Distributed Estimation Over Sensor Networks Based on Distributed Conjugate Gradient Strategies

Songcen Xu*, Rodrigo C. de Lamare, *Senior Member, IEEE,* and H. Vincent Poor, *Fellow, IEEE*

Abstract

This paper presents distributed conjugate gradient (CG) algorithms for distributed parameter estimation and spectrum estimation over wireless sensor networks. In particular, distributed conventional conjugate gradient (CCG) and modified conjugate gradient (MCG) algorithms are developed with incremental and diffusion adaptive cooperation strategies. The distributed CCG and MCG algorithms have an improved performance in terms of mean square error as compared with least–mean square (LMS)–based algorithms and a performance that is close to recursive least–squares (RLS) algorithms. 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 design of preconditioners for CG algorithms, which can improve the performance of the proposed CG algorithms is presented. Simulations show the performance of the proposed CG algorithms against previously reported techniques for distributed parameter estimation and distributed spectrum estimation applications.

Index Terms

Distributed conjugate gradient techniques, distributed parameter estimation, distributed spectrum estimation, wireless sensor networks.

S. Xu* is with the Communications and Signal Processing Research Group, Department of Electronics, University of York, YO10 5DD York, U.K. (e-mail: songcen.xu@york.ac.uk).

R. C. de Lamare is with CETUC / PUC-Rio, Brazil and Department of Electronics, University of York, U.K. (e-mail: rodrigo.delamare@york.ac.uk).

H. V. Poor is with the Department of Electrical Engineering, Princeton University, Princeton NJ 08544 USA (e-mail: poor@princeton.edu).

Part of this work has been presented at the 2012 Conference on Sensor Signal Processing for Defence, London, UK

I. INTRODUCTION

In recent years, distributed signal processing has become a popular research topic and found many applications in wireless networks. Distributed processing of information consists of collecting data at each node of a network of sensing devices spread over a geographical area, conveying information to the whole network and performing statistical inference in a distributed way [1], [2]. These techniques exhibit several distinctive advantages such as flexibility, robustness to sensor failures and improved performance. In this context, for each specific node, a set of neighbor nodes collect their local information and transmit their estimates to a specific node. Then, each specific node combines the collected information together with its local estimate to generate an improved estimate. There are three main protocols for cooperation and exchange of information for distributed processing, incremental, diffusion and consensus strategies, and recent studies indicate that the diffusion strategy is the most effective one [3].

In the last few years, several algorithms have been developed and reported in the literature for distributed networks. Steepest-descent, least mean square (LMS) [1], recursive least squares (RLS) [4] and affine projection (AP) [5] solutions have been considered with incremental adaptive strategies over distributed networks [1], while LMS, AP and RLS algorithms have been reported using diffusion adaptive strategies [6], [7], [8], [9], [10], [11]. Although the LMS–based algorithms have their own advantages, when compared with conjugate gradient (CG) algorithms, there are several disadvantages. red First, for LMS–based algorithms, the adaptation speed is often slow. Second, with the increase of the step size and the adaptation speed, the system stability may be affected [12]. Furthermore, RLS–based algorithms are usually prone to numerical instability when implemented in hardware [13]. In order to develop distributed solutions with a more attractive tradeoff between performance and complexity, we focus on the development of distributed CG algorithms. CG–based algorithms have not been specifically developed for distributed processing in wireless sensor networks even though a distributed CG algorithm has been reported for splitting the computational load of tasks in the literature of computer science [14]. In [14], a parallel CG has been developed to employ the CG algorithm in a distributed way using multiple processors for cloud applications, while in a wireless sensor network, each node employs the CG algorithm, and then performs the distributed processing cooperatively. Unlike the work in [14], the main purpose of the proposed distributed CG algorithms with incremental and diffusion strategies is to efficiently perform distributed parameter estimation by exploiting the spatial diversity in a wireless sensor network, which results in greater estimation accuracy. The existing centralized CG algorithms have a faster convergence rate than the LMS-type algorithms and a lower computational complexity than RLS–type algorithms,

depending on the number of iterations that CG employs [15], [16], [17]. We consider variants of CG algorithms, including the conventional CG (CCG) and modified CG (MCG) algorithms [17], [18].

In this paper, we propose distributed CG algorithms for both incremental and diffusion adaptive strategies. In particular, we develop distributed versions of the CCG algorithm and of the MCG algorithm for distributed estimation and spectrum estimation using wireless sensor networks. The design of preconditioners for CG algorithms, which have the ability to improve the performance of the CG algorithms is also presented in this paper. These algorithms can be applied to civilian and defence applications, such as parameter estimation in wireless sensor networks, biomedical engineering, cellular networks, battlefield information identification, movement estimation and detection and distributed spectrum estimation.

In summary, the main contributions of this paper are:

- We devise distributed CCG and MCG algorithms for incremental and diffusion strategies to perform distributed estimation tasks.
- The design of preconditioners for CG algorithms, which have the ability to improve the performance of the proposed CG algorithms.
- A simulation study of the proposed and existing distributed estimation algorithms with applications to distributed parameter estimation and spectrum estimation.

This paper is organized as follows. Section II describes the signal models. In Section III, the proposed incremental distributed CG–based algorithms are introduced. We present the proposed diffusion distributed CG–Based algorithms in Section IV. The preconditioner design is illustrated in Section V. The numerical simulation results are provided in Section VI. Finally, we conclude the paper in Section VII.

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.

II. Signal Models

In this section, we describe the signal models of two applications of distributed signal processing, namely, parameter estimation and spectrum estimation. In these applications, we consider a wireless sensor network which employs distributed signal processing techniques to perform the desired tasks. We consider a set of N nodes, distributed over a given geographical area. 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 [19].

A. Distributed Parameter Estimation

In distributed parameter estimation, we focus on the processing of estimating an unknown vector $\boldsymbol{\omega}_0$ with size $M \times 1$. The desired signal of each node at time instant i is

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

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ input signal vector, $n_k(i)$ is the Gaussian noise at each node with zero mean and variance $\sigma_{n,k}^2$. At the same time, the output of the adaptive algorithm for each node is given by

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

where $\boldsymbol{\omega}_k(i)$ is the local estimate of $\boldsymbol{\omega}_0$ for each node at time instant i.

To compute an estimate of the unknown vector, we need to solve a problem expressed in the form of a minimization of the cost function in the distributed form for each node k:

$$J_k\big(\boldsymbol{\omega}_k(i)\big) = \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big|^2 \tag{3}$$

and the global network cost function could be described as

$$J\big(\boldsymbol{\omega}_k(i)\big) = \sum_{k=1}^{N} \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big|^2. \tag{4}$$

The optimum solution for the minimization of the cost function (3) is the Wiener solution which is given by

$$\boldsymbol{\omega}_0 = \boldsymbol{R}_k^{-1}(i)\boldsymbol{b}_k(i). \tag{5}$$

where the $M \times M$ autocorrelation matrix is given by $\boldsymbol{R}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]$ and $\boldsymbol{b}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)d_k^*(i)]$ is an $M \times 1$ cross–correlation matrix. In this chapter, we focus on incremental and diffusion CG–based algorithms to solve the linear system of equations and perform parameter estimation and spectrum estimation in a distributed fashion.

B. Distributed Spectrum Estimation

In distributed spectrum estimation, we aim to estimate the spectrum of a transmitted signal $\boldsymbol{s}$ with N nodes using a wireless sensor network. Let $\boldsymbol{\Phi}_s(f)$ denote the power spectral density (PSD) of the signal $\boldsymbol{s}$. The PSD can be represented as a linear combination of some $\mathcal{B}$ basis functions, as described by

$$\Phi_s(f) = \sum_{m=1}^{\mathcal{B}} b_m(f)\omega_{0m} = \boldsymbol{b}_0^T(f)\boldsymbol{\omega}_0, \tag{6}$$

where $\boldsymbol{b}_0(f) = [b_1(f), ..., b_{\mathcal{B}}(f)]^T$ is the vector of basis functions evaluated at frequency f, $\boldsymbol{\omega}_0 = [\omega_{01}, ..., \omega_{0\mathcal{B}}]$ is a vector of weighting coefficients representing the power that transmits the signal $\boldsymbol{s}$ over each basis, and $\mathcal{B}$ is the number of basis functions. For $\mathcal{B}$ sufficiently large, the basis expansion in (6) can approximate well the transmitted spectrum. Possible choices for the set of basis $\{b_m(f)\}_{m=1}^{\mathcal{B}}$ include [20], [21], [22]: rectangular functions, raised cosines, Gaussian bells and Splines.

Let $H_k(f, i)$ be the channel transfer function between a transmit node conveying the signal $\boldsymbol{s}$ and receive node k at time instant i, the PSD of the received signal observed by node k can be expressed as

$$\begin{aligned} I_k(f,i) &= |H_k(f,i)|^2 \Phi_s(f) + v_k^2 \\ &= \sum_{m=1}^{\mathcal{B}} |H_k(f,i)|^2 b_m(f) \omega_{0m} + v_k^2 \\ &= \boldsymbol{b}_{k,i}^T(f) \boldsymbol{\omega}_0 + v_k^2 \end{aligned} \tag{7}$$

where $\boldsymbol{b}_{k,i}^T(f) = [|H_k(f,i)|^2 b_m(f)]_{m=1}^{\mathcal{B}}$ and v_k^2 is the receiver noise power at node k.

At every time instant i, every node k observes measurements of the noisy version of the true PSD $I_k(f, i)$ described by (7) over N_c frequency samples $f_j = f_{min} : (f_{max} - f_{min})/N_c : f_{max}$, for $j = 1, ..., N_c$, according to the model:

$$d_k^j(i) = \boldsymbol{b}_{k,i}^T(f_j) \boldsymbol{\omega}_0 + v_k^2 + n_k^j(i). \tag{8}$$

The term $n_k^j(i)$ denotes sampling noise and have zero mean and variance $\sigma_{n,j}^2$. The receiver noise power $v_{n,k}^2$ can be estimated with high accuracy in a preliminary step using, e.g., an energy estimator over an idle band, and then subtracted from (8) [23], [24]. Thus, collecting measurements over N_c contiguous channels, we obtain a linear model given by

$$\boldsymbol{d}_k(i) = \boldsymbol{B}_k(i) \boldsymbol{\omega}_0 + \boldsymbol{n}_k(i), \tag{9}$$

where $\boldsymbol{B}_k(i) = [\boldsymbol{b}_{k,i}^T(f_j)]_{j=1}^{N_c} \in \mathbb{R}^{N_c \times \mathcal{B}}$, with $N_c > \mathcal{B}$, and $\boldsymbol{n}_k(i) = [n_k^1(i), ...n_k^{N_c}(i)]^T$. At this point, we can generate the cost function for node k as:

$$J_{\boldsymbol{\omega}_k(i)}(\boldsymbol{\omega}_k(i)) = \mathbb{E}\big|\boldsymbol{d}_k(i) - \boldsymbol{B}_k(i)\boldsymbol{\omega}_k(i)\big|^2 \tag{10}$$

and the global network cost function could be described as

$$J_{\boldsymbol{\omega}}\big(\boldsymbol{\omega}\big) = \sum_{k=1}^{N} \mathbb{E}\big|\boldsymbol{d}_k(i) - \boldsymbol{B}_k(i)\boldsymbol{\omega}\big|^2. \tag{11}$$

III. Proposed Incremental Distributed CG–Based Algorithms

In this section, we propose two distributed CG–based algorithms which are based on the centralized CCG [17] and MCG [25] algorithms with incremental distributed solution for distributed parameter estimation and spectrum estimation over wireless sensor networks. Before we present the proposed incremental distributed CG–Based algorithms, we review the basic centralized CG algorithm in detail.

A. The Conjugate Gradient (CG) Algorithm

The conjugate gradient (CG) algorithm is well known for its faster convergence rate than the LMS algorithm and a lower computational complexity than RLS–type algorithms, depending on the number of iterations that the CG algorithm employs [15], [16], [17]. In adaptive filtering techniques, the CG algorithm applied to the system $\boldsymbol{R}\boldsymbol{\omega} = \boldsymbol{b}$, starts with an initial guess of the solution $\boldsymbol{\omega}(0)$, with an initial residual $\boldsymbol{g}(0) = \boldsymbol{b}$, and with an initial search direction that is equal to the initial residual: $\boldsymbol{p}(0) = \boldsymbol{g}(0)$, where $\boldsymbol{R}$ is the correlation matrix of the input signal and $\boldsymbol{b}$ is the cross–correlation vector between the desired signal and the input signal.

The strategy for the conjugate gradient method is that at step j, the residual $\boldsymbol{g}(j) = \boldsymbol{b} - \boldsymbol{R}\boldsymbol{\omega}(j)$ is orthogonal to the Krylov subspace generated by $\boldsymbol{R}$ and $\boldsymbol{b}$ [15], and therefore each residual is orthogonal to all the previous residuals. The residual is computed at each step. The solution at the next step is computed using a search direction that is a linear combination of the previous search directions, which for $\boldsymbol{\omega}(1)$ is just a combination between the previous and the current residual.

Then, the solution at step j, $\boldsymbol{\omega}(j)$, could be obtained through $\boldsymbol{\omega}(j-1)$ from the previous iteration plus a step size $\alpha(j)$ times the last search direction. The immediate benefit of the search directions is that there is no need to store all the previous search directions, only the search direction from the last step is needed. Using the orthogonality of the residuals to these previous search directions, the search is linearly independent of the previous directions. For the solution in the next step, a new search direction is computed, as well as a new residual and new step size. To provide an optimal approximate solution of $\boldsymbol{\omega}$, the step size $\alpha(j)$ is calculated as $\alpha(j) = \frac{\boldsymbol{g}(j-1)^H \boldsymbol{g}(j-1)}{\boldsymbol{p}(j-1)^H \boldsymbol{R}\boldsymbol{p}(j-1)}$ according to [15], [16], [17]. The main steps and recursions of the centralized CG algorithm [15], [16], [17] are summarized in Table I.

In this section, we propose two CG–based algorithms which are based on the CCG [17] and MCG [25] algorithms with incremental distributed solution for distributed parameter estimation and spectrum estimation over wireless sensor networks.

TABLE I

MAIN STEPS FOR CG ALGORITHM

Initial conditions:
$\boldsymbol{\omega}(0) = 0$, $\boldsymbol{g}(0) = \boldsymbol{b}$, $\boldsymbol{p}(0) = \boldsymbol{g}(0)$
– Step size: $\alpha(j) = \frac{\boldsymbol{g}(j-1)^H \boldsymbol{g}(j-1)}{\boldsymbol{p}(j-1)^H \boldsymbol{R}\boldsymbol{p}(j-1)}$
– Approximate solution: $\boldsymbol{\omega}(j) = \boldsymbol{\omega}(j-1) + \alpha(j)\boldsymbol{p}(j-1)$
– Residual: $\boldsymbol{g}(j) = \boldsymbol{g}(j-1) - \alpha(j)\boldsymbol{R}\boldsymbol{p}(j-1)$
– Improvement at step i: $\beta(j) = \frac{\boldsymbol{g}(j)^H \boldsymbol{g}(j)}{\boldsymbol{g}(j-1)^H \boldsymbol{g}(j-1)}$
– Search direction: $\boldsymbol{p}(j) = \boldsymbol{g}(j) + \beta(j)\boldsymbol{p}(j-1)$

B. Incremental Distributed CG–Based Solutions

In the incremental distributed strategy, each node is only allowed to communicate with its direct neighbor at each time instant. To describe the whole process, we define a cycle where each node in this network could only access its immediate neighbor in this cycle [1]. The quantity $\boldsymbol{\psi}_k(i)$ is defined as a local estimate of the unknown vector $\boldsymbol{\omega}_0$ at time instant i. As a result, we assume that node k has access to an estimate of $\boldsymbol{\omega}_0$ at its immediate neighbor node $k-1$ which is $\boldsymbol{\psi}_{k-1}(i)$ in the defined cycle. Fig.1 illustrates the main processing steps. In the following, we introduce two kinds of incremental

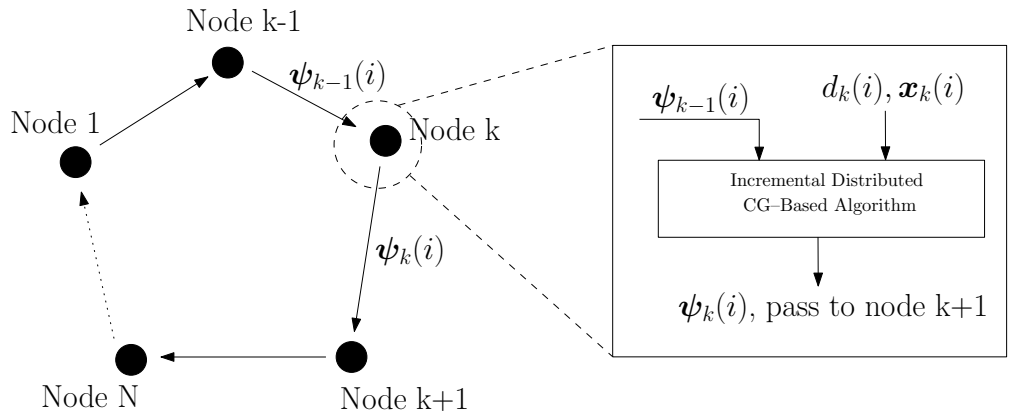

Fig. 1. Incremental distributed CG–based network processing

distributed CG–based algorithms, which are the incremental distributed CCG (IDCCG) algorithm and the incremental distributed MCG (IDMCG) algorithm.

1) Proposed IDCCG Algorithm: Based on the main steps of CG algorithm which are described in Table. I, we introduce the main steps of the proposed IDCCG algorithm. In the IDCCG algorithm, the

iteration procedure is introduced. At the jth iteration of time instant i, the step size $\alpha_k^j(i)$ for updating the local estimate at node k is defined as:

$$\alpha_k^j(i) = \frac{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}{\left(\boldsymbol{p}_k^{j-1}(i)\right)^H \boldsymbol{R}_k(i) \boldsymbol{p}_k^{j-1}(i)}, \tag{12}$$

where $\boldsymbol{p}_k^j(i)$ is the search direction and defined as:

$$\boldsymbol{p}_k^j(i) = \boldsymbol{g}_k^j(i) + \beta_k^j(i) \boldsymbol{p}_k^{j-1}(i). \tag{13}$$

In (13), the coefficient $\beta_k^j(i)$ is calculated by the Gram–Schmidt orthogonalization procedure [16] for conjugacy:

$$\beta_k^j(i) = \frac{\left(\boldsymbol{g}_k^j(i)\right)^H \boldsymbol{g}_k^j(i)}{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}. \tag{14}$$

$\boldsymbol{g}_k^j(i)$ is the residual, which is obtained as

$$\boldsymbol{g}_k^j(i) = \boldsymbol{g}_k^{j-1}(i) - \alpha_k^j(i) \boldsymbol{R}_k(i) \boldsymbol{p}_k^{j-1}(i). \tag{15}$$

The initial search direction is equal to the initial residual, which is given by $\boldsymbol{p}_k^0(i) = \boldsymbol{g}_k^0(i) = \boldsymbol{b}_k(i) - \boldsymbol{R}_k(i) \boldsymbol{\psi}_k^0(i)$. Then, the local estimate is updated as

$$\boldsymbol{\psi}_k^j(i) = \boldsymbol{\psi}_k^{j-1}(i) + \alpha_k^j(i) \boldsymbol{p}_k^{j-1}(i). \tag{16}$$

There are two ways to compute the correlation and cross–correlation matrices which are the 'finite sliding data window' and the 'exponentially decaying data window' [17]. In this chapter, we focus on the 'exponentially decaying data window'. The recursions are given by:

$$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i) \boldsymbol{x}_k^H(i) \tag{17}$$

and

$$\boldsymbol{b}_k(i) = \lambda_f \boldsymbol{b}_k(i-1) + d_k^*(i) \boldsymbol{x}_k(i) \tag{18}$$

where λ_f is the forgetting factor. The IDCCD algorithm is summarized in Table II

2) Proposed IDMCG Algorithm: The idea of the IDMCG algorithm comes from the existing centralized MCG algorithm. In the IDMCG algorithm, a recursive formulation for the residual vector is employed, which can be found by using (12), (17) and (18) [17], [18], resulting in

$$\begin{aligned} \boldsymbol{g}_k(i) &= \boldsymbol{b}_k(i) - \boldsymbol{R}_k(i) \boldsymbol{\psi}_k(i) \\ &= \lambda_f \boldsymbol{g}_k(i-1) - \alpha_k(i) \boldsymbol{R}_k(i) \boldsymbol{p}_k(i-1) + \boldsymbol{x}_k(i) [d_k(i) - \boldsymbol{\psi}_{k-1}^H(i) \boldsymbol{x}_k(i)]. \end{aligned} \tag{19}$$

Analysis:

TABLE II

THE PROPOSED IDCCG ALGORITHM

	Multiplications	Additions
Initialization:		
$\boldsymbol{\omega}(0)=\mathbf{0}$		
For each time instant $i=1,2,\ldots,I$		
$\boldsymbol{\psi}_1^0(i)=\boldsymbol{\omega}(i-1)$		
For each node $k=1,2,\ldots,N$		
$\boldsymbol{R}_k(i)=\lambda_f\boldsymbol{R}_k(i-1)+\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$	$2M^2$	M^2
$\boldsymbol{b}_k(i)=\lambda_f\boldsymbol{b}_k(i-1)+d_k^*(i)\boldsymbol{x}_k(i)$	$2M$	M
$\boldsymbol{p}_k^0(i)=\boldsymbol{g}_k^0(i)=\boldsymbol{b}_k(i)-\boldsymbol{R}_k(i)\boldsymbol{\psi}_k^0(i)$	M^2	$2M^2$
For iterations $j=1,2,\ldots,J$		
$\alpha_k^j(i)=\frac{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H\boldsymbol{g}_k^{j-1}(i)}{\left(\boldsymbol{p}_k^{j-1}(i)\right)^H\boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)}$	M^2+2M+1	M^2+M-2
$\boldsymbol{\psi}_k^j(i)=\boldsymbol{\psi}_k^{j-1}(i)+\alpha_k^j(i)\boldsymbol{p}_k^{j-1}(i)$	M	M
$\boldsymbol{g}_k^j(i)=\boldsymbol{g}_k^{j-1}(i)-\alpha_k^j(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)$	M	M
$\beta_k^j(i)=\frac{\left(\boldsymbol{g}_k^j(i)\right)^H\boldsymbol{g}_k^j(i)}{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H\boldsymbol{g}_k^{j-1}(i)}$	$M+1$	$M-1$
$\boldsymbol{p}_k^j(i)=\boldsymbol{g}_k^j(i)+\beta_k^j(i)\boldsymbol{p}_k^{j-1}(i)$	M	M
End		
When $k<N$		
$\boldsymbol{\psi}_{k+1}^0(i)=\boldsymbol{\psi}_k^J(i)$		
End		
$\boldsymbol{\omega}(i)=\boldsymbol{\psi}_N^J(i)$		
End		

In order to analyze the proposed IDMCG algorithm, we start by premultiplying (19) by $\boldsymbol{p}_k^H(i-1)$, which gives

$$\begin{aligned}\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i)=\lambda_f\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)-\alpha_k(i)\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)\\+\boldsymbol{p}_k^H(i-1)\boldsymbol{x}_k(i)[d_k(i)-\boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)].\end{aligned}\tag{20}$$

Taking the expectation of both sides and considering $\boldsymbol{p}_k(i-1)$ uncorrelated with $\boldsymbol{x}_k(i)$, $d_k(i)$ and $\boldsymbol{\psi}_{k-1}(i)$ yields

$$\begin{aligned}\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i)]\approx\lambda_f\mathbb{E}[\boldsymbol{p}_k(i-1)^H\boldsymbol{g}_{k-1}(i)]-\mathbb{E}[\alpha_k(i)]\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)]\\+\mathbb{E}[\boldsymbol{p}_k^H(i-1)]\mathbb{E}\big[\boldsymbol{x}_k(i)[d_k(i)-\boldsymbol{\omega}_{k-1}^H(i)\boldsymbol{x}_k(i)]\big].\end{aligned}\tag{21}$$

Assuming that the algorithm converges, the last term of (21) can be neglected and we obtain [17]:

$$\mathbb{E}[\alpha_k(i)] = \frac{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i)] - \lambda_f \mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)]}{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)]} \tag{22}$$

and the $\mathbb{E}[\alpha_k(i)]$ should satisfy the following equation [17]:

$$(\lambda_f - 0.5)\frac{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)]}{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)]} \leq \mathbb{E}[\alpha_k(i)] \leq \frac{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)]}{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)]}, \tag{23}$$

which establishes the bounds on the step size used by the proposed IDMCG algorithm.

The inequalities in (23) are satisfied if we define [17]:

$$\alpha_k(i) = \eta \frac{\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)}{\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)}, \tag{24}$$

where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$.

The direction vector $\boldsymbol{p}_k(i)$ for the IDMCG algorithm is defined by

$$\boldsymbol{p}_k(i) = \boldsymbol{g}_k(i) + \beta_k(i)\boldsymbol{p}_k(i-1). \tag{25}$$

For the IDMCG algorithm, for the computation of $\beta_k(i)$, the Polak–Ribiere method [17], which is given by

$$\beta_k(i) = \frac{\big(\boldsymbol{g}_k(i) - \boldsymbol{g}_k(i-1)\big)^H \boldsymbol{g}_k(i)}{\boldsymbol{g}_k^H(i-1)\boldsymbol{g}_k(i-1)} \tag{26}$$

should be used for improved performance, according to [26], [27].

In the comparison of the IDCCG algorithm with the IDMCG algorithm, the difference between these two strategies is that IDCCG needs to run J iterations while IDMCG only needs one iteration. The details of the IDMCG solution are shown in Table III.

C. Computational Complexity

To analyze the proposed incremental distributed CG algorithms, we detail the computational complexity in terms of arithmetic operations. Additions and multiplications are used to measure the complexity and are listed in Table IV. The parameter M is the length of the unknown vector $\boldsymbol{\omega}_0$ that needs to be estimated. Detailed computational complexity of each step for the proposed incremental distributed CG algorithms are shown in Table II and III. When we compare the IDMCG and IDCCG algorithms with the RLS–type algorithm, it can be seen that the complexity of IDMCG is lower than the RLS algorithm, while the complexity of the IDCCG algorithm depends on the number of iterations J.

TABLE III

THE PROPOSED IDMCG ALGORITHM

	Multiplications	Additions
Initialization:		
$\boldsymbol{\omega}(0) = \mathbf{0}$		
For each node $k = 1, 2, \ldots, N$		
$\boldsymbol{b}_k(1) = d_k^*(1)\boldsymbol{x}_k(1)$		
$\boldsymbol{p}_k(0) = \boldsymbol{g}_k(0) = \boldsymbol{b}_k(1)$		
End		
For each time instant $i = 1, 2, \ldots, I$		
$\boldsymbol{\psi}_0(i) = \boldsymbol{\omega}(i-1)$		
For each node $k = 1, 2, \ldots, N$		
$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$	$2M^2$	M^2
$\alpha_k(i) = \eta \frac{\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)}{\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)}$	$M^2 + 2M + 1$	$M^2 + M - 2$
where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$		
$\boldsymbol{\psi}_k(i) = \boldsymbol{\psi}_{k-1}(i) + \alpha_k(i)\boldsymbol{p}_k(i-1)$	M	M
$\boldsymbol{g}_k(i) = \lambda_f \boldsymbol{g}_k(i-1) - \alpha_k(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)$ $+\boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)]$	$4M$	$3M$
$\beta_k(i) = \frac{\left(\boldsymbol{g}_k(i) - \boldsymbol{g}_k(i-1)\right)^H \boldsymbol{g}_k(i)}{\boldsymbol{g}_k^H(i-1)\boldsymbol{g}_k(i-1)}$	$2M + 1$	$3M - 2$
$\boldsymbol{p}_k(i) = \boldsymbol{g}_k(i) + \beta_k(i)\boldsymbol{p}_k(i-1)$	M	M
End		
$\boldsymbol{\omega}(i) = \boldsymbol{\psi}_N(i)$		
End		

TABLE IV

COMPUTATIONAL COMPLEXITY OF DIFFERENT INCREMENTAL ALGORITHMS PER NODE

Algorithm	Additions	Multiplications
IDCCG	$2M^2 + M$ $+J(M^2 + 5M - 4)$	$3M^2 + 2M$ $J(M^2 + 6M + 2)$
IDMCG	$2M^2 + 9M - 4$	$3M^2 + 10M + 2$
Incremental LMS [1]	$4M - 1$	$3M + 1$
Incremental RLS [1]	$4M^2 + 12M + 1$	$4M^2 + 12M - 1$

IV. Proposed Diffusion Distributed CG–Based Algorithms

In this section, we detail the proposed diffusion distributed CCG (DDCCG) and diffusion distributed MCG (DDMCG) algorithms for distributed parameter estimation and spectrum estimation using wireless sensor networks.

A. Diffusion Distributed CG–Based Algorithms

In the derivation of diffusion distributed CG–based strategy, we consider a network structure where each node from the same neighborhood could exchange information with each other at every time instant. For each node in the network, the CTA scheme [28] is employed. Each node can collect information from all its neighbors and itself, and then convey all the information to its local adaptive algorithm and update the estimate of the weight vector through our algorithms. Specifically, at any time instant i, we assume that node k has access to a set of estimates $\{\boldsymbol{\omega}_l(i-1)\}_{l\in\mathcal{N}_k}$ from its neighbors, where $\mathcal{N}_k$ denotes the set of neighbor nodes of node k including node k itself. Then, these local estimates are combined at node k as

$$\boldsymbol{\psi}_k(i) = \sum_{l\in\mathcal{N}_k} c_{kl}\boldsymbol{\omega}_l(i-1) \tag{27}$$

where c_{kl} is calculated through the Metropolis rule due to its simplicity and good performance [29].

$$c_{kl} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_k|,|\mathcal{N}_l|\}}, & \text{if } k\neq l \text{ are linked} \\ 1-\sum\limits_{l\in\mathcal{N}_k\backslash k} c_{kl}, & \text{for } k = l, \end{cases} \tag{28}$$

where $|\mathcal{N}_k|$ denotes the cardinality of $\mathcal{N}_k$. For the proposed diffusion distributed CG–based algorithms, the main signal processing steps are shown in Fig. 2.

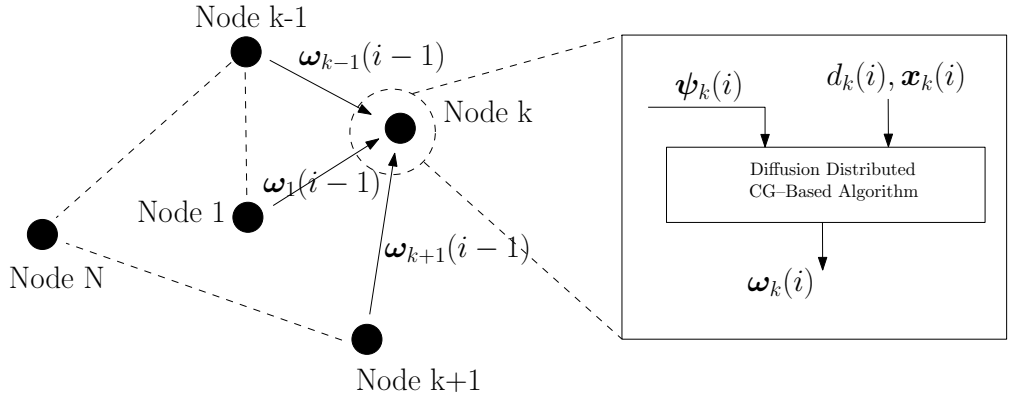

Fig. 2. Diffusion Distributed CG–Based Network Processing

TABLE V

THE PROPOSED DDCCG ALGORITHM

Initialization:

$\boldsymbol{\omega}_k(0) = \mathbf{0}, k = 1, 2, \ldots, N$

For each time instant $i = 1, 2, \ldots, I$

For each node $k = 1, 2, \ldots, N$ (Combination Step)

$\boldsymbol{\psi}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\omega}_l(i-1)$

End

For each node $k = 1, 2, \ldots, N$ (Adaptation Step)

$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$

$\boldsymbol{b}_k(i) = \lambda_f \boldsymbol{b}_k(i-1) + d_k^*(i)\boldsymbol{x}_k(i)$

$\boldsymbol{\omega}_k^0(i) = \boldsymbol{\psi}_k(i)$

$\boldsymbol{p}_k^0(i) = \boldsymbol{g}_k^0(i) = \boldsymbol{b}_k(i) - \boldsymbol{R}_k(i)\boldsymbol{\omega}_k^0(i)$

For iterations $j = 1, 2, \ldots, J$

$\alpha_k^j(i) = \frac{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}{\left(\boldsymbol{p}_k^{j-1}(i)\right)^H \boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)}$

$\boldsymbol{\omega}_k^j(i) = \boldsymbol{\omega}_k^{j-1}(i) + \alpha_k^j(i)\boldsymbol{p}_k^{j-1}(i)$

$\boldsymbol{g}_k^j(i) = \boldsymbol{g}_k^{j-1}(i) - \alpha_k^j(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)$

$\beta_k^j(i) = \frac{\left(\boldsymbol{g}_k^j(i)\right)^H \boldsymbol{g}_k^j(i)}{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}$

$\boldsymbol{p}_k^j(i) = \boldsymbol{g}_k^j(i) + \beta_k^j(i)\boldsymbol{p}_k^{j-1}(i)$

End

$\boldsymbol{\omega}_k(i) = \boldsymbol{\omega}_k^J(i)$

End

End

1) Proposed DDCCG Algorithm: In the DDCCG algorithm, (27) is employed to combine the estimates $\boldsymbol{\omega}_l(i-1), l \in \mathcal{N}_k$ from node k's neighbor nodes and then the estimate at node k is updated as follows:

$$\boldsymbol{\omega}_k^j(i) = \boldsymbol{\omega}_k^{j-1}(i) + \alpha_k^j(i)\boldsymbol{p}_k^{j-1}(i), \tag{29}$$

where $\boldsymbol{\omega}_k^0(i) = \boldsymbol{\psi}_k(i)$. The rest of the derivation is similar to the IDCCG solution and the pseudo–code is detailed in Table V.

2) Proposed DDMCG Algorithm: For the DDMCG algorithm, the iteration j is removed and the estimate at node k is updated as:

$$\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k(i) + \alpha_k(i)\boldsymbol{p}_k(i), \tag{30}$$

The complete DDMCG algorithm is described in Table VI.

TABLE VI

THE PROPOSED DDMCG ALGORITHM

Initialization:

$\boldsymbol{\omega}_k(0) = \mathbf{0}, k = 1, 2, \ldots, N$

For each node $k = 1, 2, \ldots, N$

$\boldsymbol{b}_k(1) = d_k^*(1)\boldsymbol{x}_k(1)$

$\boldsymbol{p}_k(0) = \boldsymbol{g}_k(0) = \boldsymbol{b}_k(1)$

End

For each time instant $i = 1, 2, \ldots, I$

For each node $k = 1, 2, \ldots, N$(Combination Step)

$\boldsymbol{\psi}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl}\boldsymbol{\omega}_l(i-1)$

End

For each node $k = 1, 2, \ldots, N$(Adaptation Step)

$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$

$\boldsymbol{b}_k(i) = \lambda_f \boldsymbol{b}_k(i-1) + d_k^*(i)\boldsymbol{x}_k(i)$

$\alpha_k(i) = \eta \frac{\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)}{\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)}$

where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$

$\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k(i) + \alpha_k(i)\boldsymbol{p}_k(i-1)$

$\boldsymbol{g}_k(i) = \lambda_f \boldsymbol{g}_k(i-1) - \alpha_k(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1) + \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)]$

$\beta_k(i) = \frac{\left(\boldsymbol{g}_k(i) - \boldsymbol{g}_k(i-1)\right)^H \boldsymbol{g}_k(i)}{\boldsymbol{g}_k^H(i-1)\boldsymbol{g}_k(i-1)}$

$\boldsymbol{p}_k(i) = \boldsymbol{g}_k(i) + \beta_k(i)\boldsymbol{p}_k(i-1)$

End

End

B. Computational Complexity

The proposed diffusion distributed CG–based algorithms are analysed in terms of computational complexity, where additions and multiplications are measured. The details are listed in Table VII. Similarly to the incremental distributed CG–based algorithms, it is clear that the complexity of the DDCCG solution depends on the iteration number J and both DDCCG and DDMCG solutions depend on the number of neighbor nodes $|\mathcal{N}_k|$ of node k. The parameter M is the length of the unknown vector $\boldsymbol{\omega}_0$ that needs to be estimated.

V. PRECONDITIONER DESIGN

Preconditioning is an important technique which can be used to improve the performance of CG algorithms [30], [31], [32], [33]. The idea behind preconditioning is to employ the CG algorithms on an

TABLE VII

COMPUTATIONAL COMPLEXITY OF DIFFERENT DIFFUSION ALGORITHMS PER NODE

Algorithm	Additions	Multiplications
DDCCG	$2M^2 + M$ $+J(M^2 + 5M$ $+\|\mathcal{N}_k\|M - 4)$	$3M^2 + 2M$ $+J(M^2 + 6M$ $+\|\mathcal{N}_k\|M + 2)$
DDMCG	$2M^2 + 9M - 4$ $+\|\mathcal{N}_k\|M$	$3M^2 + 10M + 2$ $+\|\mathcal{N}_k\|M$
Diffusion LMS [28]	$4M - 1 + \|\mathcal{N}_k\|M$	$3M + 1 + \|\mathcal{N}_k\|M$
Diffusion RLS [9]	$4M^2 + 16M + 1 + \|\mathcal{N}_k\|M$	$4M^2 + 12M - 1 + \|\mathcal{N}_k\|M$

equivalent system or in a transform–domain. Thus, instead of solving $\boldsymbol{R}\boldsymbol{\omega} = \boldsymbol{b}$ we solve a related problem $\tilde{\boldsymbol{R}}\tilde{\boldsymbol{\omega}} = \tilde{\boldsymbol{b}}$, which is modified with the aim of obtaining better convergence and steady state performance. The relationships between these two equations are given by

$$\tilde{\boldsymbol{R}} = \boldsymbol{T}\boldsymbol{R}\boldsymbol{T}^H, \tag{31}$$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{T}\boldsymbol{\omega} \tag{32}$$

and

$$\tilde{\boldsymbol{b}} = \boldsymbol{T}\boldsymbol{b}, \tag{33}$$

where the $M \times M$ matrix $\boldsymbol{T}$ is called a preconditioner. We design the matrix $\boldsymbol{T}$ as an arbitrary unitary matrix of size $M \times M$ and has the following property [34]

$$\boldsymbol{T}\boldsymbol{T}^H = \boldsymbol{T}^H\boldsymbol{T} = \boldsymbol{I}. \tag{34}$$

Two kinds of unitary transformations are considered to build the preconditioner $\boldsymbol{T}$, which are discrete Fourier transform (DFT) and discrete cosine transform (DCT) [34]. The motivation behind employing these two matrix is they have useful de–correlation properties and often reduce the eigenvalue spread of the auto–correlation matrix of the input signal [34].

For the DFT scheme, we employ the following expression

$$[\boldsymbol{T}_{DFT}]_{vm} \triangleq \frac{1}{\sqrt{M}} e^{-\frac{j2\pi mv}{M}}, \quad v, m = 0, 1, 2, \ldots, M-1, \tag{35}$$

where v indicates the row index and m the column index. M is the length of the unknown parameter $\boldsymbol{\omega}_0$. The matrix form of $\boldsymbol{T}_{DFT}$ is illustrated as

$$\boldsymbol{T}_{DFT} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{j2\pi}{M}} & e^{-\frac{j4\pi}{M}} & \cdots & e^{-\frac{j2(M-1)\pi}{M}} \\ 1 & e^{-\frac{j4\pi}{M}} & e^{-\frac{j8\pi}{M}} & \cdots & e^{-\frac{j4(M-1)\pi}{M}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-\frac{j2(M-1)\pi}{M}} & e^{-\frac{j4(M-1)\pi}{M}} & \cdots & e^{-\frac{j2(M-1)^2\pi}{M}} \end{bmatrix} \tag{36}$$

For the DCT scheme, the preconditioner $\boldsymbol{T}$ is defined as

$$[\boldsymbol{T}_{DCT}]_{vm} \triangleq \delta(v) \cos\left(\frac{v(2m+1)\pi}{2M}\right), \quad v, m = 0, 1, 2, \ldots, M-1, \tag{37}$$

where

$$\delta(0) = \frac{1}{\sqrt{M}} \quad \text{and} \quad \delta(v) = \sqrt{\frac{2}{M}} \text{ for } v \neq 0 \tag{38}$$

and the matrix form of $\boldsymbol{T}_{DCT}$ is illustrated as

$$\boldsymbol{T}_{DCT} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \sqrt{2}\cos(\frac{3\pi}{2M}) & \sqrt{2}\cos(\frac{5\pi}{2M}) & \cdots & \sqrt{2}\cos(\frac{(2M-1)\pi}{2M}) \\ 1 & \sqrt{2}\cos(\frac{6\pi}{2M}) & \sqrt{2}\cos(\frac{10\pi}{2M}) & \cdots & \sqrt{2}\cos(\frac{2(2M-1)\pi}{2M}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \sqrt{2}\cos(\frac{3(M-1)\pi}{2M}) & \sqrt{2}\cos(\frac{5(M-1)\pi}{2M}) & \cdots & \sqrt{2}\cos(\frac{(2M-1)(M-1)\pi}{2M}) \end{bmatrix} \tag{39}$$

Then, for the DCT scheme, we choose $\boldsymbol{T} = \boldsymbol{T}_{DCT}^H$. It should be noticed that the scaling factor $\frac{1}{\sqrt{M}}$ is added in the expression for the $\boldsymbol{T}_{DFT}$ in order to result in a unitary transformation since then $\boldsymbol{T}_{DFT}$ satisfies $\boldsymbol{T}_{DFT}\boldsymbol{T}_{DFT}^H = \boldsymbol{T}_{DFT}^H\boldsymbol{T}_{DFT} = \boldsymbol{I}$ [34].

The optimal selection of the preconditioner is the Kahunen–Loève transform (KLT) [34]. However, using KLT is not practical since it requires knowledge of the auto–correlation matrix $\boldsymbol{R}$ of the input signal and this information is generally lacking in implementations.

VI. Simulation Results

In this section, we investigate the performance of the proposed incremental and diffusion distributed CG–based algorithms in two scenarios: distributed estimation and distributed spectrum estimation in wireless sensor networks.

A. Distributed Estimation in Wireless Sensor Networks

In this subsection, we compare the proposed incremental and diffusion distributed CG–based algorithms with LMS [1], [28] and RLS [1], [9] algorithms, based on the MSE and MSD performance metrics. For each comparison, the number of time instants is set to 1000, and we assume there are 20 nodes in the network. The length of the unknown parameter vector $\boldsymbol{\omega}_0$ is 10, the variances for the input signal and the noise are 1 and 0.001, respectively. In addition, the noise samples are modeled as complex circular Gaussian noise with zero mean.

1) Performance of Proposed Incremental Distributed CG–Based Algorithms: First, we define the parameters of the performance test for each algorithm and the network. The step size μ for the LMS algorithm [1] is set to 0.2, the forgetting factor λ for the RLS [1] algorithm is set to 0.998. The parameter λ_f for IDCCG and IDMCG is set to 0.998. For IDMCG, the step size η is equal to 0.55. The iteration number J for IDCCG is set to 5. We choose the DCT matrix as the preconditioner.

The MSD and MSE performances of each algorithm is shown in Fig. 3. We can verify that, the IDMCG and IDCCG algorithm performs better than incremental LMS, while IDMCG is close to the RLS algorithm. With the preconditioning strategy, the performance of the IDCCG and IDMCG is further improved. The reason why the proposed IDMCG algorithm has a better performance than IDCCG is because IDMCG employs the negative gradient vector $\boldsymbol{g}_k$ with a recursive expression and the β_k is computed using the Polak–Ribiere approach, which results in more accurate estimates. Comparing with the IDCCG algorithm, the IDMCG is a non–reset and low complexity algorithm with one iteration per time instant. Since the frequency which the algorithm resets influences the performance, the IDMCG algorithm introduces the non–reset method together with the Polak– Ribiere approach which are used to improve the performance [17]. In detail, according to (26), when $\boldsymbol{g}_k(i) \approx \boldsymbol{g}_k(i-1)$, $\beta_k(i) \approx 0$. This will lead to $\boldsymbol{p}_k(i) \approx \boldsymbol{g}_k(i)$, which means the algorithm is reset.

2) Performance of Proposed Diffusion Distributed CG–Based Algorithms: The parameters of the performance test for each algorithm and the network are defined as follows: the step size μ for the LMS [28] algorithm is set to 0.2, the forgetting factor λ for the RLS [9] algorithm is set to 0.998. The λ_f for DDCCG and DDMCG are both 0.998. The η is equal to 0.45 for DDMCG. The iteration number J for DDCCG is set to 5. We choose the DCT matrix as the preconditioner.

For the diffusion strategy, the combining coefficients c_{kl} are calculated following the Metropolis rule. Fig. 4 shows the network structure. The results are illustrated in Fig. 5. We can see that, the proposed DDMCG and DDCCG have a better performance than the LMS algorithm and DDMCG is closer to the RLS's performance. The performance of the DDCCG and DDMCG can still benefit from the

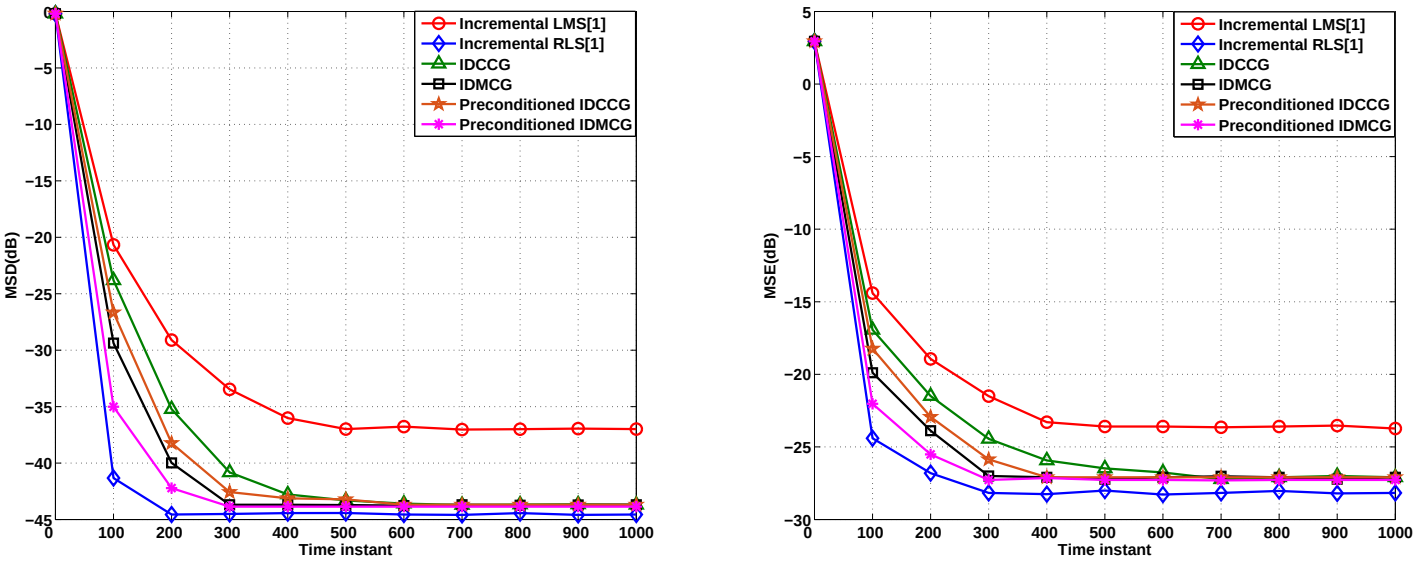

Fig. 3. MSD and MSE performance comparison for the incremental distributed strategies

preconditioning strategy.

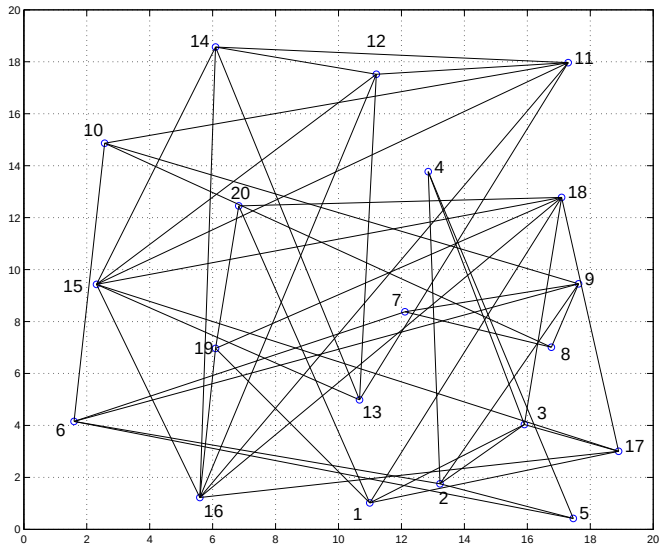

Fig. 4. Network structure

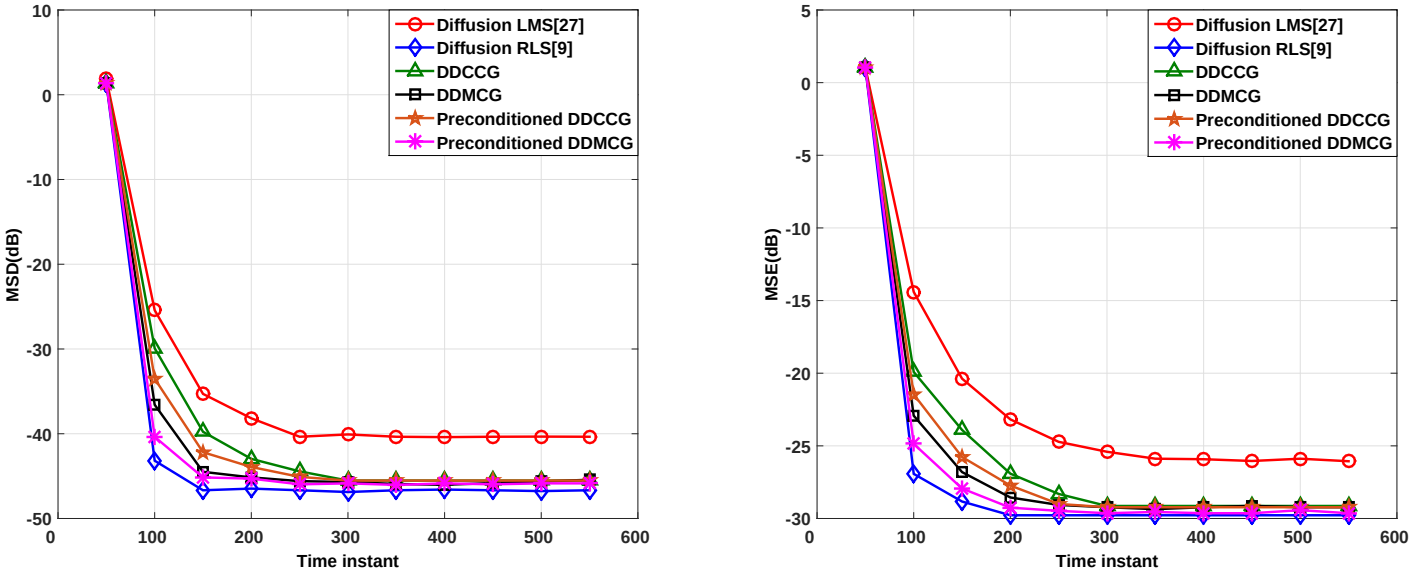

Fig. 5. MSD and MSE performance comparison for the diffusion distributed strategies

B. Distributed Spectrum Estimation

In this simulation, we consider a network composed of $N = 20$ nodes estimating the unknown spectrum $\boldsymbol{\omega}_0$, as illustrated in Fig. 4. The nodes scan $N_c = 100$ frequencies over the frequency axis, which is normalized between 0 and 1, and use $\mathcal{B} = 50$ non–overlapping rectangular basis functions to model the expansion of the spectrum [23]. The basis functions have amplitude equal to one. We assume that the unknown spectrum $\boldsymbol{\omega}_0$ is transmitted over 8 basis functions, thus leading to a sparsity ratio equal to 8/50. The power transmitted over each basis function is set equal to 0.7. The variance for the observation noise is 0.01.

For distributed estimation, we employ the DDMCG and the DDCCG algorithms, together with the preconditioned DDMCG algorithm to solve the cost function (11). The λ_f for DDCCG and DDMCG are both 0.99. The parameter η_f is set to 0.3 for DDMCG. The iteration number J for DDCCG is set to 5. The DCT matrix is employed as the preconditioner. We compare the proposed DDCCG and DDMCG algorithms with the sparse ATC diffusion algorithm [23], diffusion LMS algorithm [28] and diffusion RLS algorithm [9]. The step–sizes for the sparse ATC diffusion algorithm and diffusion LMS algorithm are set to 0.2, while for the sparse ATC diffusion algorithm, γ is set to 2.2×10^{-3} and β is set to 50. The forgetting factor λ for the diffusion RLS algorithm is set to 0.998.

We illustrate the result of distributed spectrum estimation carried out by different algorithms in the

terms of MSD performance, as shown in Fig. 6. We also consider the sparse ATC diffusion algorithm [23], diffusion LMS algorithm [28] and DDMCG to compare their performance in term of PSD in Fig. 7. The true transmitted spectrum is also depicted in Fig. 7.

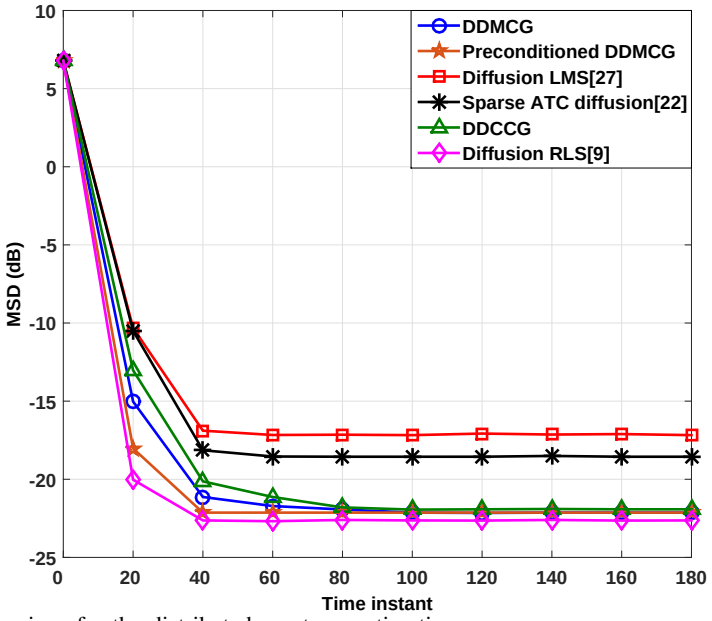

Fig. 6. Performance comparison for the distributed spectrum estimation

From Fig. 6, the DDMCG still performs better than other algorithms and is close to the diffusion RLS algorithm. From Fig. 7, we can notice that all the algorithms are able to identify the spectrum, but it is also clear that the DDMCG algorithm is able to strongly reduce the effect of the spurious terms.

VII. Conclusions

In this paper, we have proposed distributed CG algorithms for both incremental and diffusion adaptive strategies. We have investigated the proposed algorithms in distributed estimation for wireless sensor networks and distributed spectrum estimation. The CG–based strategies have a lower computational complexity than RLS–type algorithms, depending on the number of iterations that CG employs and a faster convergence than the LMS algorithm. The preconditioning strategy is also introduced to further improve the performance of the proposed algorithms. Simulation results have illustrated the advantages of the proposed IDCCG/IDMCG and DDCCG/DDMCG algorithms in different applications.

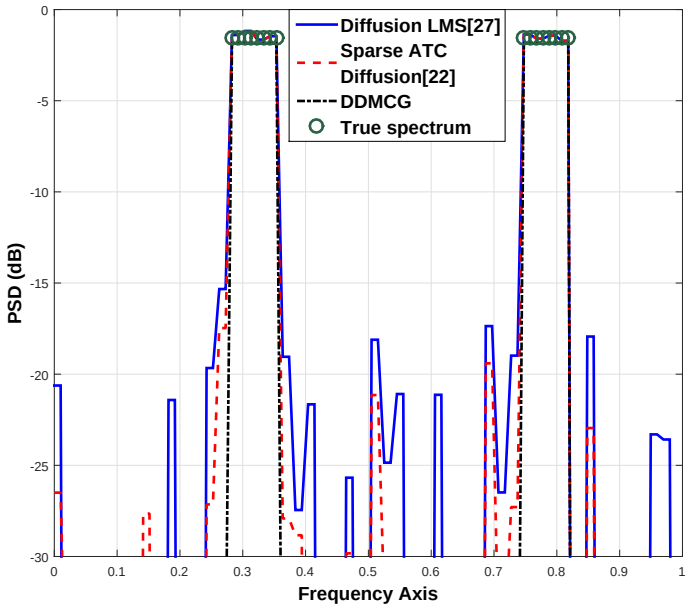

Fig. 7. Example of distributed spectrum estimation

REFERENCES

[1] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 223–229, Aug 2007.

[2] S. Xu, R. C. de Lamare, and H. V. Poor, "Adaptive link selection strategies for distributed estimation in diffusion wireless networks," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vancouver, Canada 2013.

[3] A. H. Sayed and C. G. Lopes, "Adaptive processing over distributed networks," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E90-A, no. 8, pp. 1504–1510, August 2007.

[4] A. Sayed and C. Lopes, "Distributed recursive least-squares strategies over adaptive networks," in *Fortieth Asilomar Conference on Signals, Systems and Computers*, October 2006, pp. 233–237.

[5] L. Li and J. A. Chambers, "Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology," in *Proc. IEEE Statist. Signal Process. Workshop*, Cardiff, Wales, September 2009, pp. 757–760.

[6] C. G. Lopes and A. H. Sayed, "Diffusion least–mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, July 2008.

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

[8] M. Abadi and Z. Saffari, "Distributed estimation over an adaptive diffusion network based on the family of affine projection algorithms," in *Sixth International Symposium on Telecommunications (IST)*, November 2012, pp. 607–611.

[9] F. Cattivelli, C. Lopes, and A. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[10] 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, no. 11, pp. 4583–4588, November 2009.

[11] S. Xu, R. C. de Lamare, and H. V. Poor, "Adaptive link selection algorithms for distributed estimation," *EURASIP Journal on Advances in Signal Processing*, vol. 1, no. 1, pp. 1–23, 2015.

[12] R. C. de Lamare and R. Sampaio-Neto, "Adaptive reduced-rank processing based on joint and iterative interpolation, decimation and filtering," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2503–2514, July 2009.

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

[14] L. Ismail and R. Barua, "Implementation and performance evaluation of a distributed conjugate gradient method in a cloud computing environment," *Softw. Pract.*, 2012.

[15] O. Axelsson, *Iterative Solution Methods*. New York, NY, USA: Cambridge University Press, 1995.

[16] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[17] P. S. Chang and J. A.N. Willson, "Analysis of conjugate gradient algorithms for adaptive filtering," *IEEE Trans. Signal Process.*, vol. 48, no. 2, pp. 409–418, February 2000.

[18] L. Wang and R. de Lamare, "Constrained adaptive filtering algorithms based on conjugate gradient techniques for beamforming," *IET Signal Processing*, vol. 4, no. 6, pp. 686–697, December 2010.

[19] A. Bertrand and M. Moonen, "Distributed adaptive node–specific signal estimation in fully connected sensor networks–part II: Simultaneous and asynchronous node updating," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5292–5306, 2010.

[20] J. Bazerque and G. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1847–1862, March 2010.

[21] S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM JOURNAL ON SCIENTIFIC COMPUTING*, vol. 20, pp. 33–61, 1998.

[22] Y. Zakharov, T. Tozer, and J. Adlard, "Polynomial spline–approximation of clarke's model," *IEEE Trans. Signal Process.*, vol. 52, no. 5, pp. 1198–1208, May 2004.

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

[24] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus–based in-network adaptive processing," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, June 2009.

[25] P. S. Chang and J. A.N. Willson, "Adaptive filtering using modified conjugate gradient," in *Proceedings., Proceedings of the 38th Midwest Symposium on Circuits and Systems*, vol. 1, August 1995, pp. 243–246.

[26] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester, U.K.: Wiley, 1987.

[27] D. F. Shanno, "Conjugate gradient methods with inexact searches," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 244–256, 1978.

[28] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, p. 10351048, March 2010.

[29] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over lms adaptive networks," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5107–5124, October 2012.

[30] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[31] S. C. Eisenstat, "Efficient implementation of a class of preconditioned conjugate gradient methods," *SIAM Journal on Scientific and Statistical Computing*, vol. 2, no. 1, pp. 1–4, 1981.

[32] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 23, no. 2, pp. 517–541, 2001.

[33] O. Axelsson and G. Lindskog, "On the rate of convergence of the preconditioned conjugate gradient method," *Numerische Mathematik*, vol. 48, no. 5, pp. 499–523, 1986.

[34] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: John Wiley&Sons, 2003.