EFFICIENT ADAPTIVE STRATEGIES OVER DISTRIBUTED NETWORKS

A Thesis
Submitted for partial fulfillment of

MASTER OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Spcl: COMMUNICATION AND SIGNAL PROCESSING

BY

KONDALA RAO JYOTHI

ROLL NO. #210EC4300

Department Of Electronics and communication engineering
National Institute of Technology
ROURKELA-769008

MAY 2012

EFFICIENT ADAPTIVE STRATEGIES OVER DISTRIBUTED NETWORKS

A Thesis
Submitted for partial fulfillment of

MASTER OF TECHNOLOGY
IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Spcl: COMMUNICATION AND SIGNAL PROCESSING

BY

KONDALA RAO JYOTHI

Roll No. #210EC4300

Under the guidance

Of

Prof. AJIT KUMAR SAHOO

Department Of Electronics and communication engineering
National Institute of Technology
ROURKELA-769008

MAY 2012

The Lord has given me

A wonderful family

To whom this book is dedicated.

To my parents

And to my brother and three

Sisters

ACKNOWLEDGMENTS

This dissertation would not have been possible without the guidance and the help of several individuals in way or another contributed and extended their valuable assistance in the course of this study.

My utmost gratitude to Prof. Ajit Kumar Sahoo, my dissertation adviser whose sincerity and encouragement I will never forget. Prof. Ajit has been my inspiration as I hurdle all the obstacles in the completion of this research work and has supported me throughout my project with patience and knowledge.

Sincere thanks to Prof. S. Meher, Prof. S. K. Patra, Prof. Samit Ari,Prof. S. K. Das, Prof. S. K. Behera, Prof.Poonam Singh and Prof.S.Hiremath for their constant cooperation and encouragement throughout the course. A special thanks to Prof. Upendra Kumar Sahoo for his valuable suggestions and help in Matlab coding.

I also extend my thanks to the entire faculty of Dept. of Electronics and Communication Engineering, National Institute of Technology Rourkela, Rourkela who have encouraged me throughout the course of Master's Degree.

I would like to thank all my friends, especially Chaitanya Kumar K, Samson Enosh, Chitra R, Nagaraju L, Brajesh Kumar, Pavan Kumar, Sridhar, Praveen Kumar, Raghuram, Anil Chako, Aravind, Dhanya VV and all my classmates for their help and emotional support during the course of work.

And finally thanks to God for his blessings with wisdom and to my parents and my brother for their faith, patience inspired me to walk upright in my life.

KONDALARAO J

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

CERTIFICATE

This is to certify that the thesis report entitled “**EFFICIENT ADAPTIVE STRATEGIE OVER DISTRIBUTED NETWORKS** “submitted by **Mr. KONDALARAO JYOTHI**, **Roll No: 210EC4300**, in partial fulfilment of the requirements for the award of Master of Technology degree in Electronics and Communication Engineering Department with specialization in “Communication and Signal Processing” at the National Institute of Technology, Rourkela and is an authentic work carried out by her under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Place**:** NIT Rourkela
Date:

Prof. Ajit kumar Sahoo
(Supervisor)
Dept. of Electronics & Communication Engg.
National Institute of Technology
Rourkela - 769008.

Abstract

Distributed wireless sensor networks finds many remote sensing applications like battle field surveillance, target localisation, environmental monitoring, precision agriculture, smart spaces and medical applications. Due to their vast range of applications efficient design and implementation become the current area of research.

A distributed network consists of certain number of processing elements called nodes. These nodes are distributed over a geographical area which collects the information for particular phenomena and communicates with other nodes of the network to arrive at estimation the parameter. A network needs effective and efficient designs to function properly with the limited available resources.

In this thesis, we review some of the computationally efficient adaptive distributed strategies developed using *incremental partial update techniques*. The schemes mentioned here solve the problem of linear estimation with less number of computations in a cooperative fashion. In a distributed network each node contains local computing equipment which estimates and shares them with other nodes. The resulting algorithms are less complex in competitions and in communication because of *Incremental partial update algorithms* and each node communicate with immediate node only. Computational complexity analysis is evaluated and performance characteristics of each algorithm are given with computer simulations. Simulation results show that with a small degradation in performance, a considerable amount of computational complexity is reduced.

Index terms- Adaptive strategies, distributed processing, Incremental LMS, sequential partial update LMS, stochastic partial update LMS, Max-partial update LMS, computational complexity.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

Wireless sensor network is comprised of a large number of small sensing self-powered sensor nodes distributed over a geographical area, which gather information or detect special events and communicate in a wireless fashion. Sensing, processing and communication are three key elements whose combination in one tiny device gives rise to a vast number of remote sensing applications, including environmental monitoring, precision agriculture, medical applications and battlefield surveillance. Due to their several popular applications, efficient design and implementation of wireless sensor networks have become an area of current research. Since the nodes in a network function with small and limited battery power and usually non-renewable resources, it is important to design the networks with less communication among the nodes to estimate the required parameter vector because communication and computation consumes most of the energy. However, recent advances in low power VLSI, embedded computing, communication hardware, and in general, the convergence of computing and communication are making this emerging technology a reality.

Each node in network collects noisy observations related to a certain desired parameter. In the centralized solution, every node in the network transmits its data to a central processor. This approach has the disadvantage of being non-robust to the failure of central processor and need a powerful processor. Again the central processor is lack of scalability, required a large number of communication resources. Alternatively each node in the network can estimate the parameter from the local observations and by cooperating with the neighbors.

So there is a need for distributed adaptive algorithms to reduce communication overhead for low-power consumptions, and low-latency system for real-time operation. Among them incremental algorithm is the majority choice.

1.1 Adaptive signal processing

In practice most systems are inherently time varying and or nonlinear. The signals associated with these systems often have time-varying characteristics. Adaptive signal processing that deals with the challenging problem of estimation and tracking of time varying systems. By virtue of its applicability to time varying and or nonlinear systems, adaptive signal processing finds application in a broad range of practical fields such as telecommunications, radar and sonar signal processing, biomedical engineering and entertainment systems. In order to make the estimation and tracking task tractable, the unknown system is usually modeled as time-varying linear system or in some cases as a finitely parameterized nonlinear system such as the volterra filter. This simplified system modeling is guided by prior knowledge of the system characteristics. An important objective of adaptive signal processing is to learn the unknown and possibly time-varying signal statistics in conjunction with system estimation.

The fundamental building block of an adaptive system is the adaptive filter. The objective of an adaptive filter is to learn an unknown system from observations of the system input or output signals utilizing any prior knowledge of the system and signal characteristics. The task of learning an unknown system is fundamental to many signal processing problems and comes in many disguises in the application of adaptive filters.

1.2 adaptive system identification

Most adaptive filtering problems are either a special case of adaptive system identification or utilize adaptive system identification as a means of solving another signal processing problem. In this sense, adaptive system identification provides the basis for a range of signal processing applications. It is therefore, essential that we have a good understanding of the underlying principles of and assumptions relating to adaptive system identification.

As depicted in figure1.1, in adaptive system identification, the objective is to estimate an unknown system from its and output observations given by $x(k)$ and $d(k)$, respectively. A model for the adaptive filter is chosen based on prior knowledge of the unknown system characteristics, as well as the complexity considerations.

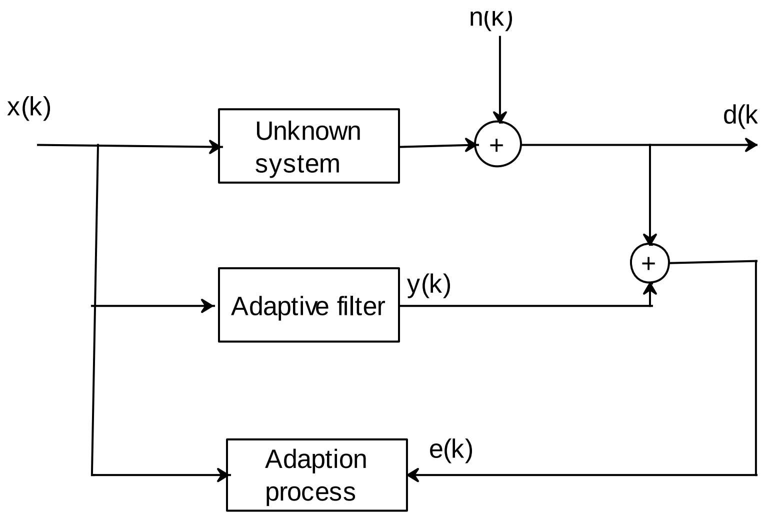

Figure 1.1 Adaptive system identification

In this and most preferred form, the adaptive filter is a finite impulse response filter of length N with the adjustable impulse response coefficients.

$$w(k) = [w_1(k), w_2(k) \ldots\ldots\ldots\ldots\ldots w_N(k)]^T \qquad \rightarrow (1.1)$$

Here T denotes the transpose operator. Equation (1) is perhaps the most widely used Adaptation filter model mainly because of its applicability to a wide range of practical problems. IN a system identification context, the adaptive filter attempt to learn the unknown system By using a model of the unknown system represented by $w(k)$. The difference between the noisy response of the unknown system (the desired response $d(k)$) and the response of the adaptive filter $y(k)$ is called the error signal $e(k)$.

$$e(k) = d(k) - yk) \qquad \rightarrow (1.2)$$

At each iteration k the adaptive filter updates coefficients in order to minimize the appropriate norm of the error signal $e(k)$. When the error norm is minimized in a statistical sense, the corresponding $w(k)$ gives an estimate of the unknown system parameters. If the unknown system is time varying, i.e. its parameters change with time, the adaptive filter can track these changes by updating its coefficients in accordance with error signal. It can take several iterations for the adoption process to converge. The time taken by the adaption process to converge provides the indication of the convergence rate.

There are two main tasks performed by the adaptive filter; viz. adaption process and filtering process. In figure 1.1 these processes are identified by the adoption process and adaptive filter block. For linear adaptive filters given by equation (1.1), the filtering process involves convolution. If the number of filter coefficients is large, the convolution operation may prove to be computationally expensive. Reduced complexity convolution techniques based on fast Fourier transform (FFT), such as overlap-add and overlap-save may use to ease computational demand. The adoption process has also become computationally expensive for long adaptive algorithms due to the arithmetic operations required to update the adaptive filter coefficients. The computational complexity of the adoption process depends on the adoption algorithm employed.

Prediction of the random signals and noise cancellation are two special cases of adaptive system identification. Figure 1.2(a) shows a one-step predictor which estimates the present value of the random signal $x(k)$ based on the past values $x(k-1),........x(k-N)$. if $x(k)$ is a stable autoregressive process of order N :

$$x(k) = a_1 x(k-1) + a_2 x(k-2) + ... + a_N x(k-N) + v(k) \qquad \rightarrow (1.3)$$

Where $v(k)$ is white noise, the adaptive filter $w(k)$ in figure 1.2(a) estimate the auto regressor coefficients $[a_1, a_2, a_3.........a_N]^T$. After convergence the prediction error $e(k)$ is equal to $v(k)$, which implies whitening of the colored noise signal $x(k)$. Referring to figure 1.1, we observe that the adaptive system identification setup can be converted to a one step predictor by replacing the unknown system with a direct connection (short-circuiting the unknown system), setting $n(k) = 0$ and inserting a one-sample delay z^{-1} at the input of the adaptive filter. Swapping $x(k) and\, n(k)$ in figure 1.1 and referring to $x(k)$ as the signal of interest and $n(k)$ as the interfering noise changes the system identification problem to a noise cancellation problem with $e(k)$ giving the cleaned signal (see figure 1.2(b)). The noise signal $n(k)$ is the reference signal and the unknown system represents any filtering that $n(k)$ may undergo before interfering with the signal of interest $x(k)$. The sum of $x(k)$ and filtered $n(k)$ is the primary signal. The unknown

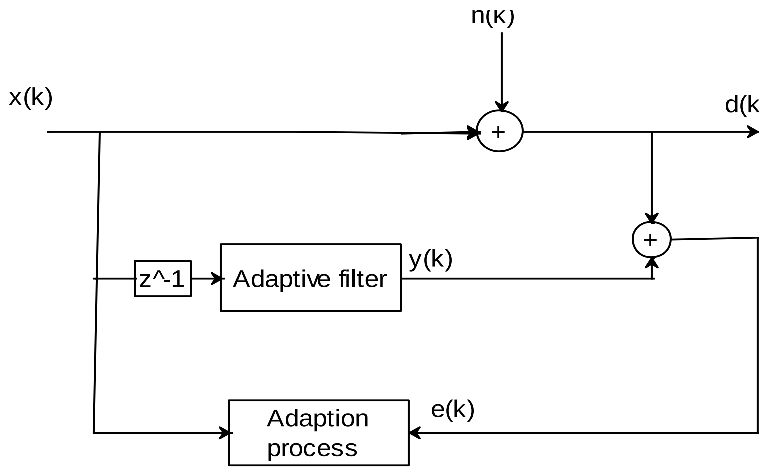

Figure 1.2 One step prediction of a random signal by an adaptive filter which identifies the auto regressive model of the random signal with y (k) giving the prediction output and e(k) the prediction error.

is identified by an adaptive filter. Subtracting the adaptive filter output from the reference signal give the error signal. Minimization of the error norm implies a minimization of the difference between the adaptive filter output and the filtered reference signal.

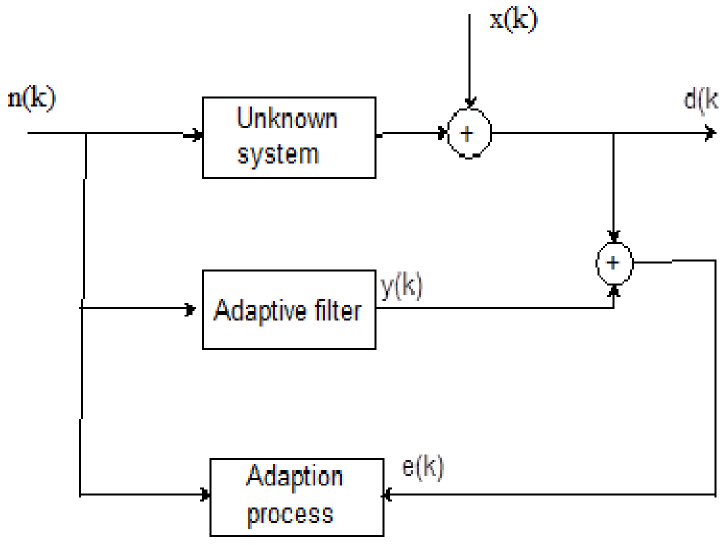

Figure 1.3 adaptive noise cancellation where the adaptive filter estimates the unknown system filtering the input noise signal n(k) with x(k) denoting the signal of interest and e(k) the cleaned signal.

If the adaptive filter provides a perfect estimate of the unknown system, then the error signal becomes identical to the signal of interest. This provides the perfect noise removal.

1.3 Motivation

The motivating factor behind employing distributed wireless sensor networks is the availability of low-power micro-sensors, actuators, embedded processors, and radios. Most of the sensor networks consists relatively small numbers of sensors, wired to a central processing unit which performs all of the signal processing. But here we focus on distributed wireless sensor networks in which sensing and signal processing is also distributed.

When the exact location of a signal of interest is unknown over a geographical area we go for distributed sensing. In distributed sensing certain number of sensors are placed closer to the phenomena being monitored instead of using a single sensor. We can improve the signal quality as the signal to noise ratio improves and improved opportunities for line of sight communication. Thus, distributed sensing provides robustness to environmental obstacles.

Though wired networking of distributed sensors has its own advantages like ease connectivity, simple system design and simplified operation when the nodes wired to renewable energy sources, we go for wireless because, in many existing and future applications, the geographic area being monitored will not infrastructure readily for communication or energy. Remote nodes must depend on local, finet, and small energy sources along with wireless communication channels.

The central processor limits the network performance as the nodes has to communicate back and forth for conveying the information which consumes most of the energy and channel bandwidth and is not true distributed process. So we go for distributed process in which each local node has its own computing system and communicates independently with other nodes of the network.

Scientists and environmentalists needs to monitor soil and chemical content of air, as well as populations of animal species and plants and their density over a particular geographical area, for these type of monitoring process the primary methods are imaging and acoustics to localize, identify and track species or phenomena based on implicit signals, or explicit signals. These facilities must be deployed in remote places which lack installed energy and communication infrastructures; this is the motion for the need for low-power wireless sensor networks.

The strategies for node cooperation have significant effect in terms of energy consumption and communication bandwidth utilization. For example in the task of bird species identification, we need several cameras. For the task to be accomplished through image analysis, we have to stream all the video and pictures back to a human operator which is a costly approach. In an alternate method, we could stream audio to a central location where central processor performs signal processing to detect and stream back only those streams that are most identical to contain a target species. This method reduces the communications complexity greatly, but it still suffers from communications latency and lacks scalability as the stream audio has to pass through a central processing point. Finally, we go for distributed solution, enabling the local node to process audio signal itself, and developing distributed algorithms that require only local cooperation to make a decision to capture images. This approach needs no long-range streaming of audio or video is necessary, resulting in scalability and more efficient use of communications bandwidth and limited energy resources.

Chapter 2

Design considerations for Incremental adaptive strategies

2.1 Introduction

A wireless sensor network consists of a certain number of sensor nodes distributed over a geographical area. These sensor nodes are self-powered and some cases consist of a local computing mechanism. Nodes of the network collect the information or detects the special events and communicates in with other nodes or with a central processor in a wireless fashion so the name wireless sensor networks. In a distributed processing each node interacts with other nodes in the network to arrive at an estimate of the particular parameter. Nodes in the network communicate in a certain manner as dictated by the topology of the network. Adaptive strategies with incremental mode of communication described in [1] focus on reduction in communication among the nodes by restricting a particular node receiving and transmitting to the immediate nodes only instead of every node of the network.

Consider a network consisting of nodes observing temporal data for a particular phenomenon from different spatial sources with different statistical profiles. The objective is to enable the nodes to arrive at an estimate of the parameter of interest from observed data. For estimating the vector of parameter of interest there are two general approaches, in first approach called centralized approach the data or local estimates from all the nodes of the network are passed to a central processor where the individual estimates from nodes are combined and vector of parameter is estimated.

- This approach requires sufficient communication to transmit data to and fro between the central processor and nodes.
- This approach is not a truly distributed system and it limits the autonomy of the network and failure of the network due to the presence of the central node at some critical points.

- The tracking performance of the network is reduced because the network fails to adapt to the real-time environmental changes.

In the second approach we eliminate the central processor and enable each node in the network to have local computing abilities. Each node acts like an adaptive filter and estimate the parameter of interest from local observations. The individual estimates at each node then conveyed to the neighboring nodes where they combine the estimates of the network in order to arrive at an estimate of the parameter influenced by the data of the neighboring nodes. The consensus strategy gives this type distributed estimation which will be studied later in this chapter.

2.2 Applications

Let us consider a network spread over a geographical area with L nodes as shown in the fig3.1. Each node calculates local temperature t_i. The objective is to provide each node of the network with the information about the average temperature t_{avg}.

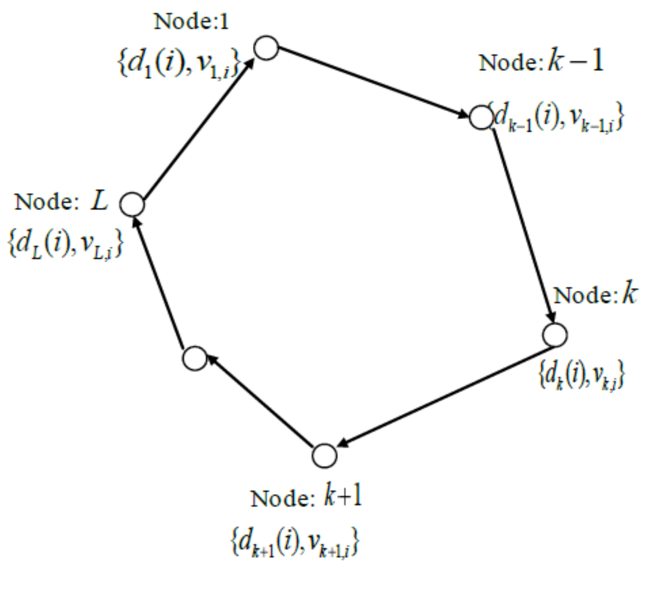

Figure 2..1 Distributed network accessing temperature data with L nodes

For providing each node with the average temperature we can use consensus implementation in which each node fuses the measurements from its neighbors. The result of the combination is the node's new measurement.

$$y_1(i) \leftarrow \alpha_1 y_1(i-1) + \alpha_L y_L(i-1) + \alpha_{k-1} y_{k-1}(i-1) \quad \text{For node 1}$$

Where y_i denotes the new measurement of the node1 at iteration i, and α's the coefficients chosen appropriately. Every node of the network computes new measurement same as the operation performed at node1 and the process is repeated. For suitable coefficients (α) and network topology all node measurements will converge to the desired average temperature t_{avg}.

Other applications distributed processing are tracking and monitoring of a moving target over a geographical area. The sensors in the network communicate with each other to detect the presence of the target. Physiological monitoring, distributed networks liking PCs, environmental monitoring, smart spaces, military, precision agriculture, target localization.

2.3 Modes of cooperation

Mode of cooperation the manner with which the nodes of the distributed network communicate with the neighboring nodes. Incremental mode, diffusion mode and probabilistic diffusion mode are the three modes of cooperation generally used in the distributed network. In this section we are going to illustrate these cooperation modes.

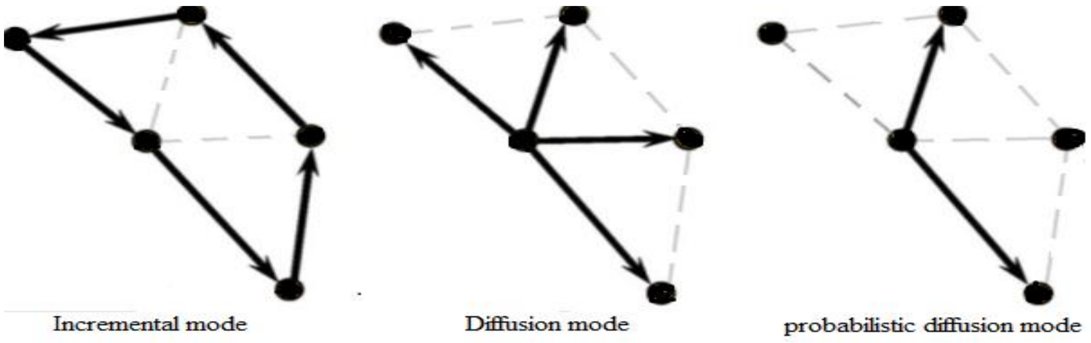

Figure2.2 modes of cooperation in distributed networks

In incremental mode of cooperation, information from one node to adjacent node flows in a sequential manner. A cyclic pattern of collaboration among the nodes is required for this mode of cooperation. A single node in the network communicates with adjacent neighboring nodes only instead of communicating with all the nodes of the network. So this mode of cooperation requires the least amount of communication and power. Each node exchanges its estimation with adjacent nodes then they are fused and given to local adaptive filter, so the estimate at each node is a combined effect of both its temporal data as well as the spatial data across the neighbors.

In diffusion mode of cooperation, each node communicates with all its neighbors depending on the network topology. The amount of communication is higher compared to the incremental mode. At each node, estimates are exchanged with neighboring nodes and combined and then fed to the local adaptive filter.

The probabilistic diffusion mode is modified version of the diffusion mode to reduce the communication complexity. A particular node is allowed to communicate with a subset of its neighbors. We can choose the randomized subset of nodes to be communicated by some performance criterion.

2.4 distributed consensus

We can find distributed averaging in many ways .Generally used and straightforward method is flooding. In this method each node contains a table of the initial node values of all the nodes in the network, initialized with their own nodes value only. For each step or iteration the nodes exchange information from tables of their neighbours and their own. After several iterations, every node comes to know all the initial values of all the nodes, so the average can be computed. But flooding is some complex and needs to higher amount of communication.

Consensus strategies solve the problem of finding linear iteration that produces the average estimation over a network. In other words it asymptotically computes the average of some initial values given at the nodes. The temperature example of fig 3.1 is a special

case of a distributed processing, known as consensus. Distributed consensus implementations employ two time scales broadly and they function as follows. Assume the network is estimating a parameter of interest. Nodes collect noisy observations over a period of time from the geographical area and then reach an individual decision about the parameter. During this time, nodes act like individual agents as there is limited interaction among the nodes. Following this initial stage, the nodes then fuse their estimates through several consensus iterations, under suitable conditions, the estimates generally converge asymptotically to the global or desired estimate of the parameter.
We will illustrate another example of a distributed consensus implementation, which can serve as motivating contributions for this work. Consider a network of nodes. Each node has access to a data vector x_k and a data matrix H_k. The x_k are noisy and distorted measurements of some unknown vector w^0, follows:

$$x_k = H_k w^0 + u_k$$

Each node can evaluate the least-squares estimate of w^0 based on its own local data $\{x_{k,} H_k\}$. To do so, each node evaluates its local cross-correlation vector $\varphi_k = H_k{}^* x_k$ and its autocorrelation matrix $R_k = H_k^* H_k$. Then, local estimate of w^0 can be found from $\hat{w} = R_k^{-1}\varphi_k$. This operation requires that each node collects sufficient data into x_k and H_k. Once the local quantities $\{\varphi_k, R_k\}$ have been evaluated at the individual nodes, one can apply consensus iterations at the nodes to determine $\hat{\varphi}_k$ and $\hat{R}_k$, which are estimates of the overall average quantities and defined, as follows:

$$R = \frac{1}{L}\sum_{k=1}^{L} R_k \text{ and } \varphi = \frac{1}{L}\sum_{k=1}^{L} \varphi_k$$

A global estimate of w^0 is given by $\hat{w} = \hat{R}^{-1}\hat{\varphi}$. For all practical considerations, least-squares implementation of this manner is an offline or non-recursive solution. For example, if a particular node collects an extra entry in x_k and an extra row in H_k, a difficulty that occurs is how to update the current solution to account for new data without having to repeat prior processing and afresh iterations. In addition to that, the offline averaging limits the ability of consensus-based solutions to track fast-changing environments, especially in networks with limited communication and power resources.

2.5 Distributed least-mean-squares (LMS) algorithm

To solve the distributed estimation that meets real-time operation, adaptive implementations, and low computational and communications complexity, we propose distributed least-mean-squares (LMS)-like algorithm that requires low computational and communication complexity and inherits the robustness of LMS implementations. The proposed method promptly adapts to data changes in the environment, as the information flows through the entire network. This algorithm requires neither intermediate averaging nor two separate time scales as in consensus implementations. This distributed adaptive solution is an extension of adaptive filters and can be implemented without requiring any prior knowledge of data statistics in other words, it is not model dependent. we focus in this chapter on LMS-type updates for simplicity and the same ideas and techniques will be applied to other types of adaptation rules.

We design and develop distributed algorithms that enable a network of nodes to function as an adaptive entity in its own right. As the regular adaptive filter has the ability to responds in real time to its data and to variations in the statistical properties of this data, the same can be extended to the network domain. Specifically, the purpose of this chapter is as follows:

- Based on the extensive works on distributed optimization we can develop a family of *incremental adaptive algorithms* for distributed estimation.
- To use the developed incremental algorithms to meet the addressed adaptive network structures composed of an interconnected set of nodes that are able to respond to data in real time and to track variations in the statistical properties of the data as follows:

  i) When a node receives a new piece of information from environment, then the node updates its local estimate of the parameter of interest with this new information.

ii) Node then shares the local estimates of the parameter with its immediate neighbours in a process that allows the information to flow to the other nodes in the network.

iii) To analyse the performance of the resulting interconnected network of nodes. This task is challenging since an adaptive network comprises a “system of systems” that processes data cooperatively in both time and space. Different nodes will converge to different mean-square-error (MSE) levels, reflecting the statistical diversity of the data and the different noise levels.

Finally, we considered all the factors that tend to design an incremental adaptive algorithm over ring topologies and we derive closed form expressions for its mean-square performance in the subsequent chapters.

Chapter 3

Distributed estimation using the Incremental LMS algorithm

There have been extensive works in the literature on incremental methods for solving distributed optimization problems. It is known that whenever a cost function can be decoupled into a sum of individual cost functions, a distributed algorithm can be developed for minimizing the cost function through an incremental procedure. We explain the procedure as follows in the context of MSE estimation.

Consider a network with L nodes. Each node k has access to time realizations $\{d_k(i), v_{k,i}\}$ of zero-mean spatial data $\{d_{k,}v_k\}, k = 1,2......L$, where each d_k is a scalar measurement and each v_k is a row regression vector. We collect the regression and measurement data into two global matrices, as follows:

$$U = \begin{bmatrix} v_1(1) & v_1(2) & . & . & v_1(M) \\ v_2(1) & v_2(2) & . & . & v_2(M) \\ . & . & . & . & . \\ . & . & . & . & . \\ v_L(1) & v_L(2) & . & . & v_L(M) \end{bmatrix} \quad (L \times M) \rightarrow (3.1)$$

$$\mathbf{d}\text{=}\begin{bmatrix} d_1 \\ d_2 \\ . \\ . \\ d_L \end{bmatrix} \quad (L \times 1) \rightarrow (3.2)$$

these quantities collect the data across all L nodes. The objective is to estimate the $M \times 1$ vector w that solves

$$\min_w J(w) \rightarrow (3.3)$$

Where the cost function $J(w)$ denotes the mean square error (MSE) as fallows

$$J(w) = E\|d - Uw\|^2 \rightarrow (3.4)$$

And E is the expectation operator. The optimal solution w^0 of (3.3) satisfies the orthogonality condition.

$$EU^*(d - Uw) = 0 \rightarrow (3.5)$$

So that w^0 is the solution to the normal equations

$$w^0 = R_v^{-1} P_v \rightarrow (3.6)$$

Which are defined in terms of the correlation and cross-correlation

$$P_V = EU^*U \quad (M \times M) \qquad R_v = EU^*d \qquad \rightarrow (3.7)$$

If the optimal solution w^0 were to be computed from (3.6), then *every node* in the network would need to have access to the global statistical information $\{P_v, R_v\}$. Alternatively, the solution could be computed centrally and the result broadcast to all nodes. Either way, these approaches drain considerable communications or computational resources or they do not endow the network with the necessary adaptively to cope with possible changes in the statistical properties of the data. We shall instead develop and study a *distributed* solution that allows cooperation among the nodes through limited local communications, while at the same time equipping the network with an distributed incremental adaptive mechanism. Specifically, in this chapter we focus on the incremental mode of cooperation, where the estimation task is distributed among the nodes and each node is allowed to cooperate only with one of its direct neighbours at a time. The single-neighbour case is already challenging in its own right, and the analysis will bring forth several interesting observations. The same mechanism can be extended to other modes of cooperation.

3.1 Incremental adaptive solution

Though there some other techniques like steepest-descent solutions they relies on the second order moments like $R_{v,k}$ and $P_{v,k}$ which are needed to evaluate the local gradients. We can replace these second-order moments $\{R_{v,k}, P_{v,k}\}$ by instantaneous approximations, say of the LMS type, as follows

$$R_{v,k} \approx v_{k,i}^* v_{k,i} \text{ and } P_{v,k} = d_k(i) v_{k,i}^* \rightarrow (8)$$

by using data realizations $\{d_k(i), v_{k,i}\}$ at time i. The approximations (8) lead to an adaptive distributed incremental algorithm, or simply a *distributed incremental LMS* algorithm of the following form:

For each time $i \geq 0$
$k = 1, \ldots\ldots. n$
$\phi_0^{(i)} = w_{i-1}$
$\phi_k^{(i)} = \phi_{k-1}^{(i)} + \mu_k v_{k,i}^* (d_k(i) - v_{k,i}\phi_{k-1}^{(i)})$
$w_i = \phi_n^{(i)} \qquad \rightarrow (9)$

Where

μ_k =step size parameter at node k

$\varphi_k^{(i)}$ = local estimate at node k at time i

w_i =estimate of w^o at node k

$v_{k,i}$ = input at node k at i^{th} iteration.

$\varphi_{k-1}^{(i)}$ = local estimate of immediate node $k-1$

$v_{k,i}^*$ is the hermitian of $v_{k,i}$. The above mentioned algorithm uses local data realizations $d_k(i), v_{k,i}$ and $\varphi_{k-1}^{(i)}$ weight estimate of immediate node. This incremental procedure purely relies on the local data estimation and gives truly distributed solution.The operation of algorithm (9) is illustrated in Fig. 5. At each time instant, each node uses local data realizations $\{d_k(i), v_{k,i}\}$ and the weight estimate received from its adjacent node to perform the following three tasks:

1) Evaluate a local error quantity: $e_k(i) = d_k(i) - v_{k,i}\phi_{k-1}^{(i)}$;

2) update its weight estimate: $\phi_k^{(i)} = \phi_{k-1}^{(i)} + \mu_k v_{k,i}^* e_k(i)$;

3) pass the updated weight estimate to its neighbour node $k+1$.

This distributed incremental adaptive implementation generally has better steady-state performance and convergence rate.The subsequence chapter shows the simulation results and performance analysis of the incremental LMS algorithm for distributed estimation.

3.2 simulations and performance analysis

An important question now is, how well does the adaptive incremental solution perform That is, how close does each $\phi_k^{(i)}$ (local estimate at node) get to the desired solution w^0 as time evolves? Studying the performance of such an interconnected network of nodes is challenging (more so than studying the performance of a single LMS filter) for the following reasons:

1) Each node k is influenced by local data with local statistics $\{R_{v,k}, P_{v,k}\}$ (spatial information);

2) Each node k is influenced by its neighbours through the incremental mode of cooperation (spatial interaction);

3) Each node is subject to local noise with variance $\sigma_{v,k}^2$ (spatial noise profile).

In the next section, we provide a framework for studying the performance of such network by examining the flow of energy through the network both in time and space. For instance, we shall derive expressions that measure for each node the steady-state values and as it will be shown that despite the quite simple cooperation strategy adopted, in steady-state each individual node is affected by the whole network, with some emphasis given to local statistics. Furthermore, as the step size is decreased asymptotically, both quantities mean-square deviation (MSD) and EMSE approach zero for every node in the network, which also drives the MSE for every node asymptotically to the background noise level. In order to pursue the performance analysis, we shall rely on the energy conservation approach. This energy-based approach needs to be extended to account for the space dimension because the distributed adaptive algorithm of (9) involves both a time variable and a space variable. Moreover, we need to deal with the

Energy flow across interconnected filters and since each node in the network can now stabilize at an individual MSE value, some of the simplifications that were possible in the single node case cannot be applied here. For example, due to cooperation, the performance of every node ends up depending on the whole network, an effect which we shall capture by a set of coupled equations. In order to evaluate individual node performance, weighting will be used to decouple the equations and to evaluate the quantities of interest in steady state. The main result of this section is Theorem 1, further ahead, which provides closed-form expressions for the performance of each node in the network in terms of the so-called MSE, MSD, and EMSE measures defined next.

Here we give the simulation results comparing the each technique. Number nodes in network *L*= 20.The regressor vector or data vector $v_{k,i}$ is $1\times M$ and collects the data as follows.

$$v_{k,i} = col\{v_k(i), v_k(i-1), ..., v_k(i-M+1)\} -\!\!\rightarrow (5)$$

In the network each node k depends on local statistics and influenced by immediate neighbors. 300 independent experiments were performed and averaged. In all the experiments step-size parameter is chosen to be small and kept constant. The curves are generated for 100 iterations. Here Mean Square Error (MSE) is taken as the performance metric. MSE gives how far the local estimate from the optimum weight w^0. The performances of the proposed algorithms are compared with that of incremental algorithm.

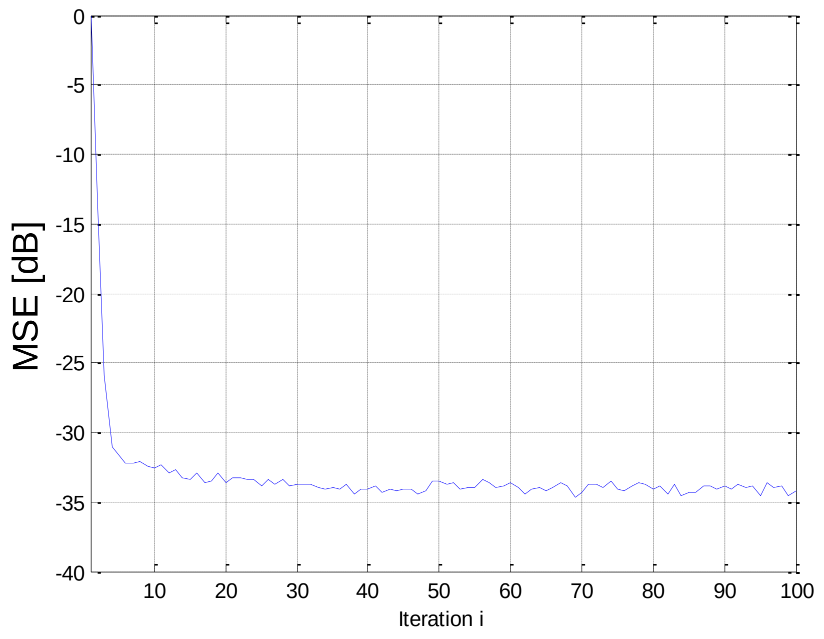

Figure 3.1 mean-square-errors (MSE) for the Incremental LMS algorithm

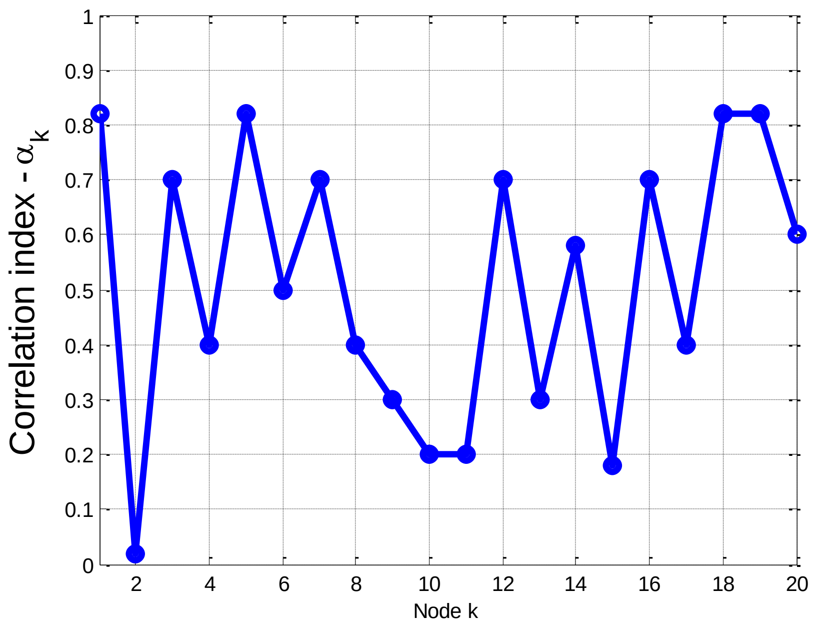

Figure 3.2 correlation index of the nodes

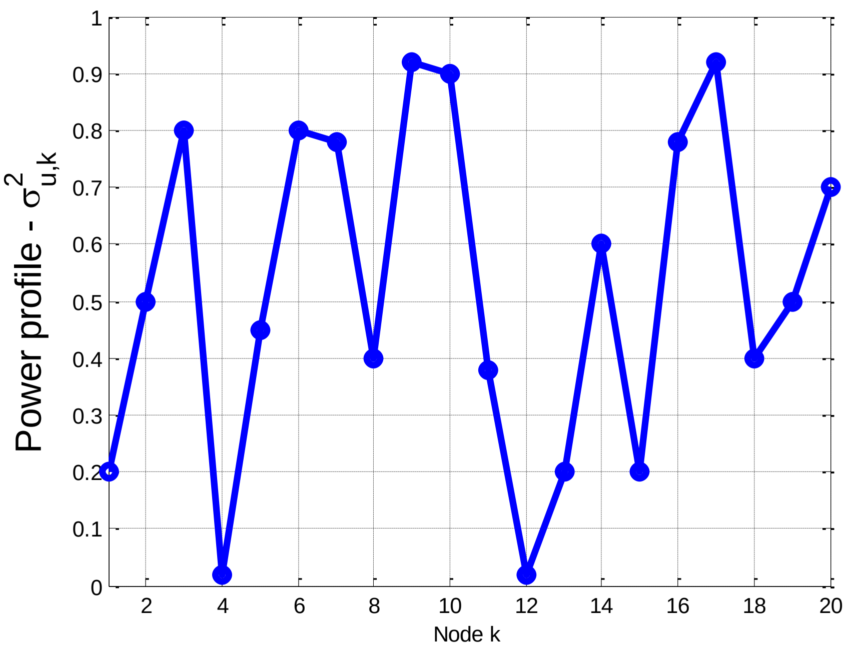

Figure 3.3 power profile of the nodes

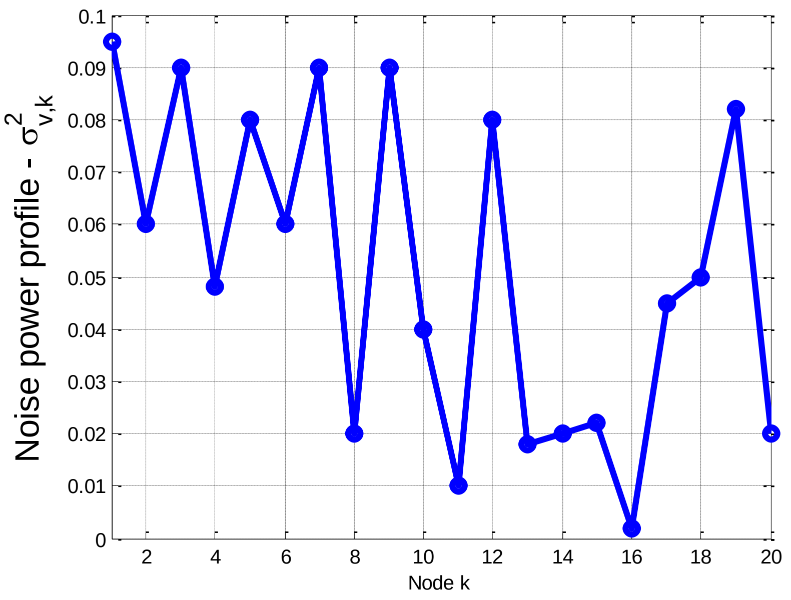

Figure 3.4 noise power profile of the nodes

Chapter 4

Incremental Partial update LMS Techniques

Partial update techniques are developed to reduce the computational complexity. Complexity depends on the number of hardware multipliers required for the computing system and decides the power consumption. Bandwidth plays an important role in communication complexity. Even though the Incremental adaptive solution mentioned in the previous chapter reduces the amount of communications to considerable amount the numbers of calculations in each iteration are equal to the LMS. We can reduce computational complexity by partial update techniques. In some applications adaptive filters have a large number of coefficients. Updating the entire coefficient vector is costly in terms of memory, computations and power consumption. Generally more number of hardware multipliers imply to more power. Here we propose incremental partial update techniques which reduce computational complexity to a considerable amount.

These Partial Update Incremental Partial update Techniques reduce both computation complexity and communication complexity to a significant amount. The techniques mentioned in this chapter have the following advantages

- Communication complexity is reduced i.e. less bandwidth is sufficient for the communication as Nodes in the network communicates with the immediate neighbors only.
- Computational complexity is reduced as number of hardware multipliers required is less compared to the usual LMS techniques.
- Physical complexity is reduced as the technique suits for the Low-energy sources and the need for the central processor is eliminated through the Incremental techniques in which nodes have local computational capabilities.

The complexity constraint posed by scarce resources necessitates the development of resource allocation schemes that assign available resources for adaption processes on the basis of a well-defined allocation or selection criterion. In general, the selection criterion must reflect the merit of a given resource assignment in terms of performance measure is the convergence rate for a given steady-state error level. The computational complexity associated with the adaption process can be reduced when an adaptive filter has M coefficients while the adaption process can update only N coefficients so as to reduce the computational complexity.

In the following chapter we review some of incremental partial update LMS algorithms like Incremental-sequential, incremental-stochastic and incremental-Max-partial update algorithms. In the chapter wherever partial update technique is mentioned it is the Incremental partial update technique until unless it is mentioned.

4.1 Sequential Partial update Incremental LMS

This method updates a subset of the adaptive filter coefficients so as to reduce the computational complexity associated with the adoption process at each iteration for every node in the network. In this sense sequential partial update results in decimation of the adaptive filter coefficient vector. Coefficient subset to be updated is selected in a deterministic fashion.

The update equation is given by

$$\phi_k^{(i)} = \phi_{k-1}^{(i)} + \mu_k \boldsymbol{I}_{N,k}^{(i)} e_k^{(i)} v_{k,i}^{*} \rightarrow \qquad (4.1)$$

Where

$$e_k^{(i)} = d_k(i) - v_{k,i}\varphi_{k-1}^{(i)}$$

$$I_{N,k}^{(i)} = \begin{bmatrix} a_1(i) & 0 & . & . & 0 \\ 0 & a_2(i) & . & . & . \\ . & . & a_3(i) & . & . \\ . & . & . & . & . \\ 0 & . & . & . & a_M(i) \end{bmatrix} \rightarrow (4.2)$$

$$\sum_{j=1}^{M} a_j(i) = N \qquad , \; a_j(i) \in \{0,1\}$$

$I_{N,k}^{(i)}$ is the coefficient selection matrix to select a subset of N coefficients out of M total coefficients at node k at i^{th} iteration.

Let the coefficient index set be $Q = \{1, 2 \ldots M\}$ i.e. there are M coefficients totally out of which N coefficients are to be updated. Then Q is divided into S number of subsets $P_{1,} P_{2,} \ldots . P_{s,}$ with each subset having N coefficients where $S = {}^{M}c_{N}$. Let $R = M / N$ be an integer then R coefficient subsets are arranged in periodic sequences with respective coefficient selection matrix $I_{N,k}^{i}$.

$$I_{N,k}^{(i)} = \begin{bmatrix} a_1(i) & 0 & . & . & 0 \\ 0 & a_2(i) & . & . & . \\ . & . & a_3(i) & . & . \\ . & . & . & . & . \\ 0 & . & . & . & a_M(i) \end{bmatrix} \rightarrow (4.3)$$

$$\boldsymbol{a_j(i)} = \begin{cases} 1 \; if \;\; j \in j_{(i \bmod R)} + 1 \\ 0 \;\; otherwise \end{cases}$$

As an example for M =4 and N =2 and we have S =6 and P_1 ={1,2}, P_2 ={1,3}, P_3 ={1,4}, P_4 ={2,3}, P_5 ={2,4}, P_6 ={3,6}. The R coefficient subsets to be updated where R =2 can be arranged into periodic sequence with corresponding coefficient selection matrix $I_{N,k}^{(i)}$ by equation 4.3. In general there is no unique way to design periodic coefficient subsets as long as the N subsets used in sequential partial updates satisfy the above mentioned conditions. Updating N coefficients in an adaptive filter of length M at each iteration leads to a complexity reduction in the adaption process roughly proportional to R.We will discuss some of the performance metrics like convergence properties of the sequential partial update technique.

4.1.1 Convergence performance

The results for the convergence performance of the sequential incremental algorithm are shown. The total number of coefficients assumed for the filter M is 10. For the 70% coefficient update, 7 coefficients to be updated in each iteration i.e. N =7. Step-size parameter (μ_k) is taken as 0.03. The MSE obtained from simulation is 0.1203.

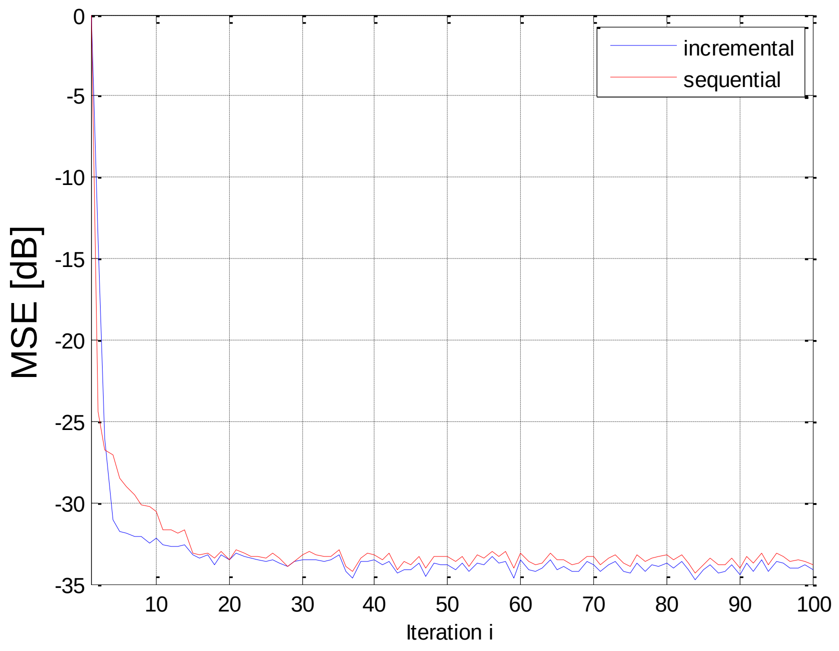

Figure 4.1 Time averaged MSE for sequential partial update LMS for the 70 % coefficient update and incremental LMS

For a 50% coefficient update 5 coefficients (N =5) are updated in each iteration and obtained MSE is 0.2005. The MSE for full-update incremental algorithm mentioned in the previous chapter is 0.0078

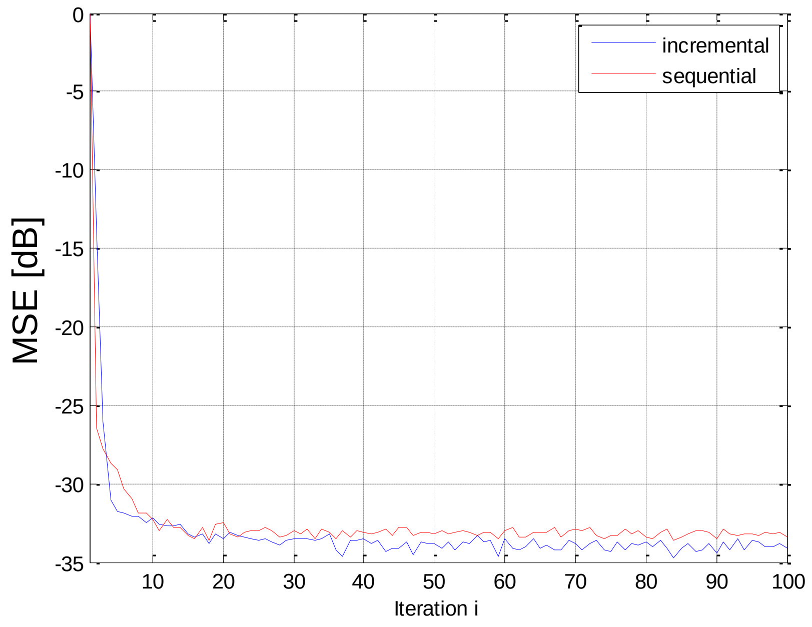

Figure4.2 Time averaged MSE for sequential partial update LMS for the 50 % coefficient update and incremental LMS

For 30% coefficient update 3coefficients (N =3) are updated in each iteration, MSE obtained is 0.2255 and the results are compared with an incremental algorithm in which all the coefficients are updated whose MSE is 0.0078.

Other performance techniques like EMSE and MSD are also calculated for each node of the network. For describing the steady-state quantities as a function of fixed step-size μ_k . Correlation index, noise power profile, power profile of the network has in following figure. It would be desirable to drive the whole network to an equalized performance. A good step-size design, together with the cooperative scheme that has been proposed, may take advantage of the spatial diversity provided by the adaptive network. By properly tuning the step size at each node, a good level of performance equalization could be achieved throughout the network. Nodes presenting poor performance, or high noise level, can be assigned with small step sizes, such that, in the limit case, they would become simply relay nodes.

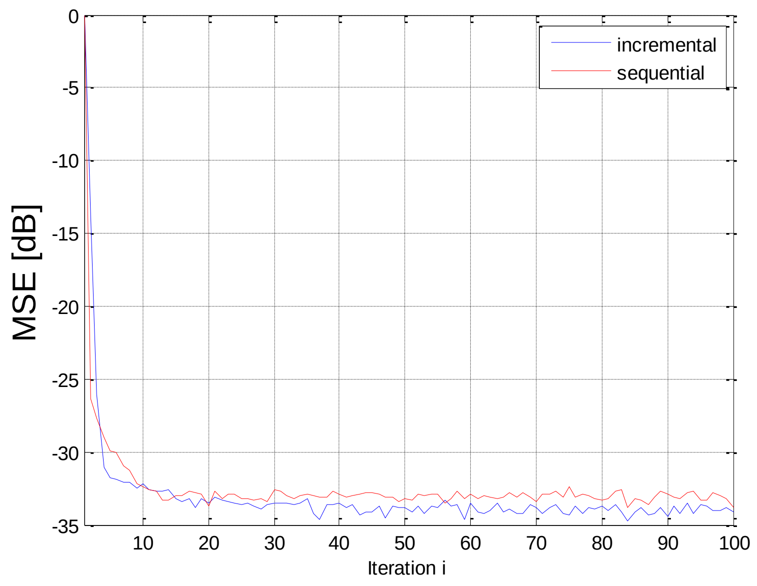

Figure 4.3 Time averaged MSE for sequential partial update LMS for the 30 % coefficient update and incremental LMS

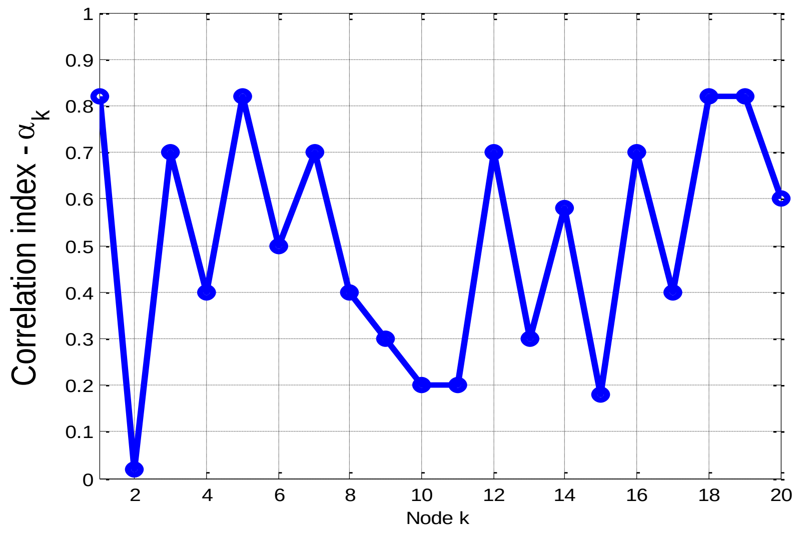

Figure 4.4 correaltion index of the nodes

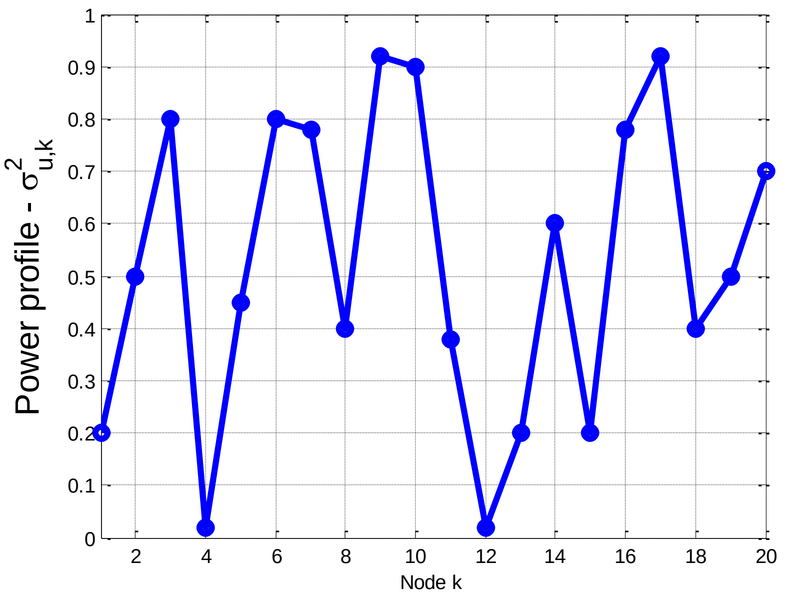

Figure 4.5 noise power profile of the networks

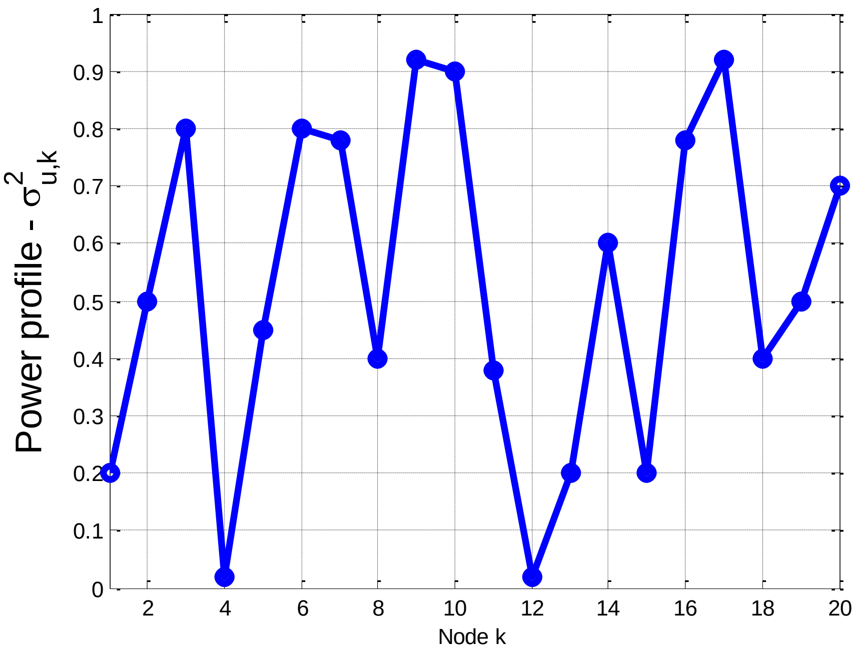

Figure 4.6 Power profile of the nodes

4.2 Stochastic partial-update Incremental LMS

Stochastic partial update improves the performance of the network over the sequential partial update algorithm with the same amount of computational complexity reduction. In this method coefficient subset to be updated are chosen randomly instead of deterministic fashion as in sequential partial update algorithm.

The update equation is given by

$$\phi_k^{(i)} = \phi_{k-1}^{(i)} + \mu_k v_{k,i}^{*} I_{N,k}^{(i)} e_k^{(i)} \qquad \rightarrow (3)$$

The coefficient selection matrix is given by

$$I_{N,k}^{(i)} = \begin{bmatrix} a_1(i) & 0 & . & . & 0 \\ 0 & a_2(i) & . & . & . \\ . & . & a_3(i) & . & . \\ . & . & . & . & . \\ 0 & . & . & . & a_M(i) \end{bmatrix},$$

$$a_j(i) = \begin{cases} 1 if\ j \in j_{m(i)} \\ 0 otherwise \end{cases}$$

Where $m(i)$ is an independent random process with probability mass-function

$$\Pr\{\ m(i) = c\ \} = \pi_c, \quad c=1...\ R$$

$$\sum_{c=1}^{R} \pi_c = 1$$

The computational complexity of stochastic algorithm is same as that of the sequential algorithm and slower than the Incremental LMS algorithm by a factor R because of the decimation of the adaptive filter coefficient.

4.2.1 Convergence performance

The results for the convergence performance of stochastic algorithm are shown. The total number of coefficients assumed for the filter M is 10. Fig 4.7 shows for a 70% coefficient update, 7 coefficients to be updated in each iteration i.e. $N=7$. Step-size parameter (μ_k) is taken as 0.03. The MSE obtained from simulation is 0.1389.

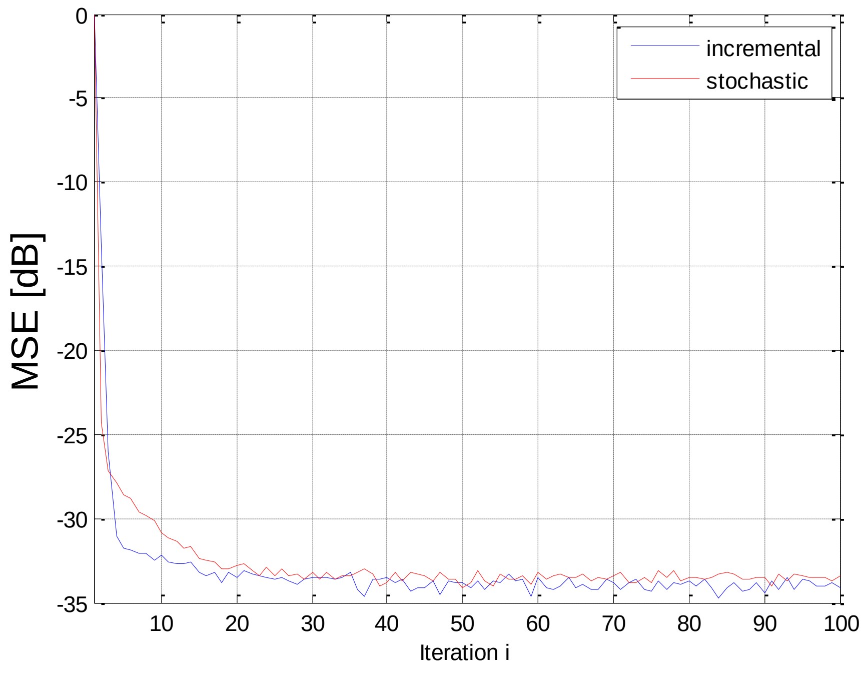

Figure4.7 Time averaged MSE for stochastic partial update Incremental LMS for the 70 % coefficient update and incremental LMS

Fig4.8 shows for 50% coefficient update 5 coefficient ($N=5$) are updated in each iteration and obtained MSE is 0.1910. The MSE for full-update incremental algorithm mentioned in the previous chapter is 0.0078. The simulation results were shown in the following figure for the 50 percent update.

Fig4.9 sows for 30% coefficient update 3coefficients ($N=3$) are updated in each iteration, MSE obtained is 0.2255 and the results are compared with an incremental algorithm in which all the coefficients are updated whose MSE is 0.0078.

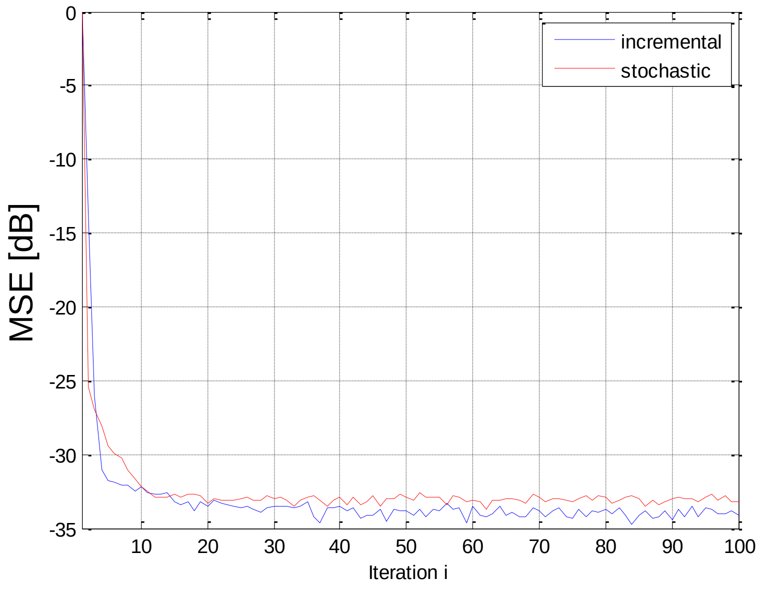

Figure 4.8 Time averaged MSE for stochastic partial update Incremental LMS for the 50 % coefficient update and incremental LMS

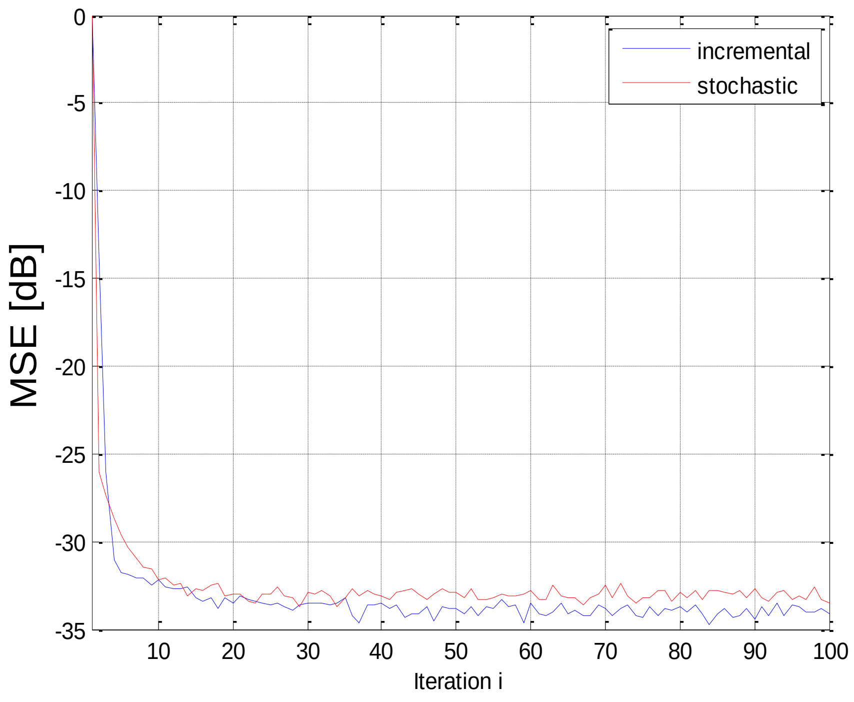

Figure4.9 Time averaged MSE for stochastic partial update Incremental LMS for the 30 % coefficient update and incremental LMS

Fig4.9 sows for 30% coefficient update 3coefficients (N =3) are updated in each iteration, MSE obtained is 0.2255 and the results are compared with an incremental algorithm in which all the coefficients are updated whose MSE is 0.0078.

4.3 Max- partial update incremental LMS

In this algorithm at iteration largest magnitude vector entries are updated. This is a data dependent partial update technique which is based on finding N largest magnitude entries from M total coefficients [4].

The update equation is given by

$$\phi_k^{(i)} = \phi_{k-1}^{(i)} + \mu_k v_{k,i}^{*} I_{N,k}^{(i)} e_k^{(i)} \rightarrow \quad (4)$$

$$e_k^{(i)} = d_k(i) - v_{k,i}\varphi_{k-1}^{(i)}$$

The coefficient selection matrix $I_{N,k}^{(i)}$ is given by

$$I_{N,k}^{(i)} = \begin{bmatrix} a_1(i) & 0 & . & . & 0 \\ 0 & a_2(i) & . & . & . \\ . & . & . & . & . \\ . & . & . & . & 0 \\ 0 & . & . & . & a_N(i) \end{bmatrix},$$

$$a_j(i) = \begin{cases} 1 \; if \; |v(i-j+1)| \in \max\limits_{1\le l\le M}\left(|v(i-l+1)|, N\right) \\ 0 \; otherwise \end{cases}$$

The Max partial update is similar to the sequential update method in decimating the coefficient update vector, but the magnitude of the update vector entries to be ranked before updating instead of deterministic fashion in sequential update method. The coefficient selection scheme determines the convergence of the algorithm. This reduces the complexity by a factor $R = M / N$.

4.3.1 Convergence performance

The results for the convergence performance of the Max - partitioning algorithm is shown. The total number of coefficients assumed for the filter M is 10. Fig 5.10 shows for a 70% coefficient update, 7 coefficients to be updated in each iteration i.e. N =7. Step-size parameter (μ_k) is taken as 0.03. The MSE obtained from simulation is 0.0593. The simulation result is shown in the following figure with comparison for full-update incremental LMS

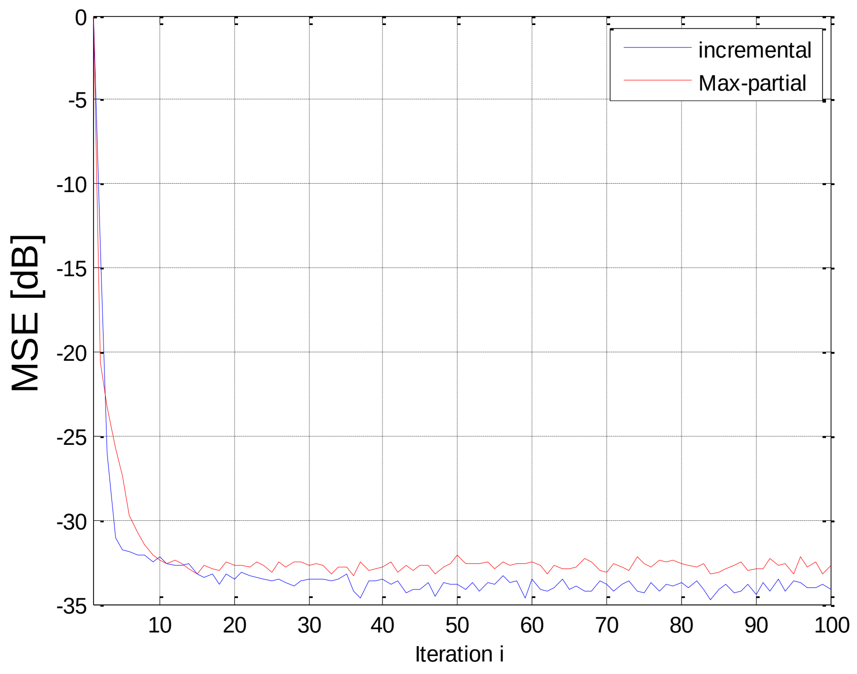

Figure 4.10 Time averaged MSE for Max-partial update Incremental LMS for the 70 % coefficient update and incremental LMS

Fig 4.11 shows for 50% coefficient update 5 coefficient (N =5) are updated in each iteration and obtained MSE is 0.1014. The MSE for full-update incremental algorithm mentioned in the previous chapter is 0.0078. The simulation results were shown in the following figure for the 50 percent update

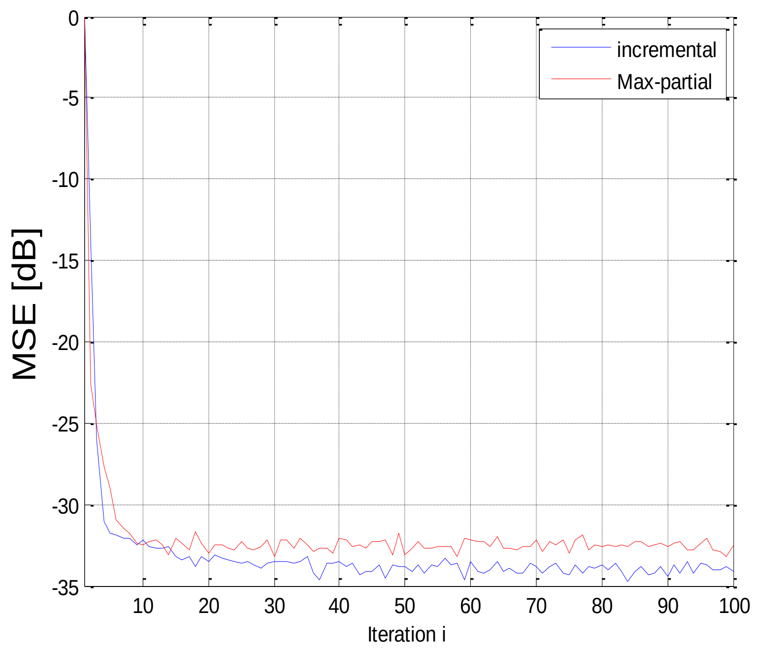

Figure 4.11 Time averaged MSE for Max-partial update Incremental LMS for the 50 % coefficient update and incremental LMS

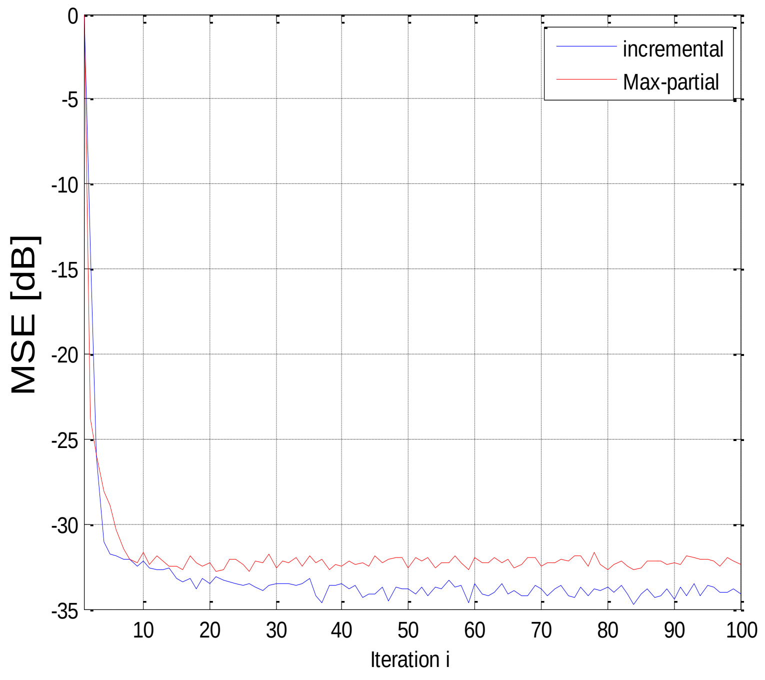

Figure 4.12 Time averaged MSE for Max-partial update Incremental LMS for the 30 % coefficient update and incremental LMS

Fig 4.12 shows for 30% coefficient update 3coefficients (N =3) are updated in each iteration, MSE obtained is 0.1954 and the results are compared with the incremental algorithm in which full-update of coefficients takes place whose MSE is 0.0078

The simulation results for performance estimation are shown after each technique. Partial update techniques are compared with incremental LMS in which all the coefficients are updated at each iteration. In all the cases Max-partial outperforms sequential partial and stochastic partial incremental techniques in performance and Stochastic technique gives better performance over sequential for the same computational complexity. But sequential partial update technique converges with a faster convergence rate compared to stochastic and Max algorithms. Stochastic algorithm converges at a faster rate over Max algorithm. The advantage of proposed algorithms over incremental algorithm is achieved at the cost of degradation in performance. From the simulation results it is obvious

- MSE depends on number of coefficients updated.
- It is more sensitive to local statistics.
- Since incremental mode of communication is taken into account each node k is influenced by its immediate neighbors only.

4.4 Summary

Here we review and compare all the techniques that were studied in the previous chapter. The three Incremental partial update techniques developed are compared with each other and with incremental algorithm developed chapter 3. The MSE is taken as the performance metric which gives the convergence rate. It is obvious that percentage of saving the computations degrades the performance. All the techniques mentioned here are compared for 70%, 50%, and 30% coefficient update. The fallowing results give the clear analysis over each technique.

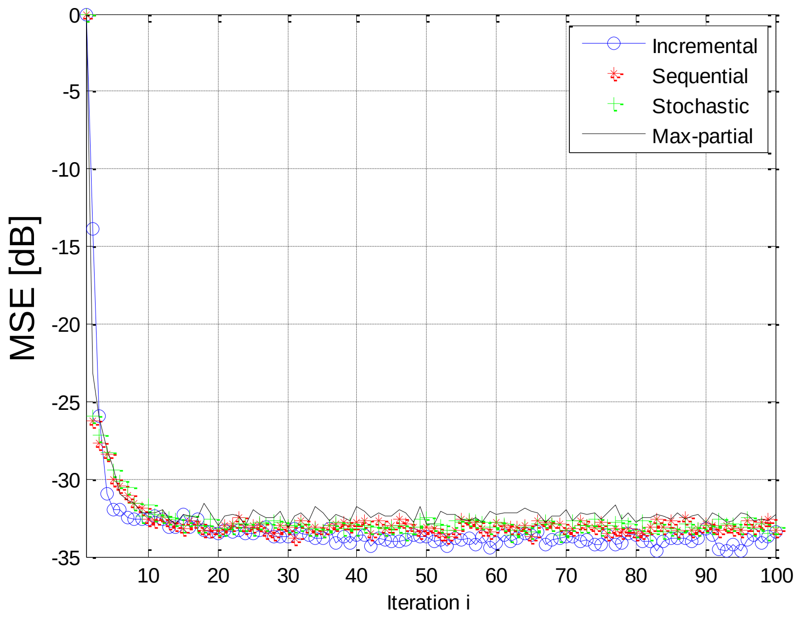

Figure4.13 Comparison of each technique with incremental LMS for 70% coefficient update

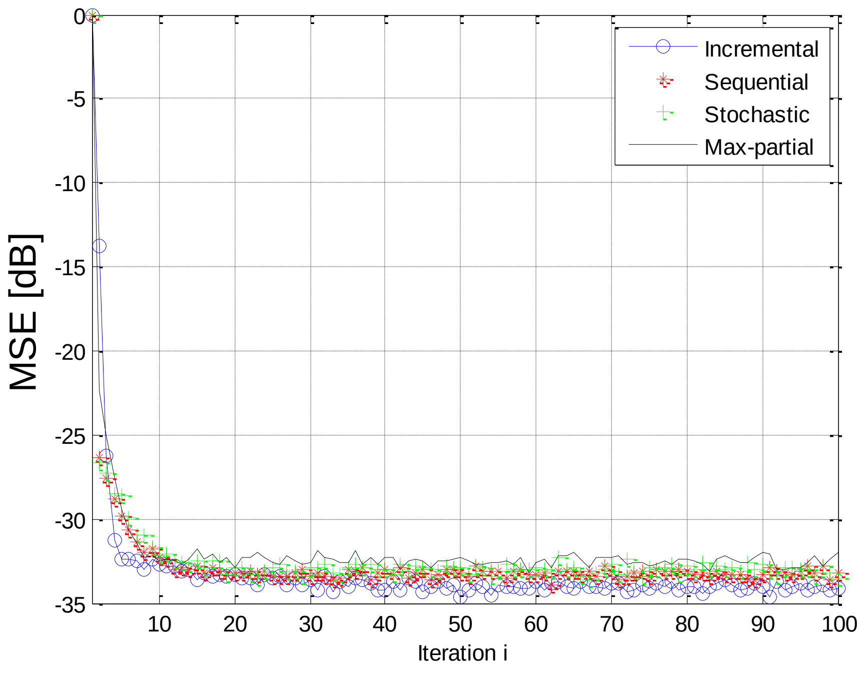

Figure4.14 Comparison of each technique with incremental LMS for 50% coefficient update

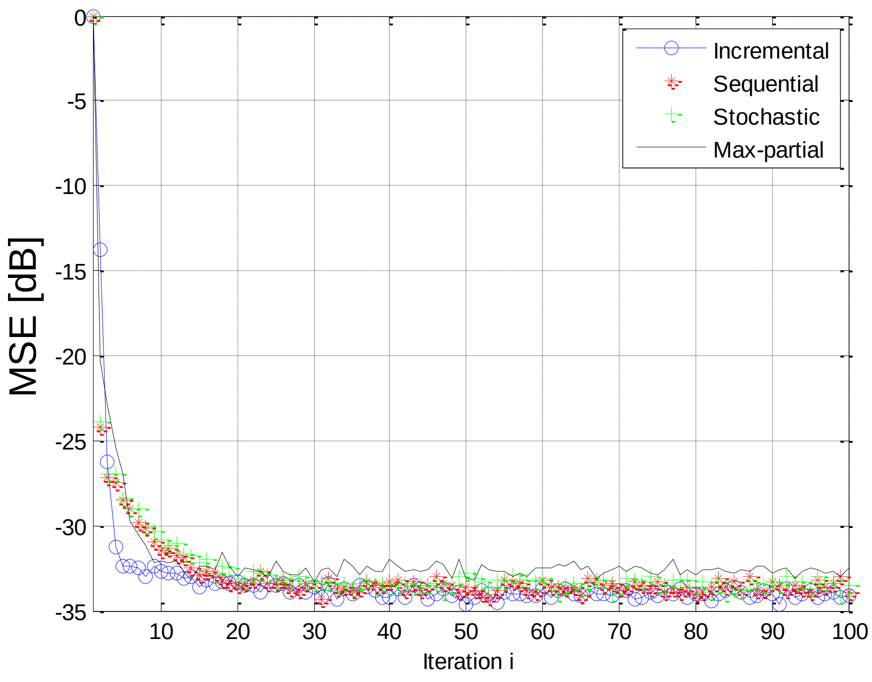

Figure4.15 comparison of each technique with incremental LMS for 30% coefficient update

For all the algorithms proposed here ring type topology is considered as shown in Fig.1. In sequential incremental algorithm number of coefficients M is 10. For 70% coefficient update, 7 coefficients to be updated in each iteration i.e. N =7. Step-size parameter is taken as 0.03.The MSE obtained from simulation is 0.1203. For 50% coefficient update 5 coefficient (N =5) are updated in each iteration and obtained MSE is 0.2005. For 30% coefficient update 3coefficients (N =3) are updated in each iteration, MSE obtained is 0.2255 and the results are compared with incremental algorithm in which all the coefficients are updated whose MSE is 0.0078.

In stochastic incremental algorithm the parameters are same as in sequential and MSE calculated from simulation results is 0.1389 for 70% update, 0.1910 for 50% update and 0.1954 for 30% coefficient update respectively.

In max-incremental algorithm also the parameters are same as in sequential algorithm and stochastic algorithms.MSE is 0.0593 for 70%, 0.1014 for 50% and 0.1416 for 30%coefficient update.

The simulation results for performance estimation are shown in Figs.4.13,4.14 and 4.15. Partial update techniques are compared with incremental LMS in which all the coefficients are updated at each iteration. In all the cases Max-partial outperforms sequential partial and stochastic partial incremental techniques in performance and Stochastic technique gives better performance over sequential for the same computational complexity. But sequential partial update technique converges with a faster convergence rate compared to stochastic and Max algorithms. Stochastic algorithm converges at a faster rate over Max algorithm. The advantage of proposed algorithms over incremental algorithm is achieved at the cost of degradation in performance.

Chapter 5

Conclusion and future work

5.1 Conclusion

In this thesis we focus on reducing the communication and computational complexity for the distributed wireless sensor networks [1]. The adaptive algorithms developed assume the robustness of the standard LMS algorithm. It is clear from the analysis that sequential and stochastic partial update algorithms reduces the computation complexity in equal manner but stochastic partial update algorithm give better performance over sequential. Max-partial algorithm converges quickly and has consistent steady state performance and reduces the computational complexity in the same amount as of other two techniques.

So with a little deterioration in the performance the computational complexity can be reduced to considerable amount. This in turn reduces the power consumption and is suitable for networks with low-energy sources.

5.2 Scope for future work

- These techniques can be implemented for networks using diffusion mode of communication which involves heavy computational complexity.
- These techniques have the possibility for no-linear cases like artificial neural networks.
- Networks using probabilistic diffusion mode of communication can implemented by this techniques.
- Communication complexity can be further reduced by transmitting differential estimation.

Bibliography

[1]. Cassio G Lopes and Ali H.Sayed , "incremental adaptive strategies over distributed networks", *IEEE Transaction on Signal processing*,vol.55,No.8,Aug.2007.

[2]. Mahesh.Godavarti and Alfred O.Hero, "Sequential partial update LMS algorithm" *IEEE Trans. on signal processing*, Oct. 2001.

[3]. Mahesh.Godavarti and Alfred O.Hero, "Stochastic partial update LMS algorithm" *IEEE Trans. on signal processing*, Oct. 2001.

[4]. Kutluyil Dogancay *partial-update adaptive signal processing Design, analysis and implementation.* Academic press,2008.

[5]. Mahesh.Godavarti and Alfred O.Hero, "Partial Update LMS algorithm", *journal of latex class files*, vol.1, No.11, November 2002.

[6]. M.G Rabbat and R.D.Nowak "Quantized incremental algorithms for distributed optimization", *IEEE J.Sel.areas commin.*,vol.23,no.4.pp798-808,Apr.2005.

[7]. D. Estrin, G. Pottie, and M. Srivastava, "Intrumenting the world with wireless sensor setworks," in *Proc. IEEE Int. Conf. Acoustics, Speech,Signal Processing (ICASSP)*, Salt Lake City, UT, May 2001, pp.2033–2036.

[8]. C. G. Lopes and A. H. Sayed, "Distributed processing over adaptive networks," in *Proc. Adaptive Sensor Array Processing Workshop*, MIT Lincoln Lab., Lexington, MA, Jun. 2006.

[9]. A. H. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley,2003.

[10]. A. H. Sayed and C. G. Lopes, "Distributed recursive least-squares strategies over adaptive networks," in *Proc. Asilomar Conf. Signals,Systems, Computers*, Monterey, CA, Oct. 2006, pp. 233–237.

[11]. M. Wax and T. Kailath, “Decentralized processing in sensor arrays,”*IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-33, no. 4, pp.1123–1129, Oct. 1985.

[12]. R. Olfati-Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” *IEEE Trans. Autom.Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[13]. J. Tsitsiklis and M. Athans, “Convergence and asymptotic agreement in distributed decision problems,” *IEEE Trans. Autom. Control*, vol.AC-29, no.1, pp. 42–50, Jan. 1984.

[14]. L. A. Rossi, B. Krishnamachari, and C.C. J. Kuo, “Distributed parameter estimation for monitoring diffusion phenomena using physical models,” in *Proc. 1st IEEE Conf. Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA, Oct. 2004, pp.

[15]. C. G. Lopes and A. H. Sayed, “Distributed adaptive incremental strategies: Formulation and performance analysis,” in *Proc. IEEEInt. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Toulouse,France, May 2006, vol. 3, pp. 584–587.

[16]. C. G. Lopes and A. H. Sayed, “Diffusion least-mean squares over adaptive networks,” in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Honolulu, HI, Apr. 2007, pp. 917–920.

[17]. A. H. Sayed and C. G. Lopes, “Distributed recursive least-squares strategies over adaptive networks,” in *Proc. Asilomar Conf. Signals,Systems, Computers*, Monterey, CA, Oct. 2006, pp. 233–237.

[18]. A. Nedic and D. Bertsekas, “Incremental sub gradient methods for non-differentiable optimization,” *SIAM J. Optim.*, vol. 12, no. 1, pp.109–138, 2001.

[19]. M. G. Rabbat and R. D. Nowak, “Decentralized source localization and tracking,” in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Montreal, QC, Canada, May 2004, vol. III, pp.921–924.

[20]. L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” in *Proc. 4th Int. Symp. Information Processing in Sensor Networks*, Los Angeles, CA, Apr. 2005, pp.63–70.

[21]. K. Doğançay and O. Tanrikulu, “Generalized subband decomposition LMS algorithm employing selective partial updates,” *Proc. of the IEEE Conf.on Acoustics Speech and Signal Processing, ICASSP 2002*, vol. 2, pp 1377-1380.

[22]. S. Douglas, “Adaptive filters Employing Partial Updates*,” IEEE Trans. on Circuits and System*s,CAS-II, Vol. 44, No 3., pp. 209-216, March 1997.

[23]. S. Werner, M. L.R. de Campos and Paulo Diniz,“Partial update NLMS algorithms with data-selective partial updating,” *IEEE Transactions on Signal Processing*, vol. 52, no. 4, pp. 938-949, April 2004.

[24]. K. Doğançay and O. Tanrikulu, “Adaptive filtering algorithms with selective partial updates*,” IEEE Transactions on Circuits And Systems-II*, vol. 48, no.8, pp. 762-769, August 2001.