Signal Processing 163 (2019) 107–114

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Short communication

Self-regularized nonlinear diffusion algorithm based on levenberg gradient descent☆

Lu Lu[a], Zongsheng Zheng[b], Benoit Champagne[c], Xiaomin Yang[a,*], Wei Wu[a]

[a] School of Electronics and Information Engineering, Sichuan University, Chengdu, Sichuan 610065, China
[b] School of Electrical Engineering, Southwest Jiaotong University, Chengdu, Sichuan 610031, China
[c] Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A0E9, Canada

ARTICLE INFO

Article history:
Received 11 September 2018
Revised 16 April 2019
Accepted 15 May 2019
Available online 16 May 2019

Keywords:
Nonlinear diffusion algorithm
Power filter
Levenberg gradient descent (LGD)
Self-regularization (SR)

ABSTRACT

In this letter, the nonlinear identification problem for distributed in-network systems using diffusion based adaptation is addressed. The contribution is threefold: First, a so-called power diffusion algorithm is proposed that adopts integer powers of the input as regressors, as a means to improve identification accuracy with low computational complexity. Second, a modified version of the power diffusion algorithm, termed as power Levenberg diffusion (PLD) algorithm, is developed based on the Levenberg gradient descent (LGD) method to further improve performance. Finally, a new self-regularization (SR) strategy for the PLD algorithm is proposed to overcome issues with parameter selection during adaptation. The simulation results for in-network distributed nonlinear system identification reveal that the newly proposed algorithms can provide superior performance when compared to existing algorithms.

1. Introduction

In recent years, distributed adaptation schemes have received considerable attention, as they offer an efficient and robust means to estimate the parameters of a physical system from noisy measurements obtained through a network of sensors [1]. Compared to other distributed strategies, the diffusion techniques can achieve a stable behavior regardless of the network topology [2], which is seen as a key advantage and has motivated several applications, such as: classification [3], frequency estimation in power networks [4] and spectrum sensing [5]. Diffusion algorithms have also been developed for distributed in-network system identification, mainly assuming a linear structure while preserving the conceptual simplicity of the classical linear adaptive filters. These algorithms include the diffusion least mean square (DLMS) [6–8] and diffusion recursive least squares (DRLS) [9,10].

It has been reported that the conventional DLMS algorithm is not effective when nonlinearities are present in the distributed in-network system model. The nonlinearities are usually contributed by the system itself, i.e., nonlinear relationship between the system's input and output [11]. A few nonlinear diffusion algorithms have been introduced recently to improve the nonlinear modeling capability under such circumstances. In [12], a cost-effective framework for diffusion algorithm was proposed, which suggests that even in cases where the system is linear, the adaptive learning technique should be nonlinear. A diffusion kernel LMS algorithm was proposed as a suitable candidate for distributed in-network nonlinear system identification in [11,13]. However, the complexity of this scheme grows linearly with the number of processed observations, which hinders its practical applications. The nonlinear Volterra diffusion algorithm [14], which is based on second-order Volterra (SOV) series, can yield a small kernel misadjustment as an effective nonlinear identification approach. Another recently reported nonlinear diffusion adaptation scheme is the diffusion interpolated Volterra (DIV) algorithm, which can significantly reduce computational complexity and maintain robustness against impulsive interference [14]. These nonlinear Volterra diffusion algorithms with logarithmic least mean *p*th-power (LLMP) adaptation have potential applications in the area of distributed wireless sensor networks, where investigations reveal that the communication process is often plagued by nonlinear distortions and impulsive interferences at the node level [14].

☆ The work of L. Lu, X. Yang and W. Wu was supported in part by the National Science Foundation of P.R. China under grant 61701327 and China Postdoctoral Science Foundation Funded Project under Grant 2018M640916. Z. Zheng's work is supported by the China Scholarship Council (CSC) under grant 201807000035 and Doctoral Innovation Fund Program of Southwest Jiaotong University. B. Champagne's work is supported by a grant from NSERC, govt. of Canada.

* Corresponding author.
E-mail addresses: lulu19900303@126.com (L. Lu), bk20095185@my.swjtu.edu.cn (Z. Zheng), benoit.champagne@mcgill.ca (B. Champagne), arielyang@scu.edu.cn (X. Yang), wuwei@scu.edu.cn (W. Wu).

https://doi.org/10.1016/j.sigpro.2019.05.018
0165-1684/

The nonlinear diffusion algorithms discussed above are derived using the steepest descent method, which guarantees their convergence but at the price of a relatively long convergence time. The convergence rate of gradient descent type algorithms can be enhanced by using the Levenberg gradient descent (LGD) method [15,16], which combines the properties of the steepest descent and the Gauss-Newton methods. When the current solution is far from the exact one, LGD behaves like a steepest descent method and provides slow, but guaranteed convergence [16]. When the current solution is close to the correct solution, it becomes a Gauss-Newton method and achieves fast convergence. Because of these remarkable properties, LGD provides an effective search method for nonlinear systems and in recent years, LGD-based learning algorithms have received significant attention [17–20].

In this paper, to improve the performance of existing diffusion algorithms for distributed in-network nonlinear system identification, we first propose a so-called power diffusion algorithm, motivated by [21,22]. By adopting integer powers of the input as regressors, the proposed algorithm offers improved capabilities in terms of both convergence rate and implementation complexity. Second, a modified version of the power diffusion algorithm, termed as power Levenberg diffusion (PLD) algorithm, is proposed that combines the former with the LGD search method. The computational cost of the PLD algorithm is higher than that of the power diffusion algorithm, but it significantly reduces the steady-state misadjustment. As a last contribution, we further propose a new self-regularization (SR) strategy for the PLD algorithm to overcome the difficult selection of internal algorithm parameters during adaptation. Through simulations, we show that the proposed algorithms achieve better performance than the conventional DLMS and Volterra diffusion algorithms for nonlinear system identification.

The rest of the paper is organized as follows. In Section 2, we formulate the problem of distributed in-network nonlinear system identification based on the diffusion strategy. In Section 3, we present the proposed power diffusion and PLD algorithms in detail. In Section 4, the new SR scheme for the PLD algorithm is developed. Simulation results and accompanying discussions are presented in Section 5. Finally, Section 6 concludes this work.

2. Problem formulation

Let us consider a network of N sensors or nodes, indexed by $k \in \{1, \ldots, N\}$, which are distributed over some geographical area. The neighborhood of node k, i.e. the set of nodes linked to k by a direct communication path, is denoted by $\mathcal{N}_k$ (including k itself). At every time instant $i \in \mathbb{N}$, every node k has at its disposition an observation $\{\boldsymbol{x}_k(i), d_k(i)\}$, where $\boldsymbol{x}_k(i) \in \mathbb{R}^{M\times 1}$ denotes a data regressor vector, whose entries may be related in a nonlinear fashion to the inputs of the unknown nonlinear system under consideration, and $d_k(i) \in \mathbb{R}$ is a scalar measurement representing a noisy output from this same system. The system output at node k can be represented by

$$d_k(i) = \boldsymbol{h}_{\text{opt}}^{\mathrm{T}}\boldsymbol{x}_k(i) + v_k(i) \tag{1}$$

where $\boldsymbol{h}_{\text{opt}} \in \mathbb{R}^{M\times 1}$ is the parameter vector of the system, $(\cdot)^{\mathrm{T}}$ denotes transposition, and $v_k(i) \in \mathbb{R}$ is the additive measurement noise. Define the error signal at node k as

$$e_k(i) \triangleq d_k(i) - y_k(i) \tag{2}$$

where $y_k(i) = \boldsymbol{h}_k^{\mathrm{T}}(i-1)\boldsymbol{x}_k(i)$ is the local filter output and $\boldsymbol{h}_k(i-1)$ is the estimate of $\boldsymbol{h}_{\text{opt}}$ at time $i-1$. Below, we briefly recall the distinguishing features of the DLMS and Volterra diffusion algorithms, which provide the starting point for this work.

DLMS algorithm [6]: This algorithm seeks to minimize the cost functions $J_k^{\text{loc}}(\boldsymbol{h}) = \sum_{l\in\mathcal{N}_k} a_{l,k}\mathrm{E}\big(e_l^2(i)\big)$ at every nodes k, where $\mathrm{E}(\cdot)$ denotes statistical expectation. Its update equations, obtained by incorporating the LMS into the diffusion scheme, are given by[1]

$$\begin{cases} \boldsymbol{\varphi}_k(i) = \boldsymbol{h}_k(i-1) + \mu\boldsymbol{x}_k(i)e_k(i) & (\texttt{adaptation}) \\ \boldsymbol{h}_k(i) = \sum\limits_{l\in\mathcal{N}_k} a_{l,k}\boldsymbol{\varphi}_l(i) & (\texttt{combination}) \end{cases} \tag{3}$$

where μ is the step size (learning rate), $\boldsymbol{x}_k(i) = [x_k(i), x_k(i-1), \ldots, x_k(i-M+1)]^{\mathrm{T}}$, and $\boldsymbol{\varphi}_k(i)$ is the intermediate estimate at node k and time i. The weighting coefficients $\{a_{l,k}\}$ are real non-negative numbers such that $a_{l,k} = 0$ if $l \notin \mathcal{N}_k$.

Volterra diffusion algorithm [14]: In the quadratic case, this algorithm can be interpreted as a nonlinear extension of DLMS, with expanded input and coefficient vectors at node k defined as

$$\begin{aligned} \boldsymbol{x}_k(i) \triangleq [&x_k(i), \ldots, x_k(i-L+1), \\ &x_k^2(i), x_k(i)x_k(i-1), \ldots, x_k^2(i-L+1)]^{\mathrm{T}} \end{aligned} \tag{4}$$

$$\begin{aligned} \boldsymbol{h}_k(i) \triangleq [&h_{k,1}(0), h_{k,1}(1), \ldots, h_{k,1}(L-1), \\ &h_{k,2}(0,0), h_{k,2}(0,1), \ldots, h_{k,2}(L-1,L-1)]^{\mathrm{T}} \end{aligned} \tag{5}$$

where $h_{k,1}(m_1)$ and $h_{k,2}(m_1, m_2)$ respectively denote the first and second (triangular) Volterra kernels at node k, the kernel lag parameters $0 \le m_1 \le m_2 < L$, and L denote the length of the linear kernel, and the length of the SOV filter can be calculated as $M = L(L+3)/2$. While the adaptation and combination rules in (3) can be applied here, an alternative form of adaptation based on an improved cost function that is robust to impulsive noise is considered in [14].

3. Proposed algorithms

3.1. Power diffusion algorithm

In this section, the power diffusion algorithm is proposed, which can be regarded as a specialized form of the Volterra diffusion algorithm where the DLMS adaptation and combination rules are exploited. In particular, the input regressor vector of the power diffusion algorithm at node k is expressed as

$$\boldsymbol{x}_k(i) = [x_k(i), x_k^2(i), \ldots, x_k^M(i)]^{\mathrm{T}} \tag{6}$$

which is a memoryless nonlinear vector function (i.e., finite sequence of integer powers) of an instantaneous scalar input $x_k(i)$. The corresponding weight vector at node k is given by

$$\boldsymbol{h}_k(i) = [h_{k,1}(i), h_{k,2}(i), \ldots, h_{k,M}(i)]^{\mathrm{T}} \tag{7}$$

which is updated based on the adaptation and combinations rules in (3).

The main difference between the proposed power diffusion and Volterra diffusion algorithms is that the former does not include past measurements and related cross-terms in the power series expansion used to model the system output. In effect, the underlying motivation is to trade-off memory for nonlinear modeling capability: for a given length of coefficient vector, a higher order of non-linearity can be modeled without increasing computational complexity. Thus, if prior knowledge about the unknown system is available, the order of non-linearity can be adjusted (possibly differently at each node) for improved performance compared to the DLMS and Volterra diffusion algorithms, as illustrated in Section 6. Besides, due to the memoryless nature of regressor vector $\boldsymbol{x}_k(i)$, the temporal correlation properties of $x_k(i)$ have limited impact on the convergence rate of the algorithm (see the proof in [23]). Indeed, when moments of the type $\mathrm{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_l^{\mathrm{T}}(j)]$ with $i \neq j$ are considered

[1] In this work, we focus on the adapt-then-combine (ATC) implementation of the diffusion strategy, which has been shown to outperform the combine-then-adapt (CTA) implementation [6].

in the convergence analysis, the various entries in these matrices decorrelate at the same rate as a function of the difference $|i-j|$ due to the absence of memory. In turn, this lack of memory leads to faster converge of the adaptive learning rule. Consequently, the power diffusion algorithm is well suited for practical applications such as nonlinear acoustic echo cancellation and nonlinear system identification.

3.2. PLD algorithm

To further improve performance, we next introduce the PLD algorithm, which combines the power diffusion algorithm with the LGD method. The LGD method is characterized by its good nonlinear optimization capability, and can converge to the optimal solution in both linear and nonlinear problems [24]. Within the diffusion framework, the LGD method allows us to formulate the following update equation for the weight vector $\boldsymbol{h}_k(i)$ at node k:

$$\boldsymbol{h}_k(i) = \boldsymbol{h}_k(i-1) - \mu \sum_{l\in\mathcal{N}_k} a_{l,k}[\mathbf{G}_l(i)+\lambda\mathbf{I}]^{-1}\nabla\xi_l(i). \tag{8}$$

where $\mathbf{G}_l(i)$ stands for the Hessian matrix of the underlying objective function $\xi_l(i)$, $\mathbf{I}$ denotes the identity matrix, and λ is a regularization parameter which determines the desired trade-off between steepest descent and Gauss-Newton search. This method is derived from Newton's method and is often used in the context of adaptive linear neuron (ADALINE) structures [16]. The conventional Levenberg-Marquardt (LM) based-algorithms require the calculation of the Hessian matrix, which for a statistically based objective function entails high computational cost and memory requirements [17]. To overcome this problem, we directly calculate the Hessian matrix based on instantaneous estimation. Specifically, at each iteration, the objection function $\xi_l(i)$ at node l is defined by the squared error criterion

$$\xi_l(i) \triangleq \frac{1}{2}(e_l(i))^2 = \frac{1}{2}\left(d_l(i) - \boldsymbol{h}_l^{\mathrm{T}}(i-1)\boldsymbol{x}_l(i)\right)^2. \tag{9}$$

Taking partial derivatives of (9), we obtain the corresponding gradient

$$\nabla\xi_l(i) = \frac{\partial\xi_l(i)}{\partial\boldsymbol{h}_l} = -e_l(i)\boldsymbol{x}_l(i) \tag{10}$$

from which the instantaneous Hessian is obtained as

$$\mathbf{G}_l(i) = \frac{\partial^2\xi_l(i)}{\partial\boldsymbol{h}_l\partial\boldsymbol{h}_l} = \boldsymbol{x}_l(i)\boldsymbol{x}_l^{\mathrm{T}}(i). \tag{11}$$

In the diffusion framework, since the local estimates $\boldsymbol{\varphi}_l(k)$ made by neighboring nodes $l\in\mathcal{N}_k$ are available at node k, an improved estimate is first obtained via linear combination of the former estimates [6], that is

$$\boldsymbol{h}_k(i-1) = \sum_{l\in\mathcal{N}_k} a_{l,k}\boldsymbol{\varphi}_l(i-1). \tag{12}$$

Substituting (12) into (8), we have

$$\begin{aligned}\boldsymbol{h}_k(i) &= \sum_{l\in\mathcal{N}_k} a_{l,k}\boldsymbol{\varphi}_l(i-1) - \mu\sum_{l\in\mathcal{N}_k} a_{l,k}[\mathbf{G}_l(i)+\lambda\mathbf{I}]^{-1}\nabla\xi_l(i)\big|_{\boldsymbol{h}_k(i-1)}\\ &= \sum_{l\in\mathcal{N}_k} a_{l,k}\Big[\boldsymbol{\varphi}_l(i-1) - \mu[\mathbf{G}_l(i)+\lambda\mathbf{I}]^{-1}\nabla\xi_l(i)\big|_{\boldsymbol{h}_k(i-1)}\Big]\end{aligned} \tag{13}$$

where the gradient is evaluated at $\boldsymbol{h}_k(i-1)$. Based on (13), we can obtain the following expression for the local estimate $\boldsymbol{\varphi}_l(i)$ at time i,

$$\begin{aligned}\boldsymbol{\varphi}_l(i) &= \boldsymbol{\varphi}_l(i-1) - \mu[\mathbf{G}_l(i)+\lambda\mathbf{I}]^{-1}\nabla\xi_l(i)\big|_{\boldsymbol{h}_k(i-1)}\\ &\approx \boldsymbol{\varphi}_l(i-1) - \mu[\mathbf{G}_l(i)+\lambda\mathbf{I}]^{-1}\nabla\xi_l(i)\big|_{\boldsymbol{h}_l(i-1)}\\ &\approx \boldsymbol{h}_l(i-1) - \mu[\mathbf{G}_l(i)+\lambda\mathbf{I}]^{-1}\nabla\xi_l(i)\big|_{\boldsymbol{h}_l(i-1)}\end{aligned} \tag{14}$$

The approximation in the second line of (14), which is valid if the differences between the estimates $\boldsymbol{h}_l(i-1)$ at different nodes l are small, makes it possible to update $\boldsymbol{\varphi}_l(i)$ locally, i.e. only based on observations available at node l. The approximation in the third line of (14), where we replace $\boldsymbol{\varphi}_l(i-1)$ by $\boldsymbol{h}_l(i-1)$, is justified since $\boldsymbol{h}_l(i-1)$ includes more information than the corresponding intermediate estimate $\boldsymbol{\varphi}_l(i-1)$. Note that with this last substitution, initialization of $\boldsymbol{\varphi}_k(0)$ is no longer needed [25]. This substitution is not necessary for the derivation, but previous studies have shown that it can improve the performance of the diffusion algorithm [6]. Finally, by combining (7), (8), (12) and (14), we obtain the main weight update equations of the proposed PLD algorithm as follows,

$$\begin{cases}\boldsymbol{\varphi}_k(i) = \boldsymbol{h}_k(i-1) + \mu\left(\boldsymbol{x}_k(i)\boldsymbol{x}_k^{\mathrm{T}}(i)+\lambda\mathbf{I}\right)^{-1}\boldsymbol{x}_k(i)e_k(i) & \texttt{(adaptation)}\\ \boldsymbol{h}_k(i) = \sum_{l\in\mathcal{N}_k} a_{l,k}\boldsymbol{\varphi}_l(i) & \texttt{(combination)}\end{cases} \tag{15}$$

It should be noted that the proposed algorithm is conceptually different from the diffusion affine projection algorithm (DAPA) with projection order set to 1. Indeed, the term $\left(\boldsymbol{x}_k(i)\boldsymbol{x}_k^{\mathrm{T}}(i)+\lambda\mathbf{I}\right)^{-1}$ of the PLD algorithm represents a matrix operation, while the corresponding term of the DAPA becomes a scalar.

One of the main problems facing the LGD algorithm in the nonlinear system identification task is the selection of the regularization (or mixing) parameter λ. Indeed, we have been able to observed that the convergence performance of PLD is greatly dependent on this parameter. Furthermore, since different nodes have access to different measurements, it would seem preferable to adjust the parameter λ separately at each node in order to optimize performance. To tackle these issues, a SR strategy is developed in the next section.

4. Self-regularization strategy

When implementing the subband adaptive filter algorithm [26], normalized LMS algorithm [27] and affine projection algorithm [28], a small regularization parameter (leading to a large step size) is required at the onset of adaptation, in order to accelerate convergence during initialization. As the iteration cycles progress, the error becomes small in steady-state, and a larger regularization parameter is required for small misadjustment. Such variable parameter scheme can also be found in recursive least squares method, as well as Gauss-Newton strategy. In [29], the Gauss-Newton type variable forgetting factor-recursive least-squares (VFF-RLS) algorithm was proposed for improving the tracking capability. Following this method, several VFF strategies were developed for distributed in-network system identification [30,31]. In these approaches, a forgetting factor is used when the error is large to achieve a faster convergence rate, whereas the forgetting factor increases when the error becomes small so as to yield better steady-state performance. In the PLD algorithm, when the error signal is small, the adaptive weight vector estimate is near its optimal value and it is hence preferable to use a smaller value of λ to reduce the influence of gradient descent. On the contrary, if the error signal becomes large, it is preferable to increase this same parameter. There is little literature addressing the adaptation problem of the mixing parameter in LM type algorithms. Motivated by [28], we propose a new SR scheme for the PLD algorithm.

In contrast to the use of a constant scalar regularization parameter λ common to every node, as in the previous section, we here consider the use of a more general time-varying mixing vector $\boldsymbol{\lambda}_k(i) \in \mathbb{R}^{M\times 1}$ at node k, that is: we replace the diagonal matrix $\lambda\mathbf{I}$ by a more general diagonal matrix $\boldsymbol{\Upsilon}_k(i) = \text{diag}[\boldsymbol{\lambda}_k(i)]$. We propose to use a stochastic gradient descent approach to update the mixing vector, with the aim to minimize the local error signal power at each node over time. Specifically, we let

$$\boldsymbol{\lambda}_k(i) = \boldsymbol{\lambda}_k(i-1) - \rho\boldsymbol{\zeta}_l(i-1) \tag{16}$$

where $\boldsymbol{\lambda}_k(i) = [\lambda_{0,k}(i), \ldots, \lambda_{M-1,k}(i)]^{\mathrm{T}}$ denotes the parameter vector at node k and time i, ρ denotes the learning rate, and $\boldsymbol{\zeta}_l(i-1) = [\zeta_{0,k}(i-1), \ldots, \zeta_{M-1,k}(i-1)]^{\mathrm{T}}$ is the search direction. The latter is defined by

$$\begin{aligned}\zeta_{j,k}(i-1) &\triangleq \frac{1}{2}\frac{\partial e_k^2(i)}{\partial \lambda_{j,k}(i-1)} \\ &= \frac{1}{2}\frac{\partial e_k^2(i)}{\partial e_k(i)}\frac{\partial e_k(i)}{\partial \boldsymbol{h}_k(i-1)}\frac{\partial \boldsymbol{h}_k(i-1)}{\partial \lambda_{j,k}(i-1)}\end{aligned} \tag{17}$$

The right-hand side of (17) can be calculated by using

$$\frac{\partial e_k^2(i)}{\partial e_k(i)} = 2e_k(i), \quad \frac{\partial e_k(i)}{\partial \boldsymbol{h}_k(i-1)} = -\boldsymbol{x}_k^{\mathrm{T}}(i) \tag{18}$$

$$\begin{aligned}\frac{\partial \boldsymbol{h}_k(i-1)}{\partial \lambda_{j,k}(i-1)} &= -\mu\sum_{l\in\mathcal{N}_k} a_{l,k}\left[\boldsymbol{x}_l(i-1)\boldsymbol{x}_l^{\mathrm{T}}(i-1) + \boldsymbol{\Upsilon}_l(i-1)\right]^{-1} \\ &\cdot \frac{\partial \boldsymbol{\Upsilon}_l(i-1)}{\partial \lambda_{j,l}(i-1)}\left[\boldsymbol{x}_l(i-1)\boldsymbol{x}_l^{\mathrm{T}}(i-1) + \boldsymbol{\Upsilon}_l(i-1)\right]^{-1} \\ &\times \boldsymbol{x}_l(i-1)e_l(i-1)\end{aligned} \tag{19}$$

Additional details about the derivation of (19) are given in Appendix A.

In the sequel, to simplify the developments considering space limitations, we let $\lambda_{0,k}(i) = \ldots = \lambda_{M-1,k}(i) \equiv \lambda_k(i)$, so that $\boldsymbol{\lambda}_k(i) = \lambda_k(i)[1, \ldots, 1]^{\mathrm{T}})$, but generalizations are possible. Combining (16), (18) and (19), and employing again the linear combination strategy, we obtain the following procedure for updating the mixing vectors $\boldsymbol{\lambda}_k(i)$ within the diffusion framework:

$$\begin{aligned}\chi_k'(i) &= \lambda_k(i-1) - \rho\mu e_k(i)\boldsymbol{x}_k^{\mathrm{T}}(i) \\ &\cdot \left[\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1) + \lambda_k(i-1)\mathbf{I}\right]^{-2}\boldsymbol{x}_k(i-1)e_k(i-1)\end{aligned} \tag{20a}$$

$$\chi_k(i) = \begin{cases}\lambda_{\max} & \text{if } \chi_k'(i) \geq \lambda_{\max} \\ \lambda_{\min} & \text{if } \chi_k'(i) \leq \lambda_{\min} \\ \chi_k'(i) & \text{otherwise}\end{cases} \tag{20b}$$

$$\lambda_k(i) = \sum_{l\in\mathcal{N}_k} a_{l,k}\chi_l(i). \tag{20c}$$

where $\chi_k(i)$ is the local intermediate estimate of $\lambda_k(i)$, $\lambda_{\min}$ is chosen to impose a minimum value for the mixing parameters $\lambda_k(i)$, and $\lambda_{\max}$ is chosen to ensure that these parameters remain bounded. By using (20a)–(20c) along with the PLD adaptation in (15), we obtain the update equation of the proposed SR-PLD algorithm,

$$\begin{cases}\boldsymbol{\varphi}_k(i) = \boldsymbol{h}_k(i-1) + \mu\left[\boldsymbol{x}_k(i)\boldsymbol{x}_k^{\mathrm{T}}(i) + \lambda_k(i)\mathbf{I}\right]^{-1}\boldsymbol{x}_k(i)e_k(i) & \texttt{(adaptation)} \\ \boldsymbol{h}_k(i) = \sum\limits_{l\in\mathcal{N}_k} a_{l,k}\boldsymbol{\varphi}_l(i) & \texttt{(combination)}.\end{cases} \tag{21}$$

5. Comparison of memory requirement

In this section, we compare the memory requirements of the proposed power diffusion and Volterra diffusion algorithms. According to [14], the number of model coefficients of the Volterra diffusion algorithm for each node is $P_1 = \frac{(L+2)!}{L!2!} - 1$. In contrast, the power diffusion algorithm requires $P_2 = M$ coefficients for each node (see (6)). Fig. 1 shows the number of coefficients of the Volterra diffusion algorithm [14] and the proposed power diffusion algorithm, where $N_k = |\mathcal{N}_k|$ denotes the cardinal of neighborhood set $\mathcal{N}_k$. One can see that the proposed algorithm has reduced memory requirement as compared to the diffusion Volterra algorithm.

6. Simulation results

To illustrate the merits of the proposed diffusion-based algorithms, simulations are carried out in the context of distributed

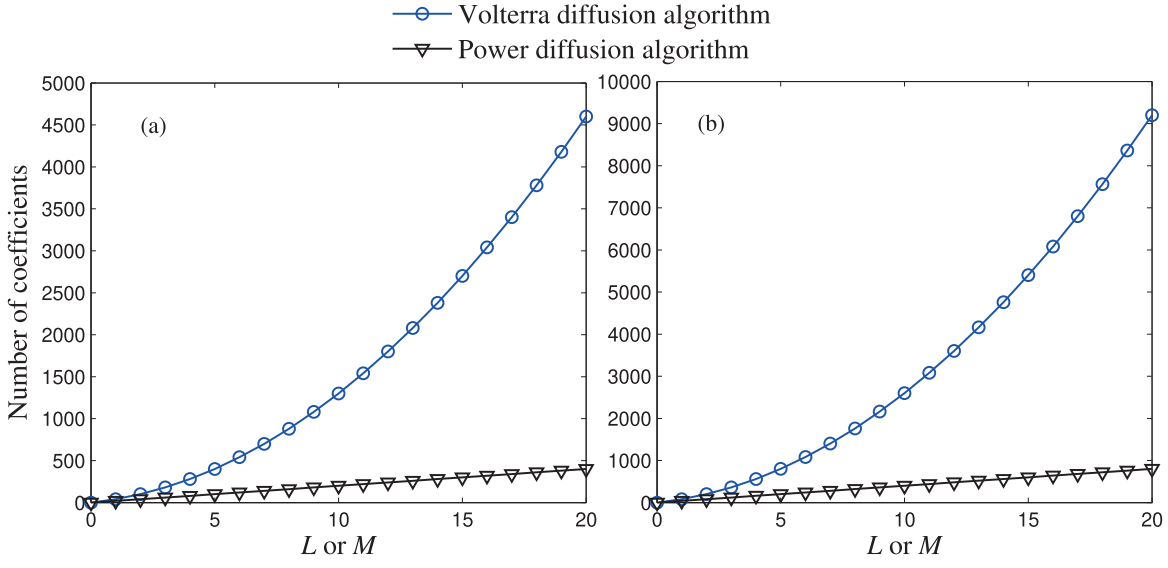

Fig. 1. Number of coefficients of the Volterra diffusion algorithm and power diffusion algorithm (for the Volterra diffusion algorithm, x-axis is L; for the power diffusion algorithm, x-axis is M.) (a) $N_k = 20$, (b) $N_k = 40$.

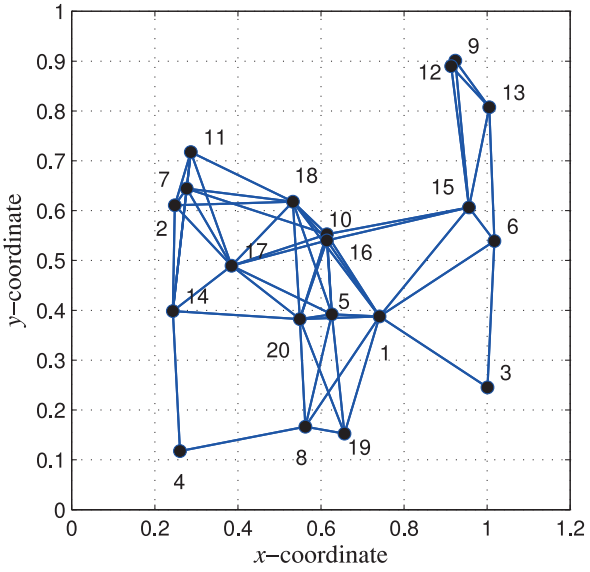

Fig. 2. Network topology.

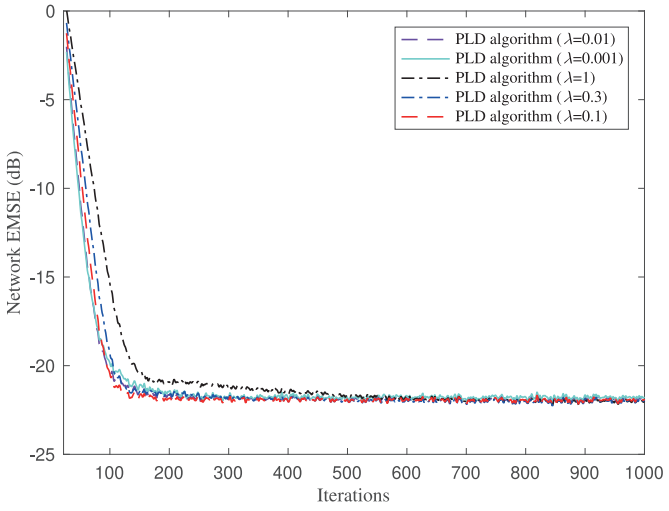

Fig. 4. Network EMSE of the PLD algorithm versus iteration number for different choices of λ.

in-network nonlinear system identification. We consider a network with $N = 20$ nodes in the plane, as shown in Fig. 2. The combination coefficients $\{a_{k,l}\}$ are set according to the uniform rule [6]. The network excess mean square error (EMSE), defined as Network EMSE $= \frac{1}{N}\sum_{k=1}^{N}(z_k(i) - y_k(i))^2$, is used as a performance measure, where $z_k(i)$ is the true unknown system output. The results are averaged over 200 independent simulations.

Example 1. In order to evaluate performance of the proposed algorithms in a nonlinear system with memory, the following model is used

$$z_k(i) = x_k(i) + 0.1x_k(i)x_k(i-1) + 0.45x_k^3(i) + 0.2x_k(i)x_k^2(i-1) + 0.4x_k^5(i). \quad (22)$$

The input signal $x_k(i)$ is obtained by truncating independent samples from a zero-mean Gaussian distribution with unit variance to the range $[-1, +1]$ [21]. The noise signal $v_k(i)$ is obtained from independent samples of a zero-mean Gaussian distribution. The distribution of the signal-to-noise ratio (SNR) at the different nodes is shown in Fig. 3. For the Volterra diffusion algorithm, we set $L = 2$ (corresponding to 5 adaptive coefficients), and for the other algorithms, we set $M = 6$. For the SR-PLD, we set $\lambda_{\min} = 0.01$, $\lambda_{\max} = 10$ and $\rho = 0.2$. Fig. 4 shows the network EMSE learning curves with different λ. As can be seen from this figure, the different choices of λ lead to a similar performance in convergence, although the best performance is obtained with $\lambda = 0.1$. To further demonstrate its performance, we compare the proposed algorithms with other existing algorithms. To obtain nonlinear modeling capability of the algorithm, the diffusion recursive least-squares (DRLS) algorithm is combined with integer power of the input regressor (6). The forgetting factor is set to $\theta = 0.999$, $\boldsymbol{P}_k(0) = \delta\mathbf{I}$, and $\delta = 15$ for the DRLS algorithm and its nonlinear variant. As can be seen from Fig. 5, the DRLS with integer power of the input regressor algorithm achieves smaller steady-state error than the linear DRLS algorithm. Moreover, the DRLS with integer power of the input regressor has similar performance as the power diffusion algorithm. It requires $4M^2 + 3M + N_kM$ multiplications and $3M^2 + N_kM$ additions. In contrast, the proposed PLD algorithm requires $2M^2 + 4M + N_kM$ multiplications and $M^2 + 2M + N_kM$ additions. Note that the term N_kM comes from the *combination step* and it depends on network size of the diffusion algorithm. These considerations strongly motivate the use of the proposed PLD algorithm for in-network distributed nonlinear system identification. The network EMSE presented in Fig. 5 also show that the proposed algorithms achieve better results than the Volterra diffusion algorithm. Specifically, among the PLD and SR-PLD algorithms exhibit

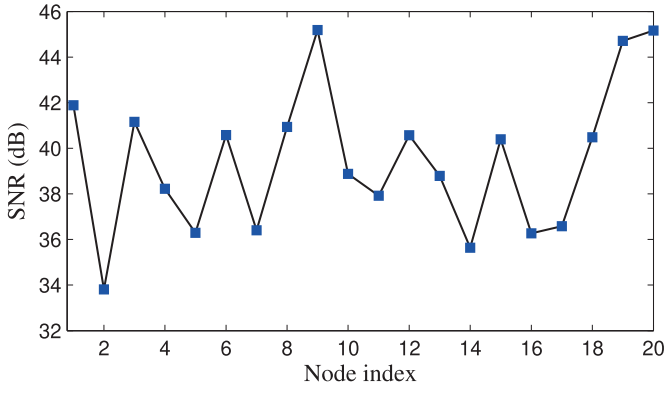

Fig. 3. Distribution of SNRs used in Example 1.

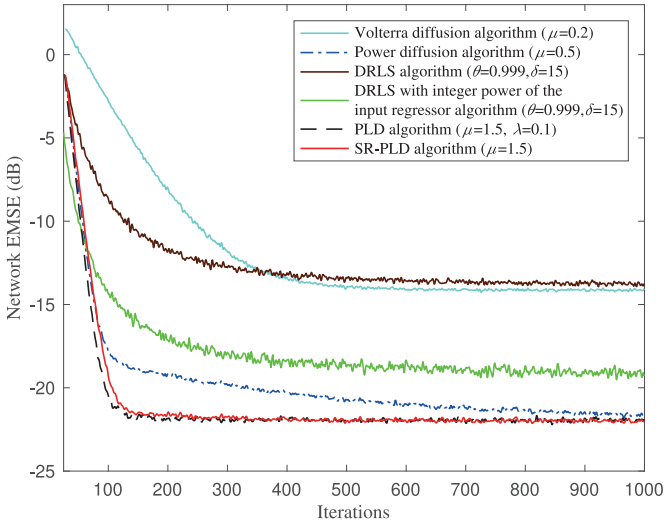

Fig. 5. Network EMSE of the proposed and existing algorithms.

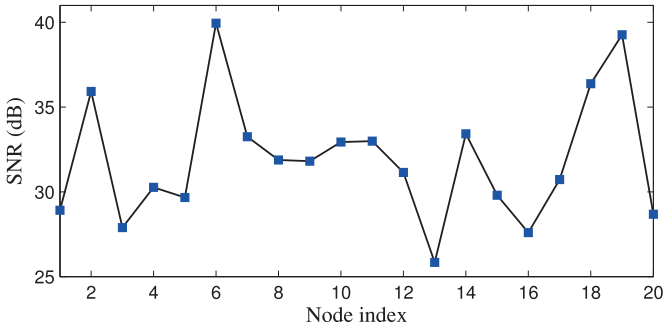

Fig. 6. Distribution of SNRs used in Example 2.

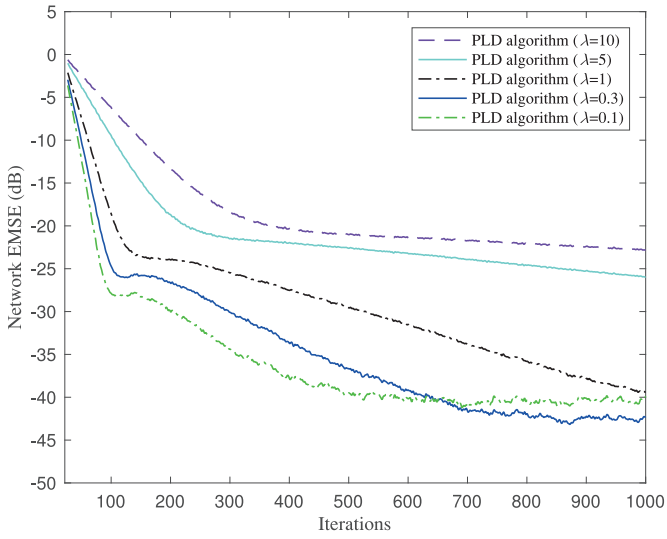

Fig. 7. Network EMSE of the PLD algorithm versus iteration number for different choices of λ.

the fastest convergence and lowest network EMSE, outperforming Volterra diffusion algorithm by more than 10dB in steady-state.

Example 2. The system output at node k is generated according to Dogariu et al. [21]

$$z_k(i) = x_k(i) + 0.3x_k^3(i) + 0.5x_k^5(i). \tag{23}$$

The input signal $x_k(i)$ and the noise signal $v_k(i)$ are obtained as in the previous example. The SNR distribution over the network is illustrated in Fig. 6. The memory size and filter length are the same as in the previous example. First, we investigate the performance of the PLD algorithm with $\mu = 1.5$ using different values of the mixing parameter λ. As can be seen in Fig. 7, the performance of PLD largely depends on the value of λ, with $\lambda = 0.3$ providing the best choice in terms of convergence speed and misadjustment. Next, we compare the performance of the power diffusion, PLD and SR-PLD algorithms to that of the Volterra diffusion algorithm. In the SR-PLD algorithm, we set $\lambda_{\min} = 0.01$, $\lambda_{\max} = 10$ and $\rho = 0.2$. As can be seen from the results in Fig. 8, when compared with a Volterra diffusion algorithm with similar complexity, the proposed algorithms can significantly improve the convergence rate and the misadjustment. Among these, the SR-PLD achieves the best overall performance in convergence, while further enabling automatic adjustment of the mixing parameters $\lambda_k(i)$.

Example 3. Next, we repeat the same simulation experiments as Example 2, but this time using input waveforms from a power system application. Such input waveforms can be expressed as (see Fig. 9) [18]

$$x_k(i) = \sin(\omega_k i + 29.3^\circ) + 0.5\sin(3\omega_k i + 81.6^\circ) + 0.1\sin(5\omega_k i - 66.2^\circ) \tag{24}$$

where ω_k denotes the fundamental frequency at node k. The distribution of SNR and ω_k values employed here are given in Fig. 10. The performance of the PLD algorithm with $\mu = 1$ for different values of λ is shown in Fig. 11, where the best choice is 0.1. Fig. 12 illustrates the comparative performance of the proposed power diffusion and Volterra diffusion algorithms, where for the SR-PLD, we set $\lambda_{\min} = 0.01$, $\lambda_{\max} = 5$ and $\rho = 0.1$. Compared to the Volterra diffusion algorithm, the power diffusion and PLD algorithms yield reduced misadjustment. In addition, the SR-PLD algorithm enhances the performance of the PLD algorithm while offering a self-regularization mechanism that allows automatic adjustment of the mixing parameters at the different nodes. To further demonstrate the adaptation of the mixing parameters in the SR-PLD algorithm, Fig. 13 shows the time evolution of $\lambda_k(i)$ as computed on the basis of adaptive update (20). As can be seen, the variations in the parameters $\lambda_k(i)$ are consistent with the changes in the size of the identification errors, in the sense that a reduction

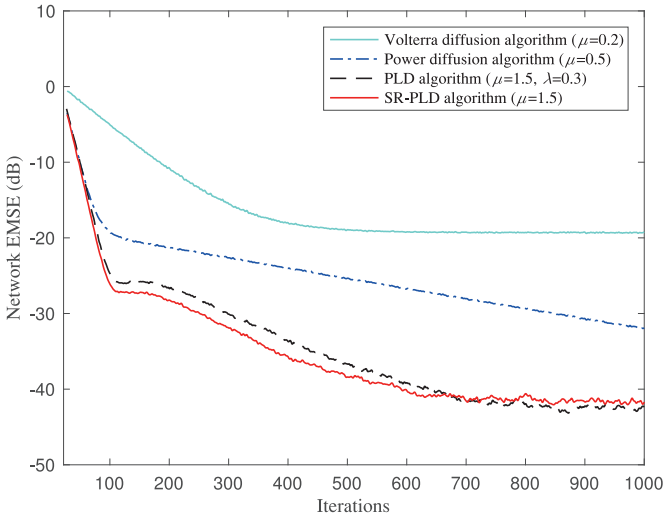

Fig. 8. Network EMSE of the proposed and existing algorithms.

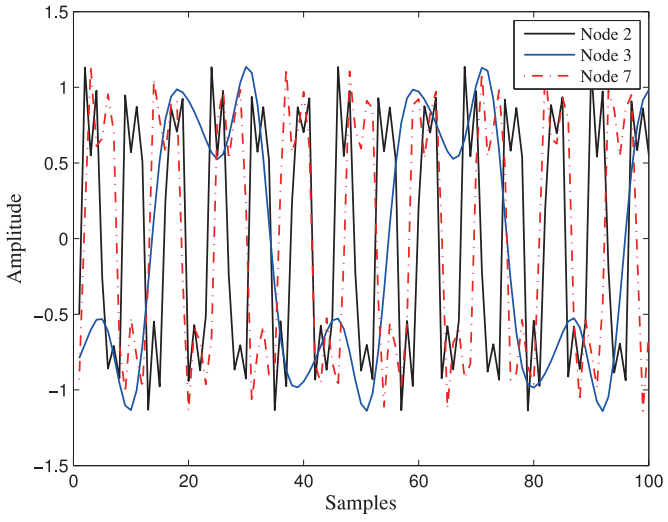

Fig. 9. Input signals used in Example 3.

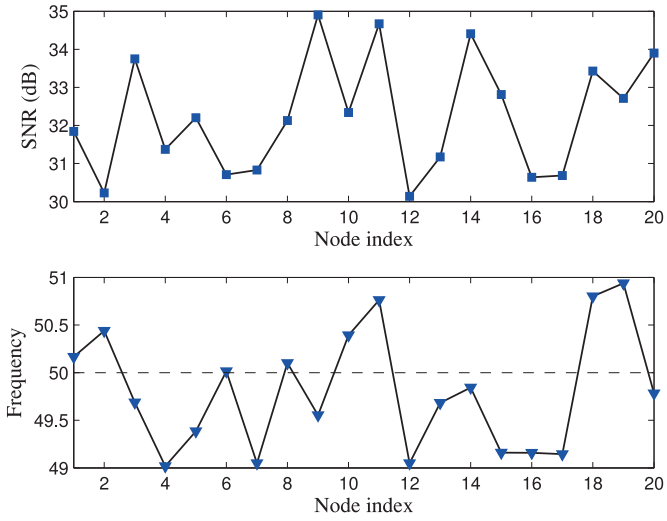

Fig. 10. Distribution of SNRs and angular frequencies used in Example 3.

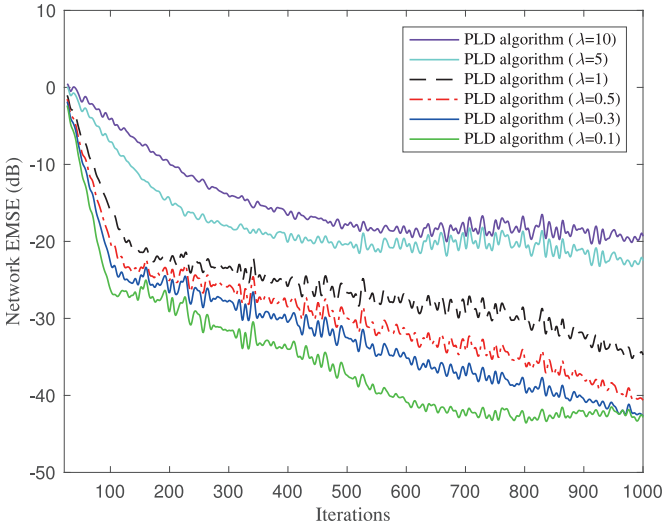

Fig. 11. Network EMSE of the power Levenberg diffusion algorithm versus different λ.

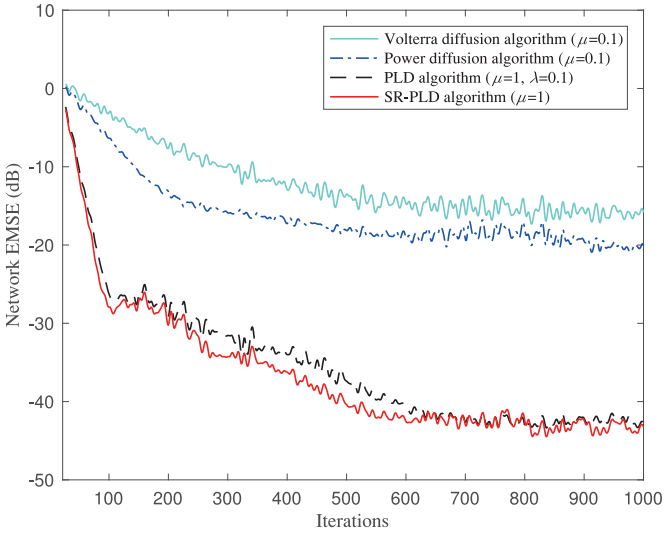

Fig. 12. Network EMSE of the proposed and existing algorithms.

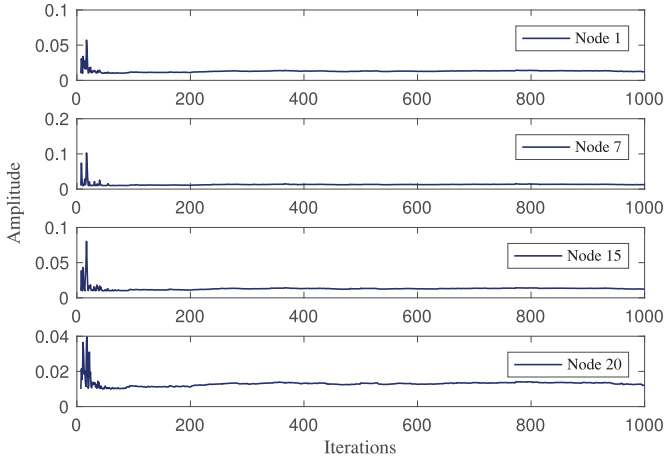

Fig. 13. Time evolution of mixing parameter $\lambda_k(i)$ in SR-PLD algorithm at selected nodes.

in the error $e_k(i)$ time leads to a reduction in the corresponding parameter value $\lambda_k(i)$.

7. Conclusion

Novel diffusion algorithms for distributed in-network nonlinear system identification have been proposed. The power diffusion algorithm can be regarded as a special form of the Volterra diffusion algorithm where only instantaneous powers of the input are employed. It was shown that in the considered application, the power diffusion algorithm enjoys improved performance in comparison to the Volterra diffusion LMS algorithm, with similar or reduced computational complexity. To further improve the convergence rate and steady-state performance of the power diffusion algorithm, we developed the PLD algorithm based on the LGD method. Considering that the selection of its regularization parameter may pose difficulties in practice, we further proposed the SR scheme for the PLD algorithm. Simulation results in the context of distributed in-network nonlinear system identification have shown that these new nonlinear diffusion algorithms can significantly outperform existing algorithms.

Conflicts of Interest

The authors declare no conflict of interest.

Appendix A. Derivation of (19)

Making use of the adaptation rules in (15), we can write,

$$\frac{\partial \boldsymbol{h}_k(i-1)}{\partial \lambda_{j,k}(i-1)} = \mu \sum_{l\in\mathcal{N}_k} a_{l,k}\left[\frac{\partial[\boldsymbol{x}_l(i-1)\boldsymbol{x}_l^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_l(i-1)]^{-1}}{\partial \lambda_{j,l}(i-1)}\right] \times \boldsymbol{x}_l(i-1)e_l(i-1). \tag{A.1}$$

We then define

$$\boldsymbol{\Gamma} \triangleq \boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1). \tag{A.2}$$

Differentiating $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^{-1}=\mathbf{I}$ with respect to $\lambda_{k,j}(i)$ yields

$$\frac{\partial \boldsymbol{\Gamma}}{\partial \lambda_{j,k}(i-1)}\boldsymbol{\Gamma}^{-1}+\boldsymbol{\Gamma}\frac{\partial \boldsymbol{\Gamma}^{-1}}{\partial \lambda_{j,k}(i-1)}=0 \tag{A.3}$$

from which we obtain

$$\frac{\partial \boldsymbol{\Gamma}^{-1}}{\partial \lambda_{j,k}(i-1)} = -\boldsymbol{\Gamma}^{-1}\frac{\partial \boldsymbol{\Gamma}}{\partial \lambda_{j,k}(i-1)}\boldsymbol{\Gamma}^{-1}. \tag{A.4}$$

Substituting $\boldsymbol{\Gamma}=\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1)$ into (A.4), we have

$$\begin{aligned}
&\frac{\partial\left[\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1)\right]^{-1}}{\partial\lambda_{j,k}(i-1)}\\
&=-\left[\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1)\right]^{-1}\\
&\quad\times\frac{\partial\left[\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1)\right]}{\partial\lambda_{j,k}(i-1)}\\
&\quad\times\left[\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1)\right]^{-1}\\
&=-\left[\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1)\right]^{-1}\\
&\quad\times\frac{\partial\boldsymbol{\Upsilon}_k(i-1)}{\partial\lambda_{j,k}(i-1)}\left[\boldsymbol{x}_k(i-1)\boldsymbol{x}_k^{\mathrm{T}}(i-1)+\boldsymbol{\Upsilon}_k(i-1)\right]^{-1}. \qquad \text{(A.5)}
\end{aligned}$$

Using (A.1) and (A.5), (19) is finally obtained.

References

[1] A.H. Sayed, Adaptation, learning, and optimization over networks, Found. Trends Mach. Learn. 7 (4–5) (2014) 311–801.
[2] A.H. Sayed, Diffusion adaptation over networks, vol. 3, Academic Press Library in Signal Processing, 2013.
[3] P. Shen, C. Li, Z. Zhang, Distributed active learning, IEEE Access 4 (2016) 2572–2579.
[4] S. Kanna, D.H. Dini, Y. Xia, S. Hui, D.P. Mandic, Distributed widely linear Kalman filtering for frequency estimation in power networks, IEEE Trans. Signal Inf. Process. Netw. 1 (1) (2015) 45–57.
[5] J.A. Bazerque, G.B. Giannakis, Distributed spectrum sensing for cognitive radio networks by exploiting sparsity, IEEE Trans. Signal Process. 58 (3) (2010) 1847–1862.
[6] F.S. Cattivelli, A.H. Sayed, Diffusion LMS strategies for distributed estimation, IEEE Trans. Signal Process. 58 (3) (2010) 1035–1048.
[7] R. Arablouei, S. Werner, Y.-F. Huang, K. Doğançay, Distributed least mean-square estimation with partial diffusion, IEEE Trans. Signal Process. 62 (2) (2014) 472–484.
[8] F. Chen, X. Shao, Broken-motifs diffusion LMS algorithm for reducing communication load, Signal Process. 133 (2017) 213–218.
[9] F.S. Cattivelli, C.G. Lopes, A.H. Sayed, Diffusion recursive least-squares for distributed estimation over adaptive networks, IEEE Trans. Signal Process. 56 (5) (2008) 1865–1877.
[10] Y. Chu, C. Mak, A variable forgetting factor diffusion recursive least squares algorithm for distributed estimation, Signal Process. 140 (2017) 219–225.
[11] R. Mitra, V. Bhatia, Diffusion-KLMS algorithm and its performance analysis for non-linear distributed networks, arXiv:1509.01352 (2015).
[12] S.P. Talebi, S. Werner, D.P. Mandic, Distributed adaptive filtering of α-stable signals, IEEE Signal Process. Lett. 25 (10) (2018) 1450–1454.
[13] S. Chouvardas, M. Draief, A diffusion kernel LMS algorithm for nonlinear adaptive networks, in: Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2016, pp. 4164–4168.
[14] L. Lu, H. Zhao, B. Champagne, Distributed nonlinear system identification in α-stable noise, IEEE Signal Process. Lett. 25 (7) (2018) 979–983.
[15] K. Levenberg, A method for the solution of certain non-linear problems in least squares, Q. Appl. Math. 2 (2) (1944) 164–168.
[16] F.M. Ham, I. Kostanic, Principles of Neurocomputing for Science and Engineering, McGraw-Hill Higher Education, 2000.
[17] J.C. Principe, Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives, Springer Science & Business Media, 2010.
[18] A. Sarkar, S.R. Choudhury, S. Sengupta, A self-synchronized ADALINE network for on-line tracking of power system harmonics, Measurement 44 (4) (2011) 784–790.
[19] J. Shawash, D.R. Selviah, Real-time nonlinear parameter estimation using the Levenberg–Marquardt algorithm on field programmable gate arrays, IEEE Trans. Ind. Electron. 60 (1) (2013) 170–176.
[20] J. Qiao, L. Wang, C. Yang, K. Gu, Adaptive Levenberg–Marquardt algorithm based echo state network for chaotic time series prediction, IEEE Access 6 (2018) 10720–10732.
[21] L.M. Dogariu, S. Ciochină, C. Paleologu, J. Benesty, P. Piantanida, An adaptive solution for nonlinear system identification, in: IEEE Int. Symposium on Signals, Circuits and Systems, 2017, pp. 1–4.
[22] S.P. Talebi, S. Kanna, D.P. Mandic, A non-linear state space frequency estimator for three-phase power systems, in: Proc. Int. Joint Conf. Neural Networks (IJCNN), 2015, pp. 1–7.
[23] A. Stenger, W. Kellermann, Adaptation of a memoryless preprocessor for nonlinear acoustic echo cancelling, Signal Process. 80 (9) (2000) 1747–1760.
[24] A.R. Heravi, G.A. Hodtani, Comparison of the convergence rates of the new correntropy-based Levenberg–Marquardt (CLM) method and the fixed-point maximum correntropy (FP-MCC) algorithm, Circuits, Syst., Signal Process. 37 (7) (2018). 2994–2910
[25] S. Huang, C. Li, Distributed sparse total least-squares over networks, IEEE Trans. Signal Process. 63 (11) (2015) 2986–2998.
[26] J. Ni, F. Li, A variable regularization matrix normalized subband adaptive filter, IEEE Signal Process. Lett. 16 (2) (2009) 105–108.
[27] J. Benesty, C. Paleologu, S. Ciochină, On regularization in adaptive filtering, IEEE Trans. Audio Speech Lang. Process. 19 (6) (2011) 1734–1742.
[28] Y.-S. Choi, H.-C. Shin, W.-J. Song, Adaptive regularization matrix for affine projection algorithm, IEEE Trans. Circuits Syst. II 54 (12) (2007) 1087–1091.
[29] S. Song, J.S. Lim, S.J. Baek, K.M. Sung, Gauss Newton variable forgetting factor recursive least squares for time varying parameter tracking, Electron. Lett. 36 (11) (2000) 988–990.
[30] L. Zhang, Y. Cai, C. Li, R.C.d. Lamare, Variable forgetting factor mechanisms for diffusion recursive least squares algorithm in sensor networks, EURASIP J. Adv. Signal Process. 2017 (1) (2017) Art.no.57.
[31] Y. Chu, C. Mak, A variable forgetting factor diffusion recursive least squares algorithm for distributed estimation, Signal Process. 140 (2017) 219–225.