Abstract—In this paper we apply Set-Membership (SM) adaptive algorithm over distributed networks based on incremental strategy. The distributed SM normalized least mean squares (dSM-NLMS) algorithm is introduced which has high convergence speed, low steady-state mean square error and low computational complexity features. The good performance of dSM-NLMS is demonstrated by several computer simulations in distributed networks.

Keywords-component; Distributed networks, adaptive algorithm, SM-NLMS, Incremental strategy

I. INTRODUCTION

Distributed processing deals with the extraction of information from data collected at nodes that are distributed over a geographic area. For example, each node in a network of nodes could collect noisy observations related to a certain parameter or phenomenon of interest. The nodes would then interact with their neighbors in a certain manner, as dictated by the network topology [1-2,8]. Network topology is the arrangement of the various elements (links, nodes, etc.) of a computer or biological network. Essentially, it is the topological structure of a network, and may be depicted physically or logically. Physical topology refers to the placement of the network's various components, including device location and cable installation, while logical topology shows how data flows within a network, regardless of its physical design. Distances between nodes, physical interconnections, transmission rates, and/or signal types may differ between two networks, yet their topologies may be identical. According to this explanation, there are three main topologies in distributed network:

- Incremental [2]
- Diffusion [2]
- Probabilistic diffusion [2]

Fig. 1 shows these topologies. In an incremental mode of cooperation, information flows in a sequential manner from one node to the adjacent node. This mode of operation requires a cyclic pattern of collaboration among the nodes, and it tends to require the least amount of communications and power [5].

An important structure in adaptive distributed networks is the kind of adaptive algorithm. The first adaptive algorithms which were applied in distributed network were least mean squares (LMS) and normalize least mean squares (NLMS). These algorithms have low computational complexity but have low convergence speed. The distributed affine projection algorithm (dAPA) and distributed recursive least squares (dRLS) have been proposed to solve this problem. Unfortunately, these algorithms have high computational complexity. In all mentioned algorithms, the weight coefficients are updated at every iteration. Therefore the main question is:

Do we need to update the weight coefficients at every iteration?

In single adaptive filter, the set-membership (SM) adaptive algorithm was proposed in which the weight coefficients are updated in some iteration. Therefore, the computational complexity will be reduced due to this kind of adaptation. The SM-NLMS in [3] has this feature. Also, SM-NLMS has high convergence speed and low steady-state mean square error (MSE) [6], [7].

In this paper we apply the approach of [3] in distributed networks to establish the distributed SM-NLMS (dSM-NLMS) algorithms. In dSM-NLMS, the weight coefficients in all nodes are updated in some iteration. The dSM-NLMS has the following important features:

- High convergence speed
- Low steady-state MSE
- Low computational complexity

![Fig. 1. Three modes of cooperation: (a) Incremental; (b) diffusion; and (c) probabilistic diffusion](image-url)
We have organized our paper as follows. In Section 2, the dNLMS will be briefly reviewed. The dSM-NLMS will be established in Section 3. Section 4 presents the computational complexity of introduced algorithms. In simulation results section, the performance of dSM-NLMS will be compared with dLMS and dNLMS algorithms. Throughout the paper the following notations are adopted:

<table>
<thead>
<tr>
<th></th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(.)'</td>
<td>Transpose of vector or matrix</td>
</tr>
<tr>
<td>(.)*</td>
<td>Complex conjugate</td>
</tr>
<tr>
<td>E{}</td>
<td>Expectation operator</td>
</tr>
</tbody>
</table>

II. DISTRIBUTED INCREMENTAL LEAST MEAN SQUARES (DLMS)

Consider a distributed network with \( N \) nodes over some local area. At every time instant \( i \), every node \( k \) obtains a scalar measurement \( d_k(i) \) and row regression vector \( u_{k,i} \) of size \( M \). The measurements \{ \( d_k(i), u_{k,i} \) \}, are assumed to be realization of zero-mean jointly wide-sense stationary random processes \{ \( d_k(i), u_{k,i} \) \}. The regressors have positive-definite covariance matrices \( R_{u,k} = E u_{k,i} u_{k,i}^* > 0 \). The objective for the network is to use the data to estimate the unknown vector \( w^o \) that minimizes the following global cost function [1]:

\[
J_{glob}(w) = E |d_k(i) - u_{k,i}w|^2
\]  

(1)

The optimal solution is given by the following equation [1],

\[
w^o = \left( \sum_{k=1}^{N} R_{u,k} \right)^{-1} \left( \sum_{k=1}^{N} d_k(i) \right)
\]  

(2)

Where \( R_{d,u,k} = E d_k(i) u_{k,i}^* \). The traditional steepest–decent technique can also be used to find the optimal solution iteratively as

\[
w_i = w_{i-1} - \mu [\nabla_w J_{glob}(w_{i-1})]^*
\]  

(3)

Where \( \mu > 0 \) is step-size parameter and \( w_i \) is an estimate for \( w^o \) at iteration \( i \). Moreover, \( \nabla_w J_{glob}(w) \) denotes the complex gradient of \( J_{glob}(w) \) with respect to \( w \) which is given by

\[
[\nabla_w J_{glob}(w_{i-1})]^* = \sum_{k=1}^{N} (R_{u,k} w - R_{d,u,k})
\]  

(4)

Substituting (4) into (3) leads to following update equation:

\[
w_i = w_{i-1} + \sum_{k=1}^{N} (d_{k}(i) - u_{k,i}w_{i-1})
\]  

(5)

Recursion (5) requires knowledge of the second-order moments \( \{ R_{d,u,k}, R_{u,k} \} \). An adaptive implementation can be obtained by replacing these second–order moment by local instantaneous approximation, say of the LMS type, as follows:

\[
R_{u,k} \approx u_{k,i}^* u_{k,i} \quad R_{d,u,k} \approx d_k(i) u_{k,i}^*
\]  

(6)

So the LMS-type recursion which is called distributed LMS (dLMS) can be stated as [2]:

\[
w_i = w_{i-1} + \mu \sum_{k=1}^{N} u_{k,i}^* (d_k(i) - u_{k,i}w_{i-1})
\]  

(7)

Based on (7) and using the data realization \{ \( d_k(i) \) \} and \{ \( u_{k,i} \) \} at time \( i \) an adaptive distributed incremental algorithm is established. For each time \( i \geq 0 \), the following adaptation is performed in each cycle [1]:

\[
k = 1, 2, ..., N
\]

\[
\begin{align*}
\varphi_0^{(i)} &= w_{i-1} \\
\varphi_k^{(i)} &= \varphi_{k-1}^{(i)} + \mu_k u_{k,i}^* (d_k(i) - u_{k,i}\varphi_{k-1}^{(i)}) \\
w_i^{(i)} &= \varphi_k^{(i)}
\end{align*}
\]  

(8)

Where \( \varphi_k^{(i)} \) is local estimate of \( w \) in node \( k \) at iteration \( i \).

III. DISTRIBUTED INCREMENTAL NORMALIZED LEAST MEAN SQUARES (DNLMS)

In previous section, we presented dLMS equation. If we use the normalized version of input data in (7), we can rewrite (7) and (8) as follows:

\[
w_i = w_{i-1} + \mu \sum_{k=1}^{N} u_{k,i}^* (d_k(i) - u_{k,i}w_{i-1}) \left\| u_{k,i} \right\|^2 + \varepsilon
\]  

(9)

For each time \( i \geq 0 \), repeat:

\[
k = 1, 2, ..., N
\]

\[
\begin{align*}
\varphi_0^{(i)} &= w_{i-1} \\
\varphi_k^{(i)} &= \varphi_{k-1}^{(i)} + \mu_k u_{k,i}^* (d_k(i) - u_{k,i}\varphi_{k-1}^{(i)}) \left\| u_{k,i} \right\|^2 + \varepsilon \\
w_i^{(i)} &= \varphi_k^{(i)}
\end{align*}
\]  

(10)

Equation (10) presents the distributed incremental normalized LMS (dNLMS).
IV. DISTRIBUTED INCREMENTAL SET-MEMBERSHIP NORMALIZED LEAST MEAN SQUARES (DSM-NLMS)

For highly correlated input signal in each node, dLMS and its normalized version (dNLMS) have low convergence speed [5]. In adaptive filter algorithms, there is trade-off between steady-state error and convergence speed. This means, if we want to have low steady-state error, we must paid a penalty which is low convergence speed. One of the best solutions for this problem is using the set-membership (SM) adaptive algorithm. This algorithm specifies an upper bound on the estimation error and reduces the computational complexity. The set-membership NLMS (SM-NLMS) algorithm in [3] has fast convergence, low steady-state MSE, and low computational complexity [3]. These good features were main reasons to use this class of adaptive algorithms in distributed networks. The SMNLMS adaptive filter update equation can be stated as [3]

\[
\begin{align*}
    w_i &= w_{i-1} + \alpha_i \frac{u_i^T}{\|u_i\|^2 + \varepsilon} e(i) \\
    &\text{where } e(i) = d(i) - u_i w_{i-1} \text{ and} \\
    \alpha_i &= \begin{cases} 
        1 - \frac{\gamma}{e(i)} & \text{if } |e(i)| > \gamma \\
        0 & \text{otherwise} 
    \end{cases} \\
    &\text{where } \gamma = \sqrt{5\delta^2_r}, \text{ and } \delta^2_r \text{ is variance of additive noise to input signal. If we apply this strategy over distributed network, we can establish DSM-NLMS as} \\
    w_i &= w_{i-1} + \alpha_k \sum_{k=1}^{N} \frac{u_k^T}{\|u_k\|^2 + \varepsilon} e_k(i) \\
    &\text{and} \\
    \alpha_k &= \begin{cases} 
        1 - \frac{\gamma_k}{e_k(i)} & \text{if } |e_k(i)| > \gamma_k \\
        0 & \text{otherwise} 
    \end{cases} \\
    &\text{where } \gamma_k = \sqrt{5\delta^2_{r,k}}, \text{ and } \delta^2_{r,k} \text{ is variance of additive noise to input signal at node } k. \text{ Table 1 summarizes the dSM-NLMS algorithm.}
\end{align*}
\]

V. COMPUTATIONAL COMPLEXITY

Table 2 compares the computational complexity of dLMS, dNLMS and dSM-NLMS algorithms. The dSMNLMS has the same multiplications with dNLMS in the worst case. But the adaptation of weight coefficients is depend on (14). If (14) is true, the adaptation is performed. We show in simulation results section that the adaptation occurs in some iteration and the computational complexity will be reduced significantly.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Multiplication</th>
<th>Division</th>
</tr>
</thead>
<tbody>
<tr>
<td>dLMS</td>
<td>3M+1</td>
<td>-</td>
</tr>
<tr>
<td>dNLMS</td>
<td>3M+1</td>
<td>1</td>
</tr>
<tr>
<td>dSM-NLMS</td>
<td>3M+1</td>
<td>1</td>
</tr>
</tbody>
</table>

VI. SIMULATION RESULTS

We justified the performance of dLMS, dNLMS, dSMNLMS, for an incremental network with \(N = 20\) in a system identification setup. The impulse response of the unknown system is \(w_p = \frac{1}{\sqrt{M}} \{1,1,1,\ldots,1,1\}\), where \(M\) is set to 16. The correlated elements of \(u_k(i)\) at each node are generated by passing a white Gaussian noise process with \(\delta^2_{r,k} \in [0,1]\), through a first order autoregressive model filter with the z-transfer function \(\frac{1-a_k^2}{1-akz^{-1}}\) and \(a_k \in [0,0.5]\). The noise sequence of each node is a white Gaussian process with variance \(\delta^2_{r,k} \in [0,0.1]\). Figs. 2-4 show the node profiles of \(\delta^2_{u,k}, \delta^2_{v,k},\) and \(a_k\). The MSE and MSD learning curves are obtained by ensemble averaging over 200 independent trials in node 5. Also, the steady-state MSE and MSD values are obtained by averaging over 1000 steady-state sample from 300 independent realizations. Figs. 5 and 6 present the MSE and MSD learning curves for dLMS, dNLMS, dSM-NLMS. As we can see, the dSM-NLMS has fast convergence speed. Also, the dSM-NLMS has lower steady-state MSE and MSD than dLMS and dNLMS algorithms. Furthermore, the computational complexity of dSM-NLMS is lower than dNLMS algorithm due to (14).
In Figures 7 and 8, the steady-state MSE and MSD of each node were presented. These figures can clearly show that the performance of dSM-NLMS is better than dLMS and dNLMS algorithms due to lower steady-state error values. Figure 9 illustrates the number of update for each node in dSM-NLMS algorithm. As we can see, the number of update in each node is very low. For example in node 4, only in 25 iteration, the adaptation will be performed. But in dLMS and dNLMS, for 1000 iteration, the weight coefficients will be updated.

VII. CONCLUSION

In this paper we presented the dSM-NLMS algorithm. In this algorithm, the weight coefficients were updated in some iteration. This algorithm had high convergence speed, low steady-state MSE and low computational complexity features. We demonstrated the good performance of this algorithm by several simulation results.
VIII. REFERENCES