptive Signal Processing. Prentice-Hall, Englewood Cliffs, NJ, USA (1985)
40. Eweda, E.: Comparison of RLS, LMS, and sign algorithms for tracking randomly time-varying channels. IEEE Trans. Signal Process. **42**, 2937–2944 (1994)
41. Cattivelli, F.S., Lopes, C.G., Sayed, A.H.: Diffusion recursive least-squares for distributed estimation over adaptive networks. IEEE Trans. Signal Process. **56**(5), 1865–1877 (2008)
42. Zhao, X., Sayed, A.H.: Single-link diffusion strategies over adaptive networks. In: Proc. IEEE International

Conference on Acoustics, Speech and Signal Processing, pp. 3749–3752 (2012)
43. Arablouei, R., Werner, S., Doğançay, K., Huang, Y.-F.: Analysis of a reduced–communication diffusion LMS algorithm. Signal Processing **117**, 355–361 (2015)
44. Bose, A.: Smart transmission grid applications and their supporting infrastructure. IEEE Trans. Smart Grid **1**(1), 11–19 (2010)