# Partial-Diffusion Recursive Least-Squares Estimation Over Adaptive Networks

Reza Arablouei [#1], Stefan Werner [*], Kutluyıl Doğançay [#]

[#] *School of Engineering, University of South Australia*
*Mawson Lakes SA, Australia*
[1] reza.arablouei@unisa.edu.au

[*] *Department of Signal Processing and Acoustics, School of Electrical Engineering, Aalto University*
*Espoo, Finland*

***Abstract*—In the diffusion strategies for distributed estimation over adaptive networks, each node calculates a weighted average of the intermediate parameter estimates of its neighboring nodes. Thus, all the nodes should continuously share their intermediate estimates with their neighbors. In this paper, we consider exchanging a predetermined number of elements of each intermediate estimate vector at each iteration rather than the entire vectors. We examine two different schemes, i.e., stochastic and sequential partial-diffusion schemes, for selecting the to-be-diffused elements at each iteration. Accordingly, we propose a partial-diffusion recursive least-squares (PDRLS) algorithm that can alleviate internode communications at the expense of estimation performance. Simulation results show that the communication-performance trade-off offered by the proposed algorithm is indeed lucrative.**

***Keywords*—*adaptive networks; diffusion adaptation; distributed estimation; partial diffusion; recursive least-squares***

## I. INTRODUCTION

Various self-organized systems and distributed learning mechanisms can be effectively modeled using adaptive networks. An adaptive network is composed of a set of spatially-scattered nodes that are able to process data and learn. The nodes are interconnected and can cooperatively perform decentralized real-time data processing and optimization through exchanging information. The cooperation spreads the information throughout the network. As a result, the nodes can adapt to possible statistical changes in the data or topographical alterations in the network [1], [2].

Several strategies for distributed estimation over adaptive networks that rely on local interactions and in-network processing have been proposed. They include the incremental [3]-[5], consensus [6]-[8] and diffusion [9]-[11] strategies. In the incremental strategies, nodes only communicate with their neighbors, which are within a predefined cyclic path. The path should visit all the nodes in the network. Defining such a path, in particular for large networks, is not generally straightforward. The consensus strategies are constrained to converge to the same optimizer/estimate at each node. They are typically realized in two time steps. In diffusion strategies, nodes diffuse their data into the network by sharing them with their neighbors. The diffusion strategies are robust to link/node failures and have good adaptability and tracking abilities as well as flexibility for ad hoc deployment. It has been shown that the diffusion strategies outperform the consensus strategies in distributed estimation over adaptive networks [12].

The diffusion recursive least-squares (diffRLS) algorithm [9] is a decentralized version of the conventional recursive least-squares (RLS) algorithm [13] that implements diffusion-based distributed estimation over adaptive networks. It adaptively seeks the distributed least-squares (LS) solution of the global estimation problem across the network and approaches the optimal LS solution without transmitting or inverting any matrix.

In wireless ad hoc networks, electrical power resources of the nodes are often restricted. When the nodes perform a collaborative task over the network, the most power-demanding action is usually the data transmission among the communicating nodes. The communications also take up bandwidth, which is a scarce commodity. Therefore, it is desirable to reduce the internode communications without compromising the benefits of cooperation significantly. Some attempts to achieve this goal by way of partial updating [14] or set-membership filtering [15] have been reported in [16]-[20].

In this paper, we propose a partial-diffusion recursive least-squares (PDRLS) algorithm in which a predefined number of elements of each node's intermediate estimate vector are diffused at each iteration. We devise two different schemes for choosing the to-be-diffused elements of each node at each iteration. Through computer simulations, we show that the proposed algorithm can mitigate internode communications at the cost of graceful performance degradation.

## II. ALGORITHM DESCRIPTION

### *A. Diffusion recursive least-squares algorithm*

Let us consider a connected network with $N$ nodes that aim to identify an unknown parameter vector, $\in \mathbb{R}^{L\times 1}$ , in a collective manner. Each node observes an input vector, $\mathbf{x}_{k,n} \in \mathbb{R}^{L\times 1}$, and an output signal, $d_{k,n} \in \mathbb{R}$, that arise from a linear system described by

$$d_{k,n} = \mathbf{x}_{k,n}^T \mathbf{h} + \nu_{k,n}.$$

Here, $k \in \{1,2,\dots,N\}$ is the node index and $n \in \mathbb{N}$ is the time index. Superscript $T$ denotes matrix/vector transposition and $\nu_{k,n} \in \mathbb{R}$ represents the background noise.

An estimate of $\mathbf{h}$ at node $k$ and time instant $n$ can be found adaptively in two phases that we explain below.

1) The *adaptation* where the node computes an *intermediate* estimate, $\mathbf{z}_{k,n} \in \mathbb{R}^{L\times 1}$ , by minimizing its local exponentially-weighted least-squares cost function and

exploiting the input-output data available up to the present time:

$$\mathbf{z}_{k,n} = \arg\min_{\mathbf{z}} \left\| \mathbf{d}_{k,n} - \mathbf{X}_{k,n}^T \mathbf{z} \right\|^2 = \left(\mathbf{X}_{k,n}\mathbf{X}_{k,n}^T\right)^{-1}\mathbf{X}_{k,n}\mathbf{d}_{k,n}$$

where

$$\mathbf{d}_{k,n} = \left[d_{k,n}, \lambda^{1/2} d_{k,n-1}, \dots, \lambda^{(n-1)/2} d_{k,1}\right]^T,$$
$$\mathbf{X}_{k,n} = \left[\mathbf{x}_{k,n}, \lambda^{1/2}\mathbf{x}_{k,n-1}, \dots, \lambda^{(n-1)/2}\mathbf{x}_{k,1}\right],$$

$\|\cdot\|$ denotes the Euclidean norm, and $0 \ll \lambda < 1$ is a forgetting factor.

2) The *combination* where the node transmits its intermediate estimate vector to its neighbors. Then, it creates a new estimate by averaging the intermediate estimates available within its neighborhood:

$$\mathbf{w}_{k,n} = \sum_{l \in \mathcal{N}_k} c_{l,k} \mathbf{z}_{l,n} \qquad (1)$$

where $\mathcal{N}_k$ denotes the closed neighborhood of node $k$ and the coefficients $\{c_{l,k}\}$ weigh the intermediate estimates of the neighbors based on their reliability or significance. They satisfy

$$c_{l,k} = 0 \text{ if } l \notin \mathcal{N}_k \ \forall k \text{ and } \textstyle\sum_{l \in \mathcal{N}_k} c_{l,k} = 1 \ \forall k.$$

Defining

$$\mathbf{P}_{k,n} = \left(\mathbf{X}_{k,n}\mathbf{X}_{k,n}^T\right)^{-1}$$

and using the recursive properties of

$$\mathbf{X}_{k,n}\mathbf{X}_{k,n}^T = \lambda \mathbf{X}_{k,n-1}\mathbf{X}_{k,n-1}^T + \mathbf{x}_{k,n}\mathbf{x}_{k,n}^T \qquad (2)$$

and

$$\mathbf{X}_{k,n}\mathbf{d}_{k,n} = \lambda \mathbf{X}_{k,n-1}\mathbf{d}_{k,n-1} + \mathbf{x}_{k,n} d_{k,n}$$

together with applying the matrix inversion lemma [21] to (2), the following recursions for calculating $\mathbf{z}_{k,n}$ are obtained:

$$\mathbf{P}_{k,n} = \lambda^{-1}\left(\mathbf{P}_{k,n-1} - \frac{\lambda^{-1}\mathbf{P}_{k,n-1}\mathbf{x}_{k,n}\mathbf{x}_{k,n}^T\mathbf{P}_{k,n-1}}{1 + \lambda^{-1}\mathbf{x}_{k,n}^T\mathbf{P}_{k,n-1}\mathbf{x}_{k,n}}\right) \qquad (3)$$

$$\mathbf{z}_{k,n} = \mathbf{z}_{k,n-1} + \mathbf{P}_{k,n}\mathbf{x}_{k,n}\left(d_{k,n} - \mathbf{x}_{k,n}^T\mathbf{z}_{k,n-1}\right). \qquad (4)$$

Since $\mathbf{w}_{k,n-1}$ is a better estimate compared with $\mathbf{z}_{k,n-1}$, it is beneficial to replace $\mathbf{z}_{k,n-1}$ with $\mathbf{w}_{k,n-1}$ in (4):

$$\mathbf{z}_{k,n} = \mathbf{w}_{k,n-1} + \mathbf{P}_{k,n}\mathbf{x}_{k,n}\left(d_{k,n} - \mathbf{x}_{k,n}^T\mathbf{w}_{k,n-1}\right). \qquad (5)$$

Hence, the local estimates are diffused outside of each node's own neighborhood. We will refer to this algorithm, i.e., (3), (5) and (1), as the diffusion recursive least-squares (diffRLS) algorithm.

It is noteworthy that, in the above algorithm, the intermediate estimates are updated using only the local (node-specific) input-output data. However, in the algorithm of [9], each node shares its input-output data with its neighbors and uses the received data to update its intermediate estimate. This is carried out via a convex combination of the update terms induced by each input-output data pair. Here, in order to minimize the communication complexity, we only consider the abovementioned diffRLS algorithm and build our partial-diffusion algorithm upon it.

### B. Partial-diffusion recursive least-squares algorithm

By diffusing $M$ out of $L$ elements (entries) of the intermediate estimate vector of each node at each iteration, we can reduce the amount of communications that take place between the nodes and establish a trade-off between communication cost and estimation performance. The selection of the to-be-diffused elements at node $k$ and time instant $n$ can be represented as multiplication of the intermediate estimate vector, $\mathbf{z}_{k,n}$, by a square selection matrix, $\mathbf{A}_{k,n} \in \mathbb{R}^{L\times L}$, that has $M$ ones on its diagonal and zeros elsewhere. The positions of the ones determine the selected elements. Multiplication of $\mathbf{z}_{k,n}$ by $\mathbf{A}_{k,n}$ replaces its non-selected elements with zeros.

The combination phase at each node needs all the elements of the intermediate estimate vectors of the node's neighbors. However, when the intermediate estimates are partially diffused ($0 < M < L$), nodes have no access to the non-diffused elements. To resolve this ambiguity, nodes can use the elements of their own intermediate estimate vectors in lieu of the ones of their neighbors that have not been transmitted. Accordingly, a partial-diffusion recursive least-squares (PDRLS) algorithm can be formulated by using (3) and (5) for adaptation and the following equation for combination:

$$\mathbf{w}_{k,n} = \sum_{l \in \mathcal{N}_k} c_{l,k}\left[\mathbf{A}_{l,n}\mathbf{z}_{l,n} + \left(\mathbf{I}_L - \mathbf{A}_{l,n}\right)\mathbf{z}_{k,n}\right] \qquad (6)$$

where $\mathbf{I}_L \in \mathbb{R}^{L\times L}$ is the identity matrix. Note that (1) and (6) have the same computational complexity. Consequently, the PDRLS algorithm, i.e., (3), (5), and (6), requires the same number of arithmetic operations as the diffRLS algorithm.

Another implication of partial diffusion is that the nodes need to know which elements of their neighbors' intermediate estimate vectors are transmitted at each iteration. Therefore, address (position in the vector) of the diffused elements should also be transmitted. In the next subsection, we will describe two partial-diffusion schemes that bias the need for addressing.

In order to quantify the savings offered by partial diffusion, we define the *communication saving* as

$$q = 1 - \frac{MB + \log_2 \binom{L}{M}}{LB}$$

where $B$ is the number of bits that represent each element of the intermediate estimate vectors and the second term of the numerator on the right-hand side stands for the *addressing overhead*. This quantity denotes the fraction of communications saved by the PDRLS algorithm at each iteration with respect to the diffRLS algorithm. In order to quantify the foreseeable performance degradation caused by partial diffusion, we also define the *performance loss* as

$$l = \frac{\eta_M - \eta_L}{\eta_0 - \eta_L}$$

where $\eta_0$, $\eta_M$ and $\eta_L$ are the steady-state mean-square deviation (MSD) of the non-cooperative RLS, PDRLS, and diffRLS algorithms, respectively, in dB scale (see Section III.A ahead for the definition of the MSD).

### C. Element selection

One way to select the elements of the intermediate estimates for diffusion is to pick them randomly. In this method, at each

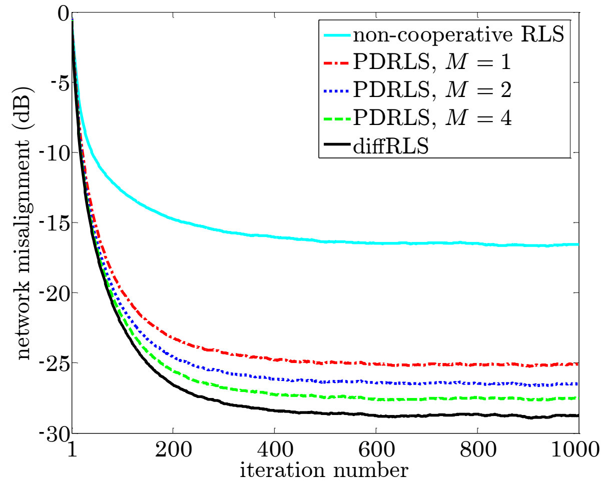

Fig. 1. MSD curves of the non-cooperative RLS algorithm, the diffRLS algorithm, and the PDRLS algorithm with different numbers of elements diffused at each iteration using the stochastic partial-diffusion scheme.

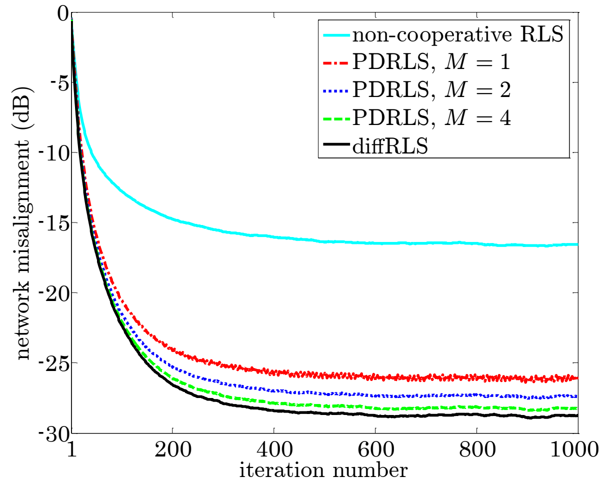

Fig. 2. MSD curves of the non-cooperative RLS algorithm, the diffRLS algorithm, and the PDRLS algorithm with different numbers of elements diffused at each iteration using the sequential partial-diffusion scheme.

iteration, $M$ out of $L$ elements of the intermediate estimate of each node are randomly selected for diffusion. We call this scheme *stochastic* partial-diffusion. If the nodes utilize pseudorandom number generators (PRNGs) for the random selection process, they may share their PRNG seeds with their neighbors only once at the beginning of adaptation and consequently eliminate the addressing overhead.

Another way is to select the $M$ to-be-diffused elements sequentially and in a round-robin fashion over the iterations (time instants). In this *sequential* partial-diffusion scheme, the elements are equitably placed in $L$ groups of size $M$ such that each element is in $M$ groups. Then, the groups are ordered in a predetermined sequence. At each iteration and in periods of $L$ iterations, only the elements of one group are diffused according to the order of the group in the sequence. The grouping and the sequence may be the same for all the nodes or alternatively different at each node. Since, in the sequential scheme, partial-diffusion scheduling is deterministic, the nodes can be informed in advance about the patterns according which their neighbors diffuse their elements. Consequently, the addressing overhead can be obviated with sequential partial-diffusion when the nodes are duly synchronized.

It is worth mentioning that the element selection processes of the abovementioned partial-diffusion schemes are analogous to the coefficient selection processes of the stochastic and sequential *partial-update* schemes [14].

## III. Numerical Studies

### A. Simulations

We consider a problem of distributed system identification over an adaptive network. The unknown system has $L = 8$ random parameters and unit energy. The network has a random topology with $N = 20$ nodes and average connectivity of four links per node. The input vectors at each node, $\mathbf{x}_{k,n}$, are i.i.d. Gaussian with covariance matrix of $\mathbf{R}_k = \mathbf{Q}_k \mathrm{diag}\{\mathbf{e}_k\}\mathbf{Q}_k^T$ where $\mathbf{Q}_k \in \mathbb{R}^{L\times L}$ is a random unitary matrix and the elements of $\mathbf{e}_k \in \mathbb{R}^{L\times 1}$ are randomly drawn from a uniform distribution in the interval $[0.2,1]$. The additive noises at the nodes, $v_{k,n}$, are also zero-mean Gaussian. The input vectors and the noises of all the nodes are independent of each other in time and space. We use relative-degree weights [9] in the combination phase, initialize the estimates to all-zero vectors, and set $\lambda = 0.995$.

In Figs. 1 and 2, we compare the performance of the non-cooperative RLS algorithm, the diffRLS algorithm, and the PDRLS algorithm with different values of $M$ for both stochastic and sequential partial-diffusion schemes by plotting the time evolution of their mean-square deviation (MSD). We define the MSD at time instant $n$ by averaging over all the nodes as

$$\frac{1}{N}\sum_{k=1}^{N} E\left[\left\|\mathbf{h} - \mathbf{w}_{k,n}\right\|^2\right]$$

and evaluate it by ensemble-averaging over $10^3$ independent runs. In the sequential scheme, all the nodes use the same element selection pattern.

In Figs. 3 and 4, we plot the communication saving, $q$, and the performance loss, $l$, as functions of the number of elements diffused at each iteration for both stochastic and sequential partial-diffusion schemes in the experiment of Figs. 1 and 2.

### B. Discussions

The simulation results show that the partial diffusion enables us to trade estimation performance for communication cost. It is clear that the larger $M$ is set, the closer to diffRLS the performance of PDRLS becomes. However, the performance degradation incurred by partial diffusion is rather graceful considering the substantial savings offered. For example, Figs. 3 and 4 show that, in the simulated scenario, diffusing only a single element of each intermediate estimate vector per iteration using the sequential partial-diffusion scheme results in 87.5% reduction in the internode communications with reference to the full-diffusion case (the diffRLS algorithm) while sustaining only a performance loss of 21.9% with respect to the diffRLS algorithm and providing a *performance gain* of 78.1% with respect to the non-cooperative case (the RLS algorithm).

Another observation is that the sequential partial-diffusion

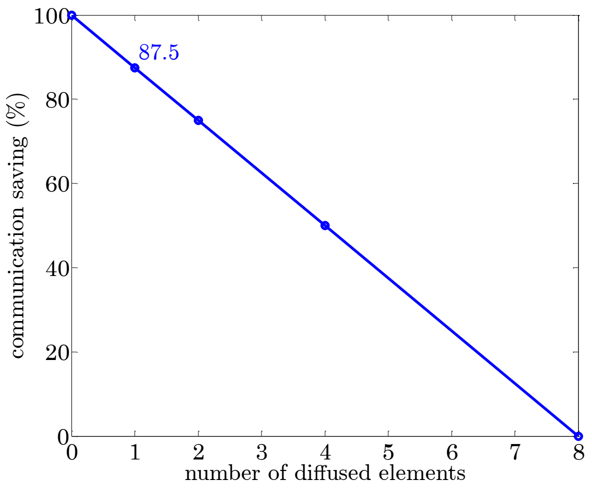

Fig. 3. Communication saving versus the number of elements diffused at each iteration for both stochastic and sequential partial-diffusion schemes.

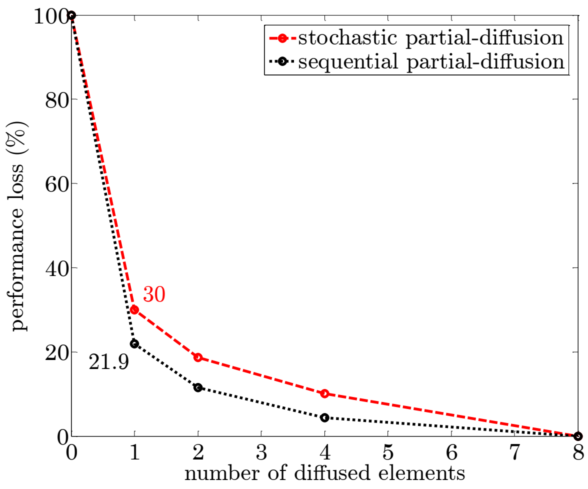

Fig. 4 Performance loss versus the number of elements diffused at each iteration for both stochastic and sequential partial-diffusion schemes in the experiment of Figs. 1 and 2.

scheme with the same element selection pattern in all the nodes outperforms the stochastic partial-diffusion scheme.

## IV. Conclusion

We introduced the notion of partial diffusion in the context of distributed estimation over adaptive networks where, at each iteration, nodes transmit a subset of the elements of their intermediate estimate vectors to their neighbors. Consequently, the amount of required internode communications is reduced compared with the conventional case where all the elements of the intermediate estimate vectors are diffused. Reduced internode communications directly translates to significant savings in power consumption and bandwidth usage. We also considered two different schemes, namely, stochastic and sequential, to select the elements of the intermediate estimate vectors for diffusion at each iteration. Based on the introduced partial-diffusion strategy, we proposed a partial-diffusion recursive least-squares (PDRLS) algorithm. Predictably, partial diffusion incurs degradation in estimation performance. However, simulation results showed that the PDRLS algorithm provides a very efficient trade-off between communication cost and estimation performance.

## Acknowledgment

This work was partly supported by the Academy of Finland.

## References

[1] A. H. Sayed, “Diffusion adaptation over networks,” *E-Reference Signal Processing*, R. Chellapa and S. Theodoridis, Eds., Elsevier, 2013, to be published; available at http://arxiv.org/abs/1205.4220.

[2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[3] C. G. Lopes and A. H. Sayed, “Incremental adaptive strategies over distributed networks,” *IEEE Trans. Signal Process.*, vol. 55, pp. 4064–4077, Aug. 2007.

[4] M. G. Rabbat and R. D. Nowak, “Quantized incremental algorithms for distributed optimization,” *IEEE J. Sel. Areas Commun.*, vol. 23, pp. 798–808, Apr. 2005.

[5] A. Nedic and D. P. Bertsekas, “Incremental subgradient methods for nondifferentiable optimization,” *SIAM J. Optim.*, vol. 12, no. 1, pp. 109–138, 2001.

[6] A. Bertrand and M. Moonen, “Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks,” *IEEE Trans. Signal Process.*, vol. 59, pp. 2320–2330, May 2011.

[7] S. S. Stankovic, M. S. Stankovic, and D. M. Stipanovic, “Decentralized parameter estimation by consensus based stochastic approximation,” *IEEE Trans. Autom. Control*, vol. 56, pp. 531–543, Mar. 2011.

[8] G. Mateos, I. D. Schizas, and G. B. Giannakis, “Distributed recursive least-squares for consensus-based in-network adaptive estimation,” *IEEE Trans. Signal Process.*, vol. 57, pp. 4583–4588, 2009.

[9] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, “Diffusion recursive least-squares for distributed estimation over adaptive networks,” *IEEE Trans. on Signal Process.*, vol. 56, pp. 1865-1877, May 2008.

[10] J. Chen and A. H. Sayed, “Diffusion adaptation strategies for distributed optimization and learning over networks,” *IEEE Trans. Signal Process.*, vol. 60, pp. 4289–4305, Aug. 2012.

[11] A. Bertrand, M. Moonen, and A. H. Sayed, “Diffusion bias-compensated RLS estimation over adaptive networks,” *IEEE Trans. Signal Process.*, vol. 59, pp. 5212–5224, Nov. 2011.

[12] S.-Y. Tu and A. H. Sayed, “Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks,” *IEEE Trans. Signal Process.*, vol. 60, pp. 6217–6234, Dec. 2012.

[13] S. Haykin, *Adaptive Filter Theory*, 4th ed., Upper Saddle River, NJ: Prentice-Hall, 2002.

[14] K. Doğançay, *Partial-Update Adaptive Signal Processing: Design, Analysis and Implementation,* Oxford, UK: Academic Press, 2008.

[15] J. R. Deller, Jr. and Y. F. Huang, “Set-membership identification and filtering for signal processing applications,” *Circuits Systems Signal Process.*, vol. 21, no. 1, pp. 69-82, 2002.

[16] S. Werner, T. Riihonen, and Y.-F. Huang, “Energy-efficient distributed parameter estimation with partial updates,” in *Proc. Int. Conf. Green Circuits Syst.*, Shanghai, China, Jun. 2010, pp. 36-40.

[17] A. Malipatil, Y.-F. Huang, and S. Werner, “An SMF approach to distributed average consensus in clustered sensor networks,” in *Proc. IEEE Int. Workshop Signal Process. Advances Wireless Commun.*, Perugia, Italy, Jun. 2009, pp. 81-85.

[18] S. Werner, Y.-F. Huang, M. L. R. de Campos, and V. Koivunen, “Distributed parameter estimation with selective cooperation,” in *Proc. Int. Conf. Acoust. Speech Signal Process.*, Taipei, Taiwan, Apr. 2009, pp. 2849-2852.

[19] S. Werner, M. Mohammed, Y.-F. Huang, and V. Koivunen, “Decentralized set-membership adaptive estimation for clustered sensor networks,” in *Proc. Int. Conf. Acoust. Speech Signal Process.*, Las Vegas, USA, Apr. 2008, pp. 3573-3576.

[20] S. Werner and Y.-F. Huang, “Time- and coefficient-selective diffusion strategies for distributed parameter estimation,” in *Proc. IEEE Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, USA, Nov. 2010, pp.696–697.

[21] G. H. Golub and C. F. Van Loan, *Matrix Computations*, third ed., Baltimore, MD: Johns Hopkins Univ. Press, 1996.