DISTRIBUTED ESTIMATION OVER ADAPTIVE NETWORKS

BY

MUHAMMAD OMER BIN SAEED

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

ELECTRICAL ENGINEERING

JUNE 2011

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MUHAMMAD OMER BIN SAEED** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING**.

Dissertation Committee

Dr. Azzedine Zerguine (Adviser)

Dr. Salam A. Zummo (Co-adviser)

Dr. Asrar-ul-Haq Sheikh (Member)

Dr. Mohamed A. Deriche (Member)

Dr. Tareq Y. Al-Naffouri (Member)

Dr. Ali A. Al-Shaikhi
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

25/7/11
Date

Dedicated to the loving memories of my beloved Abujan,
Muhammad Akbar Bhatti, and Nanajan, Muhammad Aslam Bhatti

ACKNOWLEDGMENTS

In the name of Allah, the Most Beneficent, the Most Merciful

All praise be to Allah (The One and The Only Creator of everything) for His limitless blessings. May Allah bestow peace and His choicest blessings on His last prophet, Hazrat Muhammad (Peace Be Upon Him), his family (May Allah be pleased with them), his companions (May Allah be pleased with them) and his followers.

First and foremost gratitude is due to my esteemed university, King Fahd University of Petroleum and Minerals for my admittance, and to its learned faculty members for imparting quality learning and knowledge with their valuable support and able guidance that has led my way through this point of undertaking my research work.

I would like to express my profound gratitude and appreciation to my thesis committee chairman and adviser, Dr. Azzedine Zerguine, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and skill in many areas. Dr. Zerguine is the one professor who truly made a difference in my life. It was under his tutelage that I developed a focus and became interested in research. He provided me with

direction and intellectual support, first during course work and then during the research phase, and became more of a mentor than a professor. It was through his persistence, understanding and kindness that I completed my degree. I doubt that I will ever be able to convey my appreciation fully, but I owe him my eternal gratitude.

I would like to thank my co-adviser Dr. Salam Zummo for his support and assistance throughout the thesis as well as my course work. Special thanks to my committee member Dr. Tareq Al-Naffouri for his assistance and guidance in the mathematical analysis of the thesis. I would also like to thank the other committee members, Dr. Asrar-ul-Haq Sheikh and Dr. Mohamed Deriche for their constructive and positive criticism.

I would like to specially thank Dr. Maan Kousa for his valuable support and advice during my time here at KFUPM. I cannot forget the lessons taught by him inside and outside of the classroom that have proved to be extremely valuable life lessons for me.

I would like to make a special acknowledgement to Dr. Muhammad Moinuddin, Dr. Zahid Ali, Muhammad Ajmal, Noman Tasadduq, Yasir Haroon and Saqib Sohail, for their constant support and guidance throughout my time here at KFUPM, and especially during my research work. I shall always treasure their friendship.

I also cannot forget my friends who provided me with the motivation and moral support to carry on with my work. A few of them Mohammed Aabed, Syed

Muhammad Asad, Aamir Imam, Jubran Akram, Mazhar Azeem, Muhammad Akhlaq, Asif Siddiqui, Omair Khan, Ali Zaidi, Ahmed Salim, Suhaib Al-Baseet and many others whom I shall not be able to name here. Their friendship made my stay here at KFUPM pleasant and enjoyable.

Finally, I would like to thank my parents for their prayers and support from the core of my heart. I would also like to thank my wife and my siblings for the moral support that they provided.

May Allah help us in following Islam according to Quran and Sunna!(Ameen)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: Muhammad Omer Bin Saeed

TITLE OF STUDY: Distributed Estimation Over Adaptive Networks

MAJOR FIELD: Electrical Engineering

DATE OF DEGREE: June 2011

Recently a lot of interest has been shown in parameter estimation using ad hoc wireless sensor networks. An ad hoc wireless sensor network is devoid of any centralized fusion center and thus, has a distributed structure. Several algorithms have been proposed in the literature in order to exploit this distributed structure in order to improve estimation. One algorithm that was practically sound as well as fully distributed was called Diffusion LMS (DLMS) algorithm. In this work, variations to the DLMS algorithm are incorporated.

The first algorithm improves the DLMS algorithm by varying the step-size of the algorithm and eventually the Variable Step-Size DLMS (VSSDLMS) algorithm is setup. Well known VSSLMS algorithms are compared, then the most suitable algorithm identified to provide the best trade-off between performance and complexity is chosen.

Next, an algorithm is derived using the constraint that the noise variance is known. This algorithm is akin to the VSSDLMS algorithm but is computationally more complex. Convergence and steady-state analyses are carried out in detail for both algorithms. The effect of mismatch in noise variance estimate is studied for the constraint based algorithm. Extensive simulations are carried out to assess the performance of the proposed algorithms. Simulation results are found to corroborate the theoretical findings.

Finally a new scenario is investigated. All the algorithms existing in literature assume knowledge of regressor data. However, this information is not always available. This work studies blind algorithms for adaptive networks. Inspired by second order statistics based blind estimation methods, two algorithms are first converted into recursive block blind algorithms. Then these algorithms are applied to the adaptive network scenario using the diffusion scheme. Simulation results are carried out to assess the performance of the algorithms under different scenarios.

Keywords: *Diffusion least mean square algorithm, Variable step-size least mean square algorithm, Noise constrained least mean square algorithm, Blind estimation algorithm, Distributed network.*

ملخص الرسالة

الاسم الكامل: محمد عمر بن سعيد

عنوان الرسالة: التقدير الموزع للشبكات المتكيفة

التخصص: هندسة كهربائية.

تاريخ الدرجة العلمية: يونيو 2011

تبين مؤخرا الكثير من الاهتمام في تقدير المعلمة باستخدام شبكات الاستشعار اللاسلكية المخصصة, تخلو شبكات الاستشعار اللاسلكية المخصصة من مركز انصهار مركزي وبالتالي فان لديها بنية موزعة. لقد تم اقتراح العديد من الخوارزميات في الأدب من أجل استغلال هذه البينية الموزعة من اجل تحسين التقدير, سميت إحدى الخوارزميات والتي كانت عمليا ذات بنية موزعة كليا بخوارزمية (DMLS). في هذا العمل أدرج متغيرات لخوارزمية (DMLS).

أول خوارزمية طورت من خوارزمية (DLMS) وذلك من خلال تغير حجم الخطوة ومن هنا تم إعداد خوارزمية (DMLS) ذات حجم خطوة متغير (VSSDLMS). تمت المقارنة مع خوارزمية (VSSLMS) ومن ثم يتم اختيار أنسب خوارزمية عرفت بتقديم أفضل مفاضلة بين الأداء والتعقيد.

تم بعد ذلك اشتقاق خوارزمية باستخدام قيد معرفة تفاوت الضوضاء, هذه الخوارزمية قريبة من خوارزمية (VSSDLMS) لكنها حسابيا أكثر تعقيدا. تم إجراء تحاليل الالتقاء والحالة الثابتة بشكل مفصل لكلا الخوارزميتين. تم دراسة تأثير عدم التطابق في تفاوت الضوضاء للخوارزمية المعتمدة على القيد.

تم إجراء محاكاة واسعة النطاق لتقييم أداء الخوارزميات المقترحة, وكانت نتائج هذه المحاكاة معززة لما هو في النتائج النظرية.

تم أخيرا التحقق من سيناريو جديد. جميع الخوارزميات الموجودة في الأدب تفترض معرفة بيانات المتغير المستقل (Regressor), ومع ذلك فان هذه المعلومات غير متوفرة دائما. في هذا العمل تم دراسة الخوارزمية العمياء للشبكات المتكيفة. تم باستوحاء من طرق التقدير العمياء ذات إحصاءات الدرجة الثانية أولا تحويل خوارزميتين إلى خوارزمية عمياء ذات كتلة عودية. بعد ذلك هذه الخوارزمية تم تطبيقها على سيناريو شبكة متكيفة باستخدام نظام النشر. وتم تنفيذ نتائج المحاكاة لتقييم أداء خوارزميات تحت سيناريوهات مختلف.

Nomenclature

Abbreviations

ATC	:	Adapt-then-Combine
BBRC	:	Blind Block Recursive Cholesky algorithm
BBRS	:	Blind Block Recursive SVD algorithm
DBBRC	:	Diffusion Blind Block Recursive Cholesky algorithm
DBBRS	:	Diffusion Blind Block Recursive SVD algorithm
DLMS	:	Diffusion Least Mean Square algorithm
DLMSAC	:	Diffusion Least Mean Square with Adaptive Combiners algorithm
DRLS	:	Diffusion Recursive Least Squares algorithm
DSLMS	:	Distributed Least Mean Square algorithm
EMSE	:	Excess Mean Square Error
FC	:	Fusion Center
LMS	:	Least Mean Square algorithm
MMSE	:	Minimum Mean Square Error
MSD	:	Mean Square Deviation
MSE	:	Mean Square Error
NCDLMS	:	Noise Constrained Diffusion Least Mean Square algorithm
NP	:	non-deterministic polynomial-time
RC	:	Recursive Cholesky algorithm
RCNV	:	Recursive Cholesky with No Variance estimation algorithm

RS	:	Recursive SVD algorithm
SNR	:	Signal-to-Noise Ratio
SS-EMSE	:	Steady-State Excess Mean Square Error
SS-MSD	:	Steady-State Mean Square Deviation
SVD	:	Singular Value Decomposition
VFFBBRC	:	Variable Forgetting Factor BBRC algorithm
VFFBBRS	:	Variable Forgetting Factor Blind Block Recursive SVD algorithm
VSSDLMS	:	Variable Step-Size Diffusion Least Mean Square algorithm
VSSLMS	:	Variable Step-Size Least Mean Square algorithm
WSN	:	Wireless Sensor Network
ZNCDLMS	:	Zero Noise Constrained Diffusion Least Mean Square algorithm

Notations

i	:	Iteration number
d	:	Measured value
$\mathbf{d}$	:	Measured value vector for entire network
$\mathbf{u}$	:	Input regressor vector
$\mathbf{U}$	:	Input regressor matrix for entire network
$\mathbf{w}^o$	:	Unknown vector
$\mathbf{w}^{(o)}$	:	Unknown vector for entire network
$\mathbf{w}_k$	:	Estimation vector for node k
$\mathbf{w}$	:	Estimation vector for entire network
$\mathbf{\Psi}_k$	:	Intermediate estimation vector for node k
$\mathbf{\Psi}$	:	Intermediate estimation vector for entire network
v	:	Scalar noise value
$\mathbf{v}$	:	Noise vector
e	:	Measurement error
J	:	Cost function
E[.]	:	Expectation operator
μ	:	Step-size
M	:	Length of unknown vector
N	:	Number of nodes (for a network)
		Number of blocks (for single node blind scenario)

k, l	:	Node number
$\mathcal{N}_k$	:	Number of neighbor nodes for node k
$\mathbf{R_{u,k}}$	:	Auto-correlation matrix for input regressor vector for node k
$\mathbf{R_U}$	:	Auto-correlation matrix for input regressor matrix for entire network
$\mathbf{r_{du,k}}$	:	Cross-correlation vector between input and output for node k
$.^*$	:	Conjugate
$.^T$	:	Transpose
c_{lk}	:	Combiner weight between nodes k and l
α, γ	:	Controlling parameters for step-size update equation
$\mathbf{D}$	:	Step-size matrix for entire network
$\mathbf{C}, \mathbf{G}$	:	Combiner matrix
$\mathbf{I}$	:	Identity matrix
λ	:	Eigenvalue
$\mathbf{\Lambda}$	:	Eigenvalue matrix
σ	:	Weighting vector
$\mathbf{\Sigma}, \mathbf{F}$	:	Weighting matrix
$\otimes$	:	Kronecker product
$\odot$	:	Block Kronecker product
bvec	:	Block vector operator
$\mathbf{R_v}$	:	Auto-Correlation matrix for noise
$\mathcal{M}$	:	Misadjustment

β : Lagrange multiplier

σ_v^2 : Noise variance

$\tilde{\sigma}_v^2$: Noise variance estimate mismatch

$\mathbf{W}$: Unknown vector convolution matrix

$\mathbf{D_N}$: Data block matrix

$\mathcal{U}$: Hankel matrix

$\mathbf{R_w}$: Auto-correlation matrix for the unknown vector

$\mathbf{g}$: Vector of Cholesky factored matrix

K : Block size

CHAPTER 1

INTRODUCTION

In recent years, wireless sensor networks (WSNs) have become a very hot topic of interest for researchers due to the multiplicity of their uses [1]-[5]. Several applications have emerged that use WSNs with several more in the pipeline [6]. Furthermore, decentralized estimation of signals of interest using WSNs has also attracted much attention recently [7]-[18].

A wireless sensor has the ability to sense the surrounding physical environment, perform signal processing tasks and then communicate relevant information using a wireless transceiver. A large collection of such sensors in a close-bound network is thus referred to as a wireless sensor network (WSN). In order to meet the cost for large scale deployment, sensors are small, inexpensive devices with limited computational and communication capability as well as constrained resources. However, their popularity is due to the fact that despite the constraints, they provide the user with cost-effective high performance. The ability of sensor nodes to communicate with each other further enhances their performance, giving rise to

an entirely new area of applications in environmental, domestic as well as military areas.

A few emerging applications utilizing WSNs are distributed field monitoring, localization, surveillance, power spectrum estimation, and target tracking [6]. These applications typically require estimating parameters of interest like temperature, concentration of certain chemicals, and speed and position of targets. Among surveillance applications are detection of critical events such as smoke alarms. Such applications benefit mainly from the distributed structure of a WSN. However, it has recently been realized that without empowering the sensors with some signal processing capability, the ultimate goals cannot be achieved. Sensors need to be empowered with the required signal processing tools that fully utilize the distributive nature of the network as well as provide optimal results. This need has been addressed recently and several algorithms proposed. The aim of this dissertation is to improve upon the existing algorithms [7]-[16], as well as to provide a novel algorithm that is more suitable in applications where the source is unknown.

The chapter is organized as follows. A background for wireless sensor networks and adaptive filtering is given in the context of estimation and the least mean square (LMS) algorithm is given as an example. This is followed by a detailed literature survey. The aims of the dissertation are then briefly explained and the chapter ends with a layout of the dissertation.

1.1 Background

1.1.1 Wireless Sensor Networks

There are generally two types of WSNs used in practice (see Fig. 1.1). One has a central processing unit known as a Fusion Center (FC). The sensors usually sense the required data and then transmit the data via a wireless channel to the fusion center. The sensors do not perform much processing except quantizing and coding the data before transmitting it to the fusion center. The fusion center acts as a data sink where data from all sensors is collected and then processed in order to ascertain the estimates of the parameters of interest. Unlike sensors, a fusion center has large processing capability as well as storage capacity.

A network devoid of a fusion network is generally termed as an *ad hoc* network. The sensors only communicate with neighboring sensors that are within communication range. In such a network, the sensors have access to data from their neighboring sensors only that can be attained via a wireless communication link between the sensors. The sensors are required to do a two-fold process in such cases. First, they need to acquire the available data from the nearby neighbors. Then each sensor performs some signal processing on the available data in order to estimate some parameter of interest. This is usually done over several iterations, with each iteration improving the previous estimate. Since the sensors do not have access to the entire network, the estimate may differ for each sensor. Iterative algorithms make certain that sensors can reach a mutually agreeable

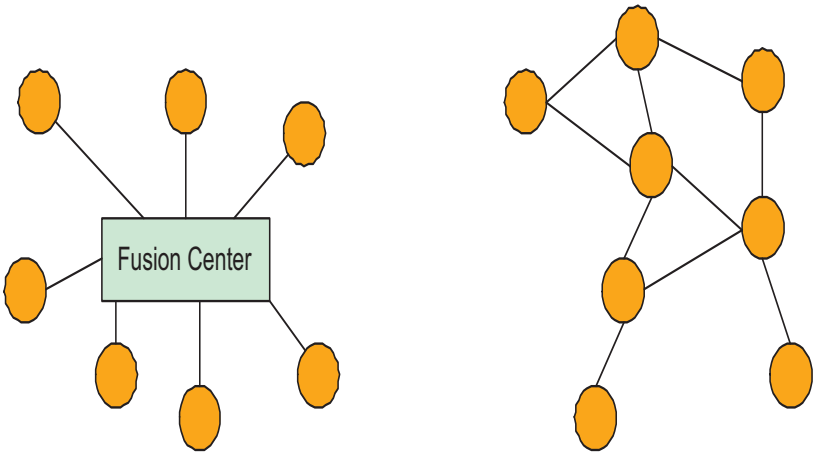

Figure 1.1: (a) A Fusion Center-based WSN; (b) An ad hoc topology

estimate. This is usually achieved by the sensors by exploiting the spatial diversity of the network available to them. In most cases, however, such an estimate is only reached asymptotically. Still, the network behaves as a self-organized entity even in the absence of a fusion center but with added intelligence and signal processing capabilities.

The network with a fusion center has a major drawback in that if the center fails then the whole network falls into disarray. Also, there exists the problem of communicating with the sensors that are located far away. Such sensors would need higher power in order to transmit their data to the center. This problem may be overcome using a multi-hop system within the network where the data from distant sensors hops from one sensor to another until it reaches the center. However, this adds additional complexity to the system but still provides an alternative to using high power, which will utilize the battery of a sensor more rapidly, thus causing the sensor to shut down. Despite this solution, the problem of the center failing still remains. In comparison, ad hoc networks do not have such a limitation as they are working without any such processing center. Even if some sensors fail, the performance degradation is the only problem to worry about in ad hoc WSNs as the network still continues to function.

It is usually the case that FC-based topologies benchmark the performance among the class of decentralized estimators that can be implemented using WSNs. This is because all network-wide information is centrally available for processing, whereas in ad hoc WSNs sensor data spreads via single-hop exchanges among

neighboring sensors. There is an inherent delay till a given sensor can make use of all the data collected by the WSN. Hence, intermediate local sensor estimates will generally be of lower quality when compared to those formed by the FC.

Resource allocation, medium access control and general communication protocols for in-network processing schemes are interesting problems in their own right, and have been studied by several researchers. However, this work looks at improving techniques for distributed estimation in WSNs.

1.1.2 Adaptive Filtering

Any system that gives a certain output given some input can be regarded as a filter. If that filter has the ability to adapt itself according to the changing surroundings in order to keep a certain parameter constant or within certain predefined limits, then such a filter is regarded as an adaptive filter. Adaptive filters are generally used in applications where some unknown parameters need to be estimated. An example is that of channel estimation in a communication system. In a communication system information travels from the transmitter to the receiver via some medium that is known as the channel. In wireless communications the channel is usually unknown and needs to be identified at the receiver in order to estimate the possible transformation that may have occurred on the transmitted information whilst it was propagating through the wireless channel. In order to identify/estimate the channel a system is needed that can adapt itself until there is an approximate match. Thus, an adaptive filter is used. A known information

signal is transmitted through the channel and the transformed signal is received at the receiver by the adaptive filter. The aim of the adaptive filter is to take in the same input signal and try to adapt itself such that its output matches that of the channel. This is usually an iterative process. At each iteration the adaptive filter outputs a certain value of the signal and tries to match it to the received signal. Based on the error, the adaptive filter adjusts itself and repeats the process. This goes on until the error between the output of the channel and the output of the filter is under a certain threshold. At this point the communication system can start functioning as the channel has been estimated and its effects can now easily be nullified.

An interesting point to note here is that the actual measure to check the performance is not the error itself. Error between the two outputs can be positive or negative and is generally a zero-mean process. Therefore, it is not reliable to develop an algorithm for adaptation based on just the error. A much better quantity would be the squared error or the absolute error. The simplest algorithms usually tend to minimize the mean square error. The error between the two outputs is squared and minimized. Repeating this process over several experiments generally gives a measure of how well the algorithm is performing. Hence the term mean square error (MSE). Recently, another measure is being adopted by researchers called mean square deviation (MSD). Instead of measuring the error between the channel output and the filter output, performance is measured by looking at the error between the coefficients of the channel and the filter. This shows how far off

the filter is from reaching the actual channel. This error is usually a vector and its inner product gives the squared deviation value at every iteration. Repeating the process several times and averaging gives the MSD which then describes how well the algorithm is performing. The simplest and most common adaptive filter in use is called Least Mean Square (LMS) Algorithm.

1.1.3 Least Mean Square Algorithm

Consider an unknown system defined by its parameters that can be represented in a column vector, $\mathbf{w}^o$, of length $(M \times 1)$. An adaptive filter is used to identify the unknown system. The input and output of the system are defined at every iteration as $\mathbf{u}(i)$ and scalar $d(i)$, respectively, where $\mathbf{u}$ is a row vector of length $(1 \times M)$ and d is a scalar given by

$$d(i) = \mathbf{u}(i)\,\mathbf{w}^o + v(i), \tag{1.1}$$

where v is a zero-mean noise being added at the output of the unknown system and i denotes the time instant. If the adaptive filter is represented at every iteration by a column vector $\mathbf{w}(i)$ then the noise-free output of the filter, $y(i)$ will be given by

$$y(i) = \mathbf{u}(i)\,\mathbf{w}(i). \tag{1.2}$$

The error is thus defined as

$$\begin{aligned} e(i) &= d(i) - y(i) \\ &= \mathbf{u}(i)\,\mathbf{w}^o + v(i) - \mathbf{u}(i)\,\mathbf{w}(i). \end{aligned} \tag{1.3}$$

Here, a cost function is defined for the adaptive filter. The aim is to minimize the cost function by choosing the appropriate value of $\mathbf{w}$ that matches $\mathbf{w}^o$ as best possible. The cost function is defined as

$$J = \mathrm{E}\left[|d(i) - y(i)|^2\right], \tag{1.4}$$

where J is the expected value of the square of the error. We thus have a mean square error-based algorithm. Minimizing this cost function with respect to $\mathbf{w}$ and after some evaluations, the resulting algorithm is called Least Mean Square (LMS) algorithm, which is a realization of the minimum mean square error (MMSE) algorithm. The update equation for the adaptive filter for LMS algorithm is, thus, given by

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu e(i)\,\mathbf{u}^*(i), \tag{1.5}$$

where $\mathbf{u}^*(i)$ is the transpose conjugate of the input vector at iteration i. This equation describes the LMS algorithm which is the simplest adaptive filtering algorithm. It has some drawbacks but because of its simplicity and ease of implementation, it is preferred in most applications.

The work being proposed in this thesis deals with adaptive filtering in WSNs and LMS algorithm, in particular, has been used mainly in literature for distributed estimation. A brief literature survey of related work will be presented next.

1.2 Literature Survey

The advent of WSNs has created renewed interest in the field of distributed computing, calling for collaborative solutions that enable low-cost estimation of stationary signals as well as reduced-complexity tracking of non-stationary processes. Different from WSN topologies that include an FC, ad hoc ones are devoid of hierarchies and rely on in-network processing to effect agreement among sensors on the estimate of interest. A great body of literature has been amassed in recent years, building up the field of consensus-based distributed signal processing. The tutorial in [19] gives general results and a vast list of related works and thus serves as a good initial reading for a beginner in the field.

In [20], consensus among nearest neighbors was considered to coordinate vehicular motion among particles such that they are made to move in the same direction despite the absence of a centralized system for coordination. The authors in [21] discuss consensus issues among various types of graphs and provide some theoretical basis for their further development. The authors in [22] develop methods for getting the best possible consensus average in a distributed network by studying the results over several vast networks. A recent work suggests projection into linear subspace in order to overcome constraints of centralized processing and thus suggests a decentralized algorithm that utilizes consensus to produce results similar to a centralized system [23]. This technique was used for successful spectrum sensing with a decentralized network in cognitive radios [24]. A constraint-based solution for decentralized estimation was presented in [9],[10] that treats each

sensor as a separate entity and thus provides a simple algorithm for distributed estimation that results in a solution to which all sensors converge asymptotically. In all the schemes mentioned thus far, sensors collect data all at once and then reach consensus by locally exchanging messages.

The work in [25] provides a frequency-domain version of the tracking algorithm for mobile sensor networks. A consensus algorithm is used for mobile tracking in frequency-domain. The authors in [26] provide further extension of the work done for tracking in mobile environments by providing algorithms for sensor fusion using novel consensus filters and also suggest methods for designing such filters. The authors in [27] suggest exchanging sequential peer-to-peer data in order to achieve a least squares solution. The algorithm provides good results but at the cost of high computational cost as well as requiring extensive communication between sensors. Also, the algorithm is not robust enough to tackle the problem of estimating time-varying signals or dynamic systems. This problem of using ad hoc WSNs for distributed state estimation of dynamical systems has also received a lot of attention recently. The authors in [28] suggest a diffusion scheme for distributed Kalman filtering. The data between neighboring sensors is diffused before each sensor updates its own estimate using a Kalman filter, thus improving the overall performance of the system. The work in [29] provides a similar diffusion scheme for Kalman smoothing. In [30], three different distributed Kalman filtering algorithms are designed and discussed. Solutions to some potential problems are also suggested. The works in [10] and [31] suggest distributed Kalman filtering

and Kalman tracking algorithms but with certain constraints on channel models.

In many applications however, sensors need to perform estimation in a constantly changing environment without having available a (statistical) model for the underlying processes of interest. This motivates the development of distributed adaptive estimation algorithms, the subject dealt with in the current work. The first such approach introduced a sequential scheme, whereby information circulated through a topological cycle in conjunction with LMS-type adaptive filtering per sensor, allowing the network to account for time variations in the signal statistics [7]. This information could comprise of estimation updates of the node and/or the complete data available. The sensors follow a Hamiltonian cycle and each sensor transmits its update to the next sensor in the cycle, which then uses its own data to update this estimate. The sensors use newly acquired data at each iteration to update the estimate, thus, accounting for any time variations in the process. In [32], a more general algorithm for the incremental scheme of [7] is given. Such incremental schemes may provide a solution that converges faster than a centralized solution as well as providing a low steady-state error at a very low complexity cost. These features make the incremental algorithm very attractive. However, in case of node failure, the cycle is broken and the network would seize to function until the cycle is regenerated using the remaining functioning sensor nodes. This process is a non-deterministic polynomial-time (NP)-hard problem [33]. For medium- to large-sized WSNs this can be a very complex problem and not very applicable. Also, time delay can cause problems in cases where applica-

tions are time-critical. A solution to the node failure problem was suggested in [14]. However, the computational complexity of the algorithm increased at the cost of performance degradation.

A new algorithm was proposed in [8] that got rid of the topological constraints in [7] and fully exploited the distributed nature of the network. The computational cost increased but the overall solution was more practical. The algorithm was termed as (combine-then-adapt) diffusion LMS. Each sensor forms a convex combination of the local estimates acquired from the nearby neighbors. This combined estimate is then used in the LMS recursion to update the local estimate. This process is repeated for each sensor at each iteration. An improved version of the algorithm was suggested in [34] in which the steps of the process were reversed, that is, the LMS recursion updates the local estimate at each sensor at every iteration and then the convex combination of the estimates of the nearby neighbors is taken. This new (adapt-then-combine) diffusion LMS improves the performance over the previous algorithm and provides a variant of the algorithm that was originally proposed in [35]. Another way to improve performance is to diffuse not only the local estimates but also the sensor observations to nearby neighbors [34, 15]. This results in improving the flow of data across the WSN but can be computationally expensive, especially in the presence of communication noise. The authors in [15] also give a generalized form of the diffusion LMS algorithm as well as a detailed-yet-generic analysis of the possible variants that were separately addressed in previous publications. A variant of diffusion

LMS was suggested in [13] in which the network was divided into several small networks, each having its own diffusion LMS algorithm network. A sensor node from each small network was then chosen as a *supernode* and then the supernodes formed a new network which conducted its own diffusion LMS algorithm. The hierarchical structure provided a multilevel diffusion LMS algorithm that improved performance but at the cost of extra computational cost. Another variant used adaptive combiners instead of a fixed combination technique [16]. At each iteration, a sensor forms an error-based matrix from the estimates of the previous two iterations of each nearby neighbor. Based on the cross-correlation of this error matrix, the combiner weight of a certain neighbor is selected at every iteration and then the diffusion LMS algorithm is run. This scheme slightly improved results but was computationally very extensive. For applications where fast convergence is required and sensors can be burdened with increased computational load, a distributed RLS scheme was introduced in [36]. A detailed analysis and design of the diffusion RLS scheme was given in [37]. An improved variant of [7] was introduced in [14]. The authors use a Markovian chain to randomly select the order of the cycle for the incremental scheme at each iteration. This solves the problem of the link failure faced by the original algorithm in [7] but comes at the price of extra computational cost required to select the next sensor node every time.

Several distributed estimation algorithms are dependent on iterative optimization methods, which capitalize upon the separable structure of the cost function

that defines the specific estimator. The sample mean estimator was formulated in [38] as an optimization problem, and was solved in a distributed fashion using a primal dual approach such as the one described in [39]. Similarly, the incremental schemes in [7, 32, 36, 40] are all based in incremental (sub)gradient methods that have been described generally in [41] and specifically for applications in [42]. Even the diffusion LMS algorithm in [8] has been recently shown to have a connection with incremental strategies, when these are applied to optimize an approximate reformulation of the LMS cost [34]. Building on the framework introduced in [9, 10], the Distributed-LMS algorithm was developed in [43], and was obtained upon recasting the respective decentralized estimation problem as multiple equivalent constrained subproblems. The resulting minimization subtasks can be simultaneously solved across sensors, when carried out using the alternating-direction method of multipliers (AD-MoM) [39, 44]. As a result, the algorithm divides the sensors into bridge nodes and data nodes, connected via Lagrangian multipliers. The constraint set upon the estimates is such that all nodes reach the same estimate asymptotically. The work in [12] gives a detailed performance analysis of this algorithm comparing it with diffusion LMS algorithms given in [8, 34]. Based on a similar approach, the distributed RLS algorithm was developed in [11]. However, the hierarchical division of bridge and data nodes was removed using the alternating minimization algorithm of [45], making it simpler and more robust. Similar ideas have been applied in spectrum cartography for cognitive radio networks [46], distributed demodulation [47] and distributed classification

[48].

Recently, the diffusion algorithm was used to synchronize the movement of mobile sensors moving towards a specific target [49]. This work showed the robustness of the algorithm in an environment where the parameters being estimated are constantly changing. Each sensor has access to a direction vector as well its own position. The data being sensed by each node is simply the position of the target towards which the network has to travel. Since the sensed data is noisy, the exact position has to be estimated and each node has to make sure that it is moving in sync with the rest of the nodes. Therefore, each node estimates the position of the target and also updates its own position and speed with respect to its neighboring sensors. Although the work presented in [49] is application specific, yet it can be extended to other applications, showing the usefulness of the algorithm for systems working in stationary as well as non-stationary environments.

All the algorithms discussed above assume that each node has access to complete regressor data. The problem boils down to estimating an unknown system for which the input and output are known. This problem is akin to a system identification problem. However, in certain applications this luxury is not available. Taking the example of the work in [49], the direction vector may not be known to the sensor nodes. This makes the problem a blind estimation problem. So far no research has been done to find out a solution for the blind estimation problem. Although blind estimation has been studied in great detail in literature yet none of the blind estimation algorithms have been applied to the case of distributed

estimation in a WSN.

1.3 Dissertation Contributions

The Thesis contributions are briefly listed here. First, a variable step-size diffusion least mean square (VSSDLMS) algorithm is formulated. A generalized version of the algorithm is also proposed. A complete performance analysis of the algorithm is carried out.

Next, a noise-constrained diffusion least mean square (NCDLMS) algorithm is derived based on the constraint that the noise variance is known. A generalized version of the new algorithm is also proposed. A complete performance analysis is carried out including the case where the noise variance is not estimated correctly and also when the noise variance is completely omitted from the calculations.

Finally, blind block diffusion algorithms are proposed. Inspired from algorithms based on second order statistics, two recursive algorithms are proposed and then used for blind block estimation over adaptive networks. Their performance and computational complexity is discussed.

1.4 Dissertation Layout

The remainder of this thesis is organized as follows. Chapter 2 describes the system model and details the problem statement. The diffusion LMS algorithm [15] is given as an existing solution. Chapter 3 introduces variable step-size LMS

(VSSLMS) algorithms and states the benefits of using these algorithms. The VSSLMS algorithm providing the best trade-off between complexity and performance is then chosen and incorporated in the diffusion scheme. The resulting algorithm, named variable step-size diffusion LMS (VSSDLMS) algorithm is then studied in detail. Convergence and steady-state analyses are carried out followed by simulations results for the proposed algorithm under different scenarios. Chapter 4 introduces and derives the so-called noise-constrained diffusion LMS (NCDLMS) algorithm after motivating the need to use the said constraint. Convergence and steady-state analyses are carried out. At the end, simulation results are given followed by a discussion on the performance of the proposed algorithm. Chapter 5 begins with a motivation for the need of blind algorithms. Two blind estimation algorithms are discussed. These algorithms are then converted into a recursive form and then incorporated in the diffusion scheme. Simulation results compare the algorithms followed by a discussion on the performance of the algorithms. Chapter 6 lists the contributions of this work followed by some future recommendations and ends with a conclusion to this work.

CHAPTER 2

DISTRIBUTED ESTIMATION PROBLEM AND NETWORK MODEL

In this work, different algorithms have been developed to deal with the problem of distributed estimation over adaptive wireless sensor networks (WSNs). This chapter gives an overview of the system model of a wireless sensor network and then formulates the problem statement for distributed estimation. In the end, existing solutions to the problem are briefly explained.

2.1 System Model

Consider a set of sensor nodes spread over a geographical area in close proximity to each other as depicted in Fig. 2.1. The nodes are assumed to be spread in

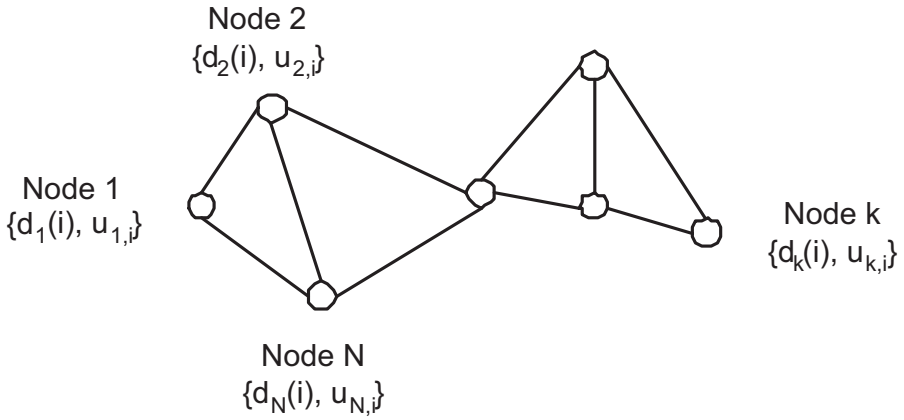

Figure 2.1: Adaptive network of N nodes.

such a fashion that the whole network is interconnected. However, each node has access to only the neighbor nodes which are within communication range. Data is shared over a single-hop only even though the nodes are all connected through multi-hops. Each node k has access to scalar measurements d_k, which are sensed by the node, as well as the regressor vector $\mathbf{u}_k$, which is of size $(1 \times M)$. The neighbors of any node k are given by $\mathcal{N}_k$.

2.2 Problem Statement

The purpose of the nodes in the network is to estimate a certain parameter of interest. The parameter can be denoted by a vector $\mathbf{w^o}$ of size $(M \times 1)$. The scalar measurement sensed by node k, d_k at any time instant i, is given as

$$d_k(i) = \mathbf{u}_k(i)\,\mathbf{w}^o + v_k(i), \tag{2.1}$$

where $v_k(i)$ is zero-mean additive white noise. The simplest solution to this estimation problem is for each node to estimate the unknown vector using only its own set of data. Such a case is termed as the no cooperation case as the nodes are not communicating with each other. The spatial diversity of the nodes is not being utilized here and so this case is counter productive as the poor performance of the nodes with low SNR will result in poor performance of the network. In order to obtain a fully distributed estimation algorithm, a cost function is needed that defines the complete network. Thus, the cost function is defined as follows:

$$
\begin{aligned}
J(\mathbf{w}) &= \sum_{k=1}^{N} J_k(\mathbf{w}) \\
&= \sum_{k=1}^{N} \mathrm{E}\left[|d_k - \mathbf{u}_k\mathbf{w}|^2\right]. \qquad (2.2)
\end{aligned}
$$

Consequently, the steepest descent solution for this problem is given as

$$
\mathbf{w}_k(i+1) = \mathbf{w}_k(i) + \mu \sum_{k=1}^{N} (\mathbf{r}_{d\mathbf{u},k} - \mathbf{R}_{\mathbf{u},k}\mathbf{w}_k(i)), \qquad (2.3)
$$

where $\mathbf{r}_{d\mathbf{u},k} = \mathrm{E}\left[d_k\mathbf{u}_k^T\right]$ is the cross-correlation between d_k and $\mathbf{u}_k$, and $\mathbf{R}_{\mathbf{u},k} = \mathrm{E}\left[\mathbf{u}_k^T\mathbf{u}_k\right]$ is the auto-correlation of $\mathbf{u}_k$. The recursion (2.3) requires full knowledge of the statistics of the entire network. Moreover, it requires exact statistical knowledge of the data, which is not possible in a practical scenario. A more practical solution utilizes the distributive nature of the network by dividing the cost function into local cost functions that add up to the global cost function. The solution to the local cost functions is similar to (2.3). However, a practical approach leads to the use of the least mean square (LMS) algorithm as a solution. The work in [8] gives a fully distributed solution, which is modified and improved in [15]. Using the *Adapt-then-Combine* (ATC) scheme, the diffusion LMS (DLMS) algorithm [15] is given as

$$
\begin{aligned}
\mathbf{\Psi}_k(i+1) &= \mathbf{w}_k(i) + \mu_k\mathbf{u}_k(i)\left[d_k(i) - \mathbf{u}_k(i)\,\mathbf{w}_k(i)\right] \\
\mathbf{w}_k(i+1) &= \sum_{l\in\mathcal{N}_k} c_{lk}\mathbf{\Psi}_l(i+1), \qquad (2.4)
\end{aligned}
$$

where $\mathbf{\Psi}_k(i+1)$ is the intermediate update, c_{lk} is the weight connecting node k to its neighboring node $l \in \mathcal{N}_k$ and can be fixed according to a chosen rule [8], and μ_k is the step-size for the k^{th} node. Each node uses its own set of data, $\{d_k(i), \mathbf{u}_k(i)\}$, to get an intermediate update for the estimate. Then intermediate updates from neighbor nodes are combined together through a weighted sum to get the final update for the estimate.

Unlike c_{lk} in (2.4), which have constant values, the work in [16] improves the DLMS algorithm by introducing adaptive combiner weights $c_{lk}(i)$. An error matrix is defined for each node at every iteration, defined as

$$\Delta\psi_k(i) \triangleq [\psi_l(i) - \psi_l(i-1)]_{l\in\mathcal{N}_k}, \tag{2.5}$$

which is used to form the error correlation matrix given by

$$\mathbf{Q}_{\Psi,k} = \Delta\Psi_k^T(i)\,\Delta\Psi_k(i). \tag{2.6}$$

Based on this matrix, new weighting metrics are calculated and the combiner weights are then updated using these metrics. Ultimately, the second equation in (2.4) becomes

$$\mathbf{w}_k(i+1) = \sum_{l\in\mathcal{N}_k} c_{lk}(i)\mathbf{\Psi}_l(i+1), \tag{2.7}$$

where the combiner weights are updated at every iteration. This algorithm requires $\mathcal{N}_k^2 M$ extra multiplications as well as $\mathcal{N}_k^2(M-1)$ extra additions just to evaluate the error correlation matrix at each iteration. For a relatively large sized network, the algorithm becomes very heavy. An improvement in performance is

achieved using this algorithm but at the cost of significant increase in computational complexity.

Another algorithm was suggested in [11] based on the constraint that each node arrives at the same estimate asymptotically. As a result of this constraint, the cost function is modified accordingly to look like the following:

$$J(\mathbf{w}) = \sum_{k=1}^{N} \left(\mathrm{E}\left[|d_k - \mathbf{u}_k \mathbf{w}_k|^2 \right] + \mathbf{p}_k^T \sum_{l \in \mathcal{N}_k} (\mathbf{w}_k - \mathbf{w}_l) + \sum_{l \in \mathcal{N}_k} \frac{c}{2} \|\mathbf{w}_k - \mathbf{w}_l\|^2 \right), \tag{2.8}$$

where $\mathbf{p}_k$ defines the Lagrangian vector for node k, $\mathcal{N}_k$ defines the number of neighbors for node k and c is a positive constant. The distributed algorithm is given as

$$\mathbf{p}_k(i) = \mathbf{p}_k(i-1) + c \sum_{l \in \mathcal{N}_k} (\mathbf{w}_k(i) - \mathbf{w}_l(i)), \tag{2.9}$$

and

$$\mathbf{w}_k(i+1) = \mathbf{w}_k(i) \quad + \quad \mu_k \left[2\mathbf{u}_k^T(i) e_k(i) - \mathbf{p}_k(i) - c \sum_{l \in \mathcal{N}_k} (\mathbf{w}_k(i) - \mathbf{w}_l(i)) \right]. \tag{2.10}$$

As can be seen from (2.9) and (2.10), the algorithm is computationally more complex than the DLMS algorithm. However, the performance of this algorithm is not as good as that of the DLMS algorithm except when there is a very noisy connection between neighbor nodes. Even in such a case the performance of the two algorithms is comparable, making the DLMS algorithm the preferred choice.

2.3 Network Model

The network setup is done as follows. The sensor nodes are spread randomly over an area normalized to (1×1) square units. Based on the amount of transmitting power each node is allowed, the communication range r is set. All the nodes that are within the range r of any node k comprise the neighbors of that node k. This model is followed throughout this work. It is also assumed that communication between nodes is noise free.

Consider a network of $N = 20$ nodes with communication range $r = 0.3$. Here we compare the performance of the above given algorithms at an SNR of 20 dB. The combiner weights for the DLMS algorithm are formed using the Metropolis rule [8]. The value for c in the distributed LMS algorithm is chosen to be 1. The results are shown in Fig. 2.2. As shown in the figure, the DLMS with adaptive combiners algorithm performs best. Despite its computational complexity, the distributed LMS algorithm does not perform well compared with the DLMS algorithm.

Next, the effect of varying the network size is compared for all the algorithms. Figure 2.3 shows how the connectivity varies with the range for each node. The average number of neighbors per node increases as the number of nodes in the network increases. Figure 2.4 shows how the performance of each algorithm improves as the network size increases resulting in greater connectivity for each node. The figure shows steady-state MSD values. For a small network size, the performance of each algorithm is similar. However, the performance of the DLMS algorithm

and the DLMS with adaptive combiners algorithm improves significantly as the connectivity increases. The increase in connectivity results in an increase in computational complexity as well. However, as shown in Fig. 2.4, this increase in complexity is well compensated by improvement in performance.

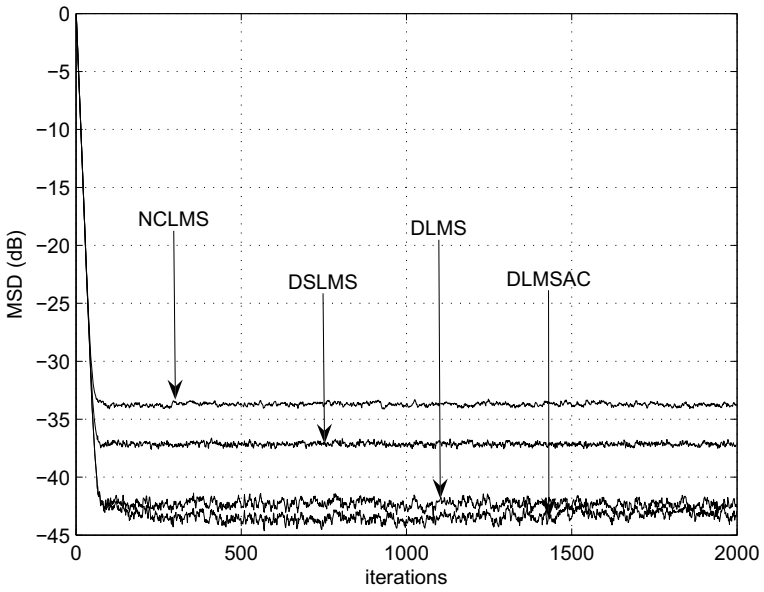

Figure 2.2: MSD comparison for SNR of 20 dB with $N = 20$ and $r = 0.3$.

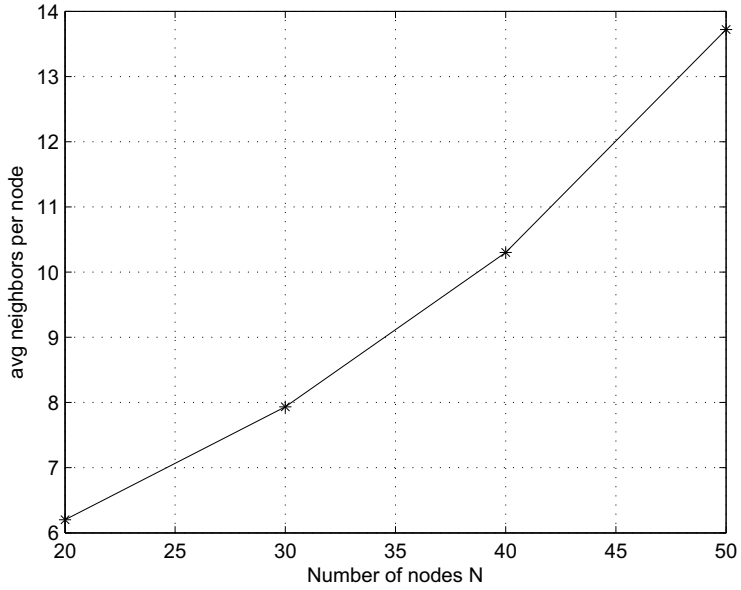

Figure 2.3: Average number of neighbors per node vs number of nodes for $r = 0.3$.

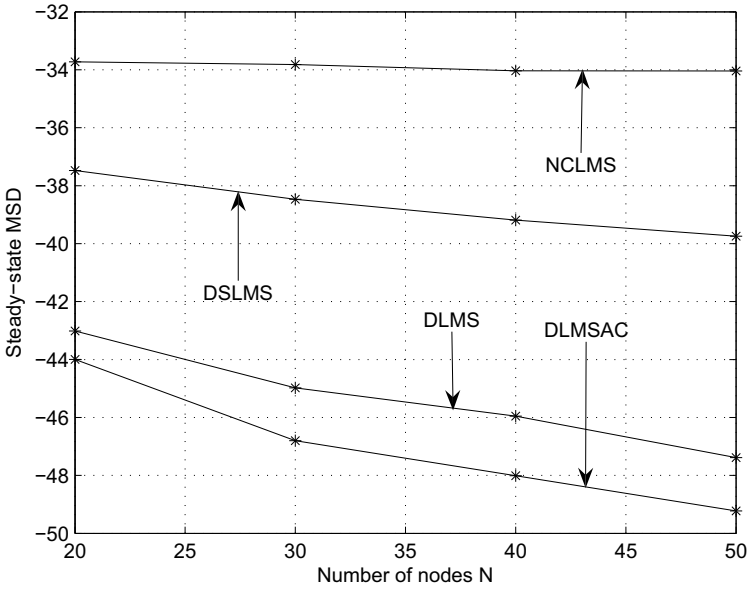

Figure 2.4: Steady-state MSD values for varying number of nodes and $r = 0.3$.

CHAPTER 3

VARIABLE STEP-SIZE DIFFUSION LEAST MEAN SQUARE ALGORITHM

3.1 Introduction

The DLMS algorithm [8],[15] uses a fixed step-size LMS algorithm. Improvement in performance is achieved through the use of adaptive combiner weights in [16] but the cost is heavy computational complexity as shown in [16]. Taking motivation from the algorithm in [16], the work in this chapter improves the DLMS algorithm by using a variable steps-size LMS (VSSLMS) algorithm [50]-[54]. Here, first the well known VSSLMS algorithms are compared, then the most suitable algorithm identified to provide the best trade-off between performance and complexity is chosen. Second, detailed convergence and steady-state analyses are carried out.

Next, computational complexity of the proposed algorithm is compared with that of the previous algorithms. Finally, extensive simulations are carried out to test the robustness of the proposed algorithm under different scenarios. Moreover, the simulation results are found to corroborate the theoretical findings very well.

3.2 Variable Step-Size LMS (VSSLMS) Algorithms

Many VSSLMS algorithms have been devised over the years to eliminate the problem of the LMS algorithm with fixed step-size. Some have been designed for specific applications. Other algorithms, however, are general and can be slightly modified to fit a certain application. Most algorithms provide good convergence with a low error floor but are computationally complex. Two algorithms that are widely regarded as simple yet effective were developed separately in [50] and [51]. An improved version for [50] was presented in [52]. Another variant of the VSSLMS algorithm was suggested in [53] and further improved in [54]. In the ensuing, these algorithms are discussed and used for comparison in this work.

For the LMS algorithm, the update equation is given by

$$\mathbf{w}\left(i+1\right)=\mathbf{w}\left(i\right)+\mu\mathbf{u}^{T}\left(i\right)e\left(i\right), \tag{3.1}$$

where $\mathbf{w}\left(i\right)$ is the estimate of the unknown vector, μ is the step-size, $\mathbf{u}\left(i\right)$ is a row input regressor vector, and $e\left(i\right)$ is the instantaneous error. Here, the step-size μ

is fixed. On the other hand, the algorithms that vary the step-size use different strategies to control the variations in the step-size. The step-size can initially be taken large but within the stability range, for fast initial convergence. The update equation then automatically adjusts the step-size so that it becomes small, resulting in a low steady-state mean square error. The initial fast convergence allows the algorithm to reach the error floor much faster than a fixed step-size LMS algorithm does. The parameters of the algorithm, however, have to be chosen carefully so that the algorithm does not diverge.

The algorithm in [50] provides remarkable improvement in performance with small computational complexity and is therefore the most opted VSSLMS algorithm. Only a few extra multiplications and additions are required. However, the algorithm is directly dependent on the energy of the measurement noise and may result in a large steady-state misadjustment [51], especially at low SNR. Another approach was presented in [51] to counter this problem. This algorithm uses the correlation between the current error and the previous error in the update equation instead of the error energy. This algorithm tends to remove the problem by using the correlation as the update criterion in order to provide the required solution. The work also shows improved performance over that of [50]. Lately, however, the work in [55] shows that if the parameters are chosen carefully then the VSSLMS algorithm in [50] outperforms that of [51]. The work in [52] enhances the algorithm of [50] by using data correlation in the update equation of the step-size. Finally, the strategies in [53] and [54] simply utilize the cross-correlation

between the present and past regressor input vectors to update the step-size. The various update equations are summarized in 3.1 below. Simulation results show that for the current scenario, the algorithm in [50] provides the best trade-off between performance and complexity. Therefore, this algorithm is adopted in this work.

In [16], the authors introduce a method to vary the combination weights at every iteration in order to improve performance. This method is computationally very complex and results only in slight improvement in performance. To prove useful, a VSSLMS-based algorithm that is computationally efficient and yielding better performance than that of [16] was introduced in [17]. The VSSLMS algorithm used in [17] was adopted from [50]. Next, detailed analysis of the proposed VSSLMS algorithm is carried out in the coming sections. The authors in [17] introduced a VSSLMS based algorithm that is computationally efficient compared with the work in [16] and shows better performance as well. The VSSLMS algorithm used in [17] was taken from [50]. Here, we discuss other VSSLMS algorithms to find the best option.

3.3 The Proposed Algorithm

The VSSLMS algorithms show marked improvement over the LMS algorithm at a low computational complexity [50]-[54]. Therefore, if this variation is inserted in the distributed algorithm this would achieve an improved performance. A heavy

Algorithm	Step-size update equation
Kwong-Johnston [50]	$\mu(i+1) = \alpha\mu(i) + \gamma e^2(i)$
Aboulnasr-Mayyas [51]	$p(i) = \beta p(i-1) + (1-\beta)\, e(i)\, e(i-1)$ $\mu(i+1) = \alpha\mu(i) + \gamma p^2(i)$
Costa-Bermudez [52]	$\mu(i+1) = \alpha\mu(i) + \gamma\left[k\mathbf{u}(i)\,\mathbf{u}^T(i-1) - 1\right] e^2(i)$
Mathews-Xie [53]	$\mu(i+1) = \mu(i) + \gamma e(i)\, e(i-1)\, \mathbf{u}^T(i-1)\, \mathbf{u}(i)$

Table 3.1: Step-size update equations for VSSLMS algorithms

step-size adaptation algorithm would not be suitable because of the physical limitations of the sensor node. However, the algorithms being discussed here are not computationally heavy and are therefore well suited for this application.

The proposed algorithm simply incorporates the VSSLMS algorithm into the diffusion scheme, given here again for ease of explanation

$$\begin{aligned} \mathbf{\Psi}_k(i+1) &= \mathbf{w}_k(i) + \mu_k \mathbf{u}_k(i) \left[d_k(i) - \mathbf{u}_k(i) \mathbf{w}_k(i) \right] \\ \mathbf{w}_k(i+1) &= \sum_{l \in \mathcal{N}_k} c_{lk} \mathbf{\Psi}_l(i+1). \end{aligned} \tag{3.2}$$

Using a VSSLMS algorithm, the step-size will also become variable in this system of equations. Then the variable step-size diffusion LMS (VSSDLMS) algorithm is governed by the following:

$$\begin{aligned} \mathbf{\Psi}_k(i+1) &= \mathbf{w}_k(i) + \mu_k(i) \mathbf{u}_k(i) \left(d_k(i) - \mathbf{u}_k(i) \mathbf{w}_k(i) \right), \\ \mu_k(i+1) &= f\left[\mu_k(i) \right], \\ \mathbf{w}_k(i+1) &= \sum_{l \in \mathcal{N}_k} c_{lk} \mathbf{\Psi}_l(i+1), \end{aligned} \tag{3.3}$$

where $f\left[\mu_k(i)\right]$ is the step-size adaptation function that can be defined by one of the recursions given in Table 3.1. For our analysis we will use the update equation of [50]. So the update for the step-size becomes

$$\begin{aligned} \mu_k(i+1) &= \alpha \mu_k(i) + \gamma \left(d_k(i) - \mathbf{u}_k(i) \mathbf{w}_k(i) \right)^2 \\ &= \alpha \mu_k(i) + \gamma e_k^2(i), \end{aligned} \tag{3.4}$$

where $e_k(i) = d_k(i) - \mathbf{u}_k(i)\,\mathbf{w}_k(i)$.

3.3.1 Generalized VSSDLMS algorithm

A specialized case of the above proposed algorithm is now presented. Assuming that enough time delay is allowed between iterations for the nodes to share twice the amount of data and that the extra energy required can also be compensated, the algorithm can be generalized. Instead of sharing only the intermediate estimates with neighbor nodes, all the available data is shared. The generalized algorithm is thus given by

$$\begin{aligned}
\Psi_{k,gen}(i+1) &= \mathbf{w}_{k,gen}(i) + \mu_{k,gen}(i) \sum_{l \in \mathcal{N}_k} s_{lk} \mathbf{u}_l^T(i) \left(d_l(i) - \mathbf{u}_l(i)\,\mathbf{w}_{l,gen}(i)\right), \\
\mu_{k,gen}(i+1) &= \alpha \mu_{k,gen}(i) + \gamma e_{k,gen}^2(i) \qquad (3.5) \\
\mathbf{w}_{k,gen}(i+1) &= \sum_{l \in \mathcal{N}_k} c_{lk} \mathbf{\Psi}_{l,gen}(i+1),
\end{aligned}$$

where $e_{k,gen}(i) = \sum_{l \in \mathcal{N}_k} s_{lk} \left(d_l(i) - \mathbf{u}_l(i)\,\mathbf{w}_{l,gen}(i)\right)$ and s_{lk} is the combiner weight for the sensor data being shared by node l with node k. As can be seen, each node is now transmitting twice the amount of data compared with the originally proposed algorithm. An extra weighted sum is required and the computational cost increases slightly. However, the main cost will be the added time delay and the extra energy required for processing and sharing the data, which will result in a reduction in lifetime of the sensor node. However, if these can be compensated for, depending on the application, then the generalized algorithm can be used for better performance, as will be shown later through simulation results.

In order to better understand the cost effect of the generalized case, let us look at an example. Suppose that the time required to transmit or receive a packet is τ seconds. The number of neighbors for any node k is $\mathcal{N}_k$. So the total time required for node k to transmit or receive shared data is given by $\mathcal{N}_k\tau$ seconds. If we assume the total processing time for the original algorithm to be τ_1 seconds and the power required per process to be P_k, then the total energy consumed by the node for a single iteration will be given by

$$E_{k,0} = P_k(\mathcal{N}_k\tau + \tau_1). \tag{3.6}$$

The generalized algorithm requires to transmit and receive an extra set of packets and then perform an extra weighted sum operation. So the total time required can be given as $(2\mathcal{N}_k\tau + \tau_1 + \Delta)$ seconds, where Δ seconds is the time required for the extra calculation. So the total energy consumed by the node for a single iteration of the generalized algorithm will be given by

$$\begin{aligned} E_{k,1} &= P_k(2\mathcal{N}_k\tau + \tau_1 + \Delta) \\ &= E_{k,0} + P_k(\mathcal{N}_k\tau + \Delta) \\ &= E_{k,0} + \Delta E_k, \end{aligned} \tag{3.7}$$

where $\Delta E_k = P_k(\mathcal{N}_k\tau + \Delta)$. Since the lifetime of a node is inversely proportional to the energy consumed, the ratio between the new and old lifetimes of the sensor

is given as

$$\begin{aligned} \zeta_k &= \frac{E_{k,0}}{E_{k,1}} \\ &= \frac{E_{k,0}}{E_{k,0} + \Delta E_k} \\ &= \frac{\mathcal{N}_k \tau + \tau_1}{2\mathcal{N}_k \tau + \tau_1 + \Delta} \end{aligned} \tag{3.8}$$

So it can be seen that the energy consumed mainly depends on the time taken by the node to transmit and receive data and to process it. For relatively smaller networks the generalized algorithm can be used without much loss in battery life. However, as the network size grows significantly large, the number of neighbors increases for each node and compared with the processing time, the time required for transmitting and receiving shared data becomes significantly large. As a result, the lifetime of a node becomes nearly half. So, even though the generalized algorithm provides significant improvement in performance, it is not very cost efficient and therefore, is considered here only as a special case.

3.4 Performance analysis

Since data between nodes is exchanged, each update is affected by the weighted average of the previous estimates. Therefore, it is suitable to study the performance of the complete network. Hence, some new variables need to be introduced

and the local variables are transformed into global variables as follows [8]:

$$\begin{aligned}
\mathbf{w}(i) &= col\{\mathbf{w}_1(i), ..., \mathbf{w}_N(i)\}, & \boldsymbol{\Psi}(i) &= col\{\boldsymbol{\Psi}_1(i), ..., \boldsymbol{\Psi}_N(i)\}, \\
\mathbf{U}(i) &= diag\{\mathbf{u}_1(i), ..., \mathbf{u}_N(i)\}, & \mathbf{D}(i) &= diag\{\mu_1(i)\mathbf{I}_M, ..., \mu_N(i)\mathbf{I}_M\}, \\
\mathbf{d}(i) &= col\{d_1(i), ..., d_N(i)\}, & \mathbf{v}(i) &= col\{v_1(i), ..., v_N(i)\}.
\end{aligned}$$

From these new variables a completely new set of equations representing the entire network is formed, starting with the relation between the measurements

$$\mathbf{d}(i) = \mathbf{U}(i)\mathbf{w}^{(o)} + \mathbf{v}(i), \tag{3.9}$$

where $\mathbf{w}^{(o)} = \mathbf{Q}\mathbf{w}^o$ is the global unknown vector and $\mathbf{Q} = col\{\mathbf{I}_\mathrm{M}, \mathbf{I}_\mathrm{M}, ..., \mathbf{I}_\mathrm{M}\}$ is a $MN \times M$ matrix. Similarly, the update equations can be remodeled to represent the entire network

$$\begin{aligned}
\boldsymbol{\Psi}(i+1) &= \mathbf{w}(i) + \mathbf{D}(i)\mathbf{U}^T(i)(\mathbf{d}(i) - \mathbf{U}(i)\mathbf{w}(i)), \\
\mathbf{D}(i+1) &= f[\mathbf{D}(i)], \\
\mathbf{w}(i+1) &= \mathbf{G}\boldsymbol{\Psi}(i+1),
\end{aligned} \tag{3.10}$$

where $\mathbf{G} = \mathbf{C} \otimes \mathbf{I}_M$, $\mathbf{C}$ is a $N \times N$ weighting matrix, where $\{\mathbf{C}\}_{lk} = c_{lk}$, $\otimes$ is the block-Kronecker product and $f[\mathbf{D}(i)]$ is the step-size update function that can be applied individually for each node as the matrix $\mathbf{D}(i)$ is diagonal. For the case of

the VSSLMS algorithm of [50], this update equation will become

$$\mathbf{D}(i+1) = \alpha \mathbf{D}(i) + \gamma \mathbf{E}(i), \tag{3.11}$$

where

$$\mathbf{E}(i) = diag\left\{e_1^2(i)\,\mathbf{I}_M, e_2^2(i)\,\mathbf{I}_M, ..., e_N^2(i)\,\mathbf{I}_M\right\}. \tag{3.12}$$

Considering the above set of equations, the mean analysis, the mean-square analysis and the steady-state behavior of the VSSDLMS algorithm are carried out.

3.4.1 Mean Analysis

To begin with, let us introduce the global weight-error vector, defined as

$$\tilde{\mathbf{w}}(i) = \mathbf{w}^{(o)} - \mathbf{w}(i). \tag{3.13}$$

Since $\mathbf{G}\mathbf{w}^{(\mathbf{o})} \triangleq \mathbf{w}^{(\mathbf{o})}$, incorporating the global weight-error vector into (3.10), we get

$$\begin{aligned}
\tilde{\mathbf{w}}(i+1) &= \mathbf{G}\tilde{\mathbf{\Psi}}(i+1) \\
&= \mathbf{G}\tilde{\mathbf{w}}(i) - \mathbf{G}\mathbf{D}(i)\,\mathbf{U}^T(i)\left(\mathbf{U}(i)\,\tilde{\mathbf{w}}(i) + \mathbf{v}(i)\right) \\
&= \mathbf{G}\left(\mathbf{I}_{MN} - \mathbf{D}(i)\,\mathbf{U}^T(i)\,\mathbf{U}(i)\right)\tilde{\mathbf{w}}(i) - \mathbf{G}\mathbf{D}(i)\,\mathbf{U}^T(i)\,\mathbf{v}(i).
\end{aligned} \tag{3.14}$$

where $\tilde{\mathbf{\Psi}}(i) = \mathbf{w}^{(o)} - \mathbf{\Psi}(i)$. Taking the expectation on both sides of the above equation gives

$$\mathrm{E}\left[\tilde{\mathbf{w}}(i+1)\right] = \mathbf{G}\left(\mathbf{I}_{MN} - \mathrm{E}\left[\mathbf{D}(i)\right]\mathbf{R}_{\mathbf{U}}\right)\mathrm{E}\left[\tilde{\mathbf{w}}(i)\right], \tag{3.15}$$

where we have assumed that the step-size matrix $\mathbf{D}(i)$ is independent of the regressor matrix $\mathbf{U}(i)$ [50]. According to this assumption, for small values of γ,

$$\mathrm{E}\left[\mathbf{D}(i)\,\mathbf{U}^{T}(i)\,\mathbf{U}(i)\right] \approx \mathrm{E}\left[\mathbf{D}(i)\right]\mathrm{E}\left[\mathbf{U}^{T}(i)\,\mathbf{U}(i)\right], \tag{3.16}$$

where $\mathrm{E}\left[\mathbf{U}^{T}(i)\,\mathbf{U}(i)\right] = \mathbf{R}_{\mathbf{U}}$ is the auto-correlation matrix of $\mathbf{U}(i)$. Also, since the measurement noise is spatially uncorrelated, the expectation of the second part of the right-hand side of (3.14) is zero.

From [15], we see that the diffusion algorithm is stable if the combination weights are restricted to within the unit circle. However, in this case stability is also dependent on the step-size. In this case, the algorithm will be stable if

$$\prod_{i=0}^{n}\left(\mathrm{I} - \mathrm{E}\left[\mu_k(i)\right]\mathbf{R}_{\mathbf{u},k}\right) \to 0, \quad as\ \ n \to \infty \tag{3.17}$$

which holds true if the mean of the step-size is governed by

$$0 < \mathrm{E}\left[\mu_k(i)\right] < \frac{2}{\lambda_{\max}\left(\mathbf{R}_{\mathbf{u},k}\right)}, \quad 1 \le k \le N, \tag{3.18}$$

where $\lambda_{\max}\left(\mathbf{R}_{\mathbf{u},k}\right)$ is the maximum eigenvalue of the auto-correlation matrix $\mathbf{R}_{\mathbf{u},k}$.

3.4.2 Mean-Square Analysis

In this section the mean-square analysis of the VSSDLMS algorithm is investigated. We take the weighted norm of (3.14) [56]-[57] and then applying the expectation operator on both sides of the equation. This yields the following:

$$\begin{aligned}
& \mathrm{E}\left[\|\tilde{\mathbf{w}}\left(i+1\right)\|_{\mathbf{\Sigma}}^{2}\right] \\
= \; & \mathrm{E}\left[\left\|\mathbf{G}\left(\mathbf{I}_{MN}-\mathbf{D}\left(i\right)\mathbf{U}^{T}\left(i\right)\mathbf{U}\left(i\right)\right)\tilde{\mathbf{w}}\left(i\right)-\mathbf{G}\mathbf{D}\left(i\right)\mathbf{U}^{T}\left(i\right)\mathbf{v}\left(i\right)\right\|_{\mathbf{\Sigma}}^{2}\right] \\
= \; & \mathrm{E}\left[\|\tilde{\mathbf{w}}\left(i\right)\|_{\mathbf{G}^{T}\mathbf{\Sigma}\mathbf{G}}^{2}\right]-\mathrm{E}\left[\|\tilde{\mathbf{w}}\left(i\right)\|_{\mathbf{G}^{T}\mathbf{\Sigma}\mathbf{Y}(i)\mathbf{U}(i)}^{2}\right] \\
& -\mathrm{E}\left[\|\tilde{\mathbf{w}}\left(i\right)\|_{\mathbf{U}^{T}(i)\mathbf{Y}^{T}(i)\mathbf{\Sigma}\mathbf{G}}^{2}\right]+\mathrm{E}\left[\|\tilde{\mathbf{w}}\left(i\right)\|_{\mathbf{U}^{T}(i)\mathbf{Y}^{T}(i)\mathbf{\Sigma}\mathbf{Y}(i)\mathbf{U}(i)}^{2}\right] \\
& +\mathrm{E}\left[\mathbf{v}^{T}\left(i\right)\mathbf{Y}^{T}\left(i\right)\mathbf{\Sigma}\mathbf{Y}\left(i\right)\mathbf{v}\left(i\right)\right] \\
= \; & \mathrm{E}\left[\|\tilde{\mathbf{w}}\left(i\right)\|_{\mathbf{\Sigma}'}^{2}\right]+\mathrm{E}\left[\mathbf{v}^{T}\left(i\right)\mathbf{Y}^{T}\left(i\right)\mathbf{\Sigma}\mathbf{Y}\left(i\right)\mathbf{v}\left(i\right)\right], \qquad (3.19)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{Y}\left(i\right) &= \mathbf{G}\mathbf{D}\left(i\right)\mathbf{U}^{T}\left(i\right) \qquad (3.20) \\
\mathbf{\Sigma}' &= \mathbf{G}^{T}\mathbf{\Sigma}\mathbf{G}-\mathbf{G}^{T}\mathbf{\Sigma}\mathbf{Y}\left(i\right)\mathbf{U}\left(i\right)-\mathbf{U}^{T}\left(i\right)\mathbf{Y}^{T}\left(i\right)\mathbf{\Sigma}\mathbf{G} \\
&\quad +\mathbf{U}^{T}\left(i\right)\mathbf{Y}^{T}\left(i\right)\mathbf{\Sigma}\mathbf{Y}\left(i\right)\mathbf{U}\left(i\right). \qquad (3.21)
\end{aligned}$$

Using the data independence assumption [57] and applying the expectation operator gives

$$
\begin{aligned}
\boldsymbol{\Sigma}' &= \mathbf{G}^T\boldsymbol{\Sigma}\mathbf{G} - \mathbf{G}^T\boldsymbol{\Sigma}\mathrm{E}\left[\mathbf{Y}\left(i\right)\mathbf{U}\left(i\right)\right] - \mathrm{E}\left[\mathbf{U}^T\left(i\right)\mathbf{Y}^T\left(i\right)\right]\boldsymbol{\Sigma}\mathbf{G} \\
&\quad + \mathrm{E}\left[\mathbf{U}^T\left(i\right)\mathbf{Y}^T\left(i\right)\boldsymbol{\Sigma}\mathbf{Y}\left(i\right)\mathbf{U}\left(i\right)\right] \\
&= \mathbf{G}^T\boldsymbol{\Sigma}\mathbf{G} - \mathbf{G}^T\boldsymbol{\Sigma}\mathbf{G}\mathrm{E}\left[\mathbf{D}\left(i\right)\right]\mathrm{E}\left[\mathbf{U}^T\left(i\right)\mathbf{U}\left(i\right)\right] \\
&\quad - \mathrm{E}\left[\mathbf{U}^T\left(i\right)\mathbf{U}\left(i\right)\right]\mathrm{E}\left[\mathbf{D}\left(i\right)\right]\mathbf{G}^T\boldsymbol{\Sigma}\mathbf{G} \\
&\quad + \mathrm{E}\left[\mathbf{U}^T\left(i\right)\mathbf{Y}^T\left(i\right)\boldsymbol{\Sigma}\mathbf{Y}\left(i\right)\mathbf{U}\left(i\right)\right]. \qquad (3.22)
\end{aligned}
$$

Gaussian Data

The evaluation of the expectation in the last term in (3.22) is very complex for non-Gaussian data. Therefore, it is assumed here that the data is Gaussian in order to evaluate (3.22). For Gaussian data, the auto-correlation matrix can be decomposed as $\mathbf{R}_{\mathbf{U}} = \mathbf{T}\boldsymbol{\Lambda}\mathbf{T}^T$, where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues for the entire network and $\mathbf{T}$ is a matrix containing the eigenvectors corresponding to these eigenvalues. Using this eigenvalue decomposition, we define the following relations

$$
\bar{\mathbf{w}}\left(i\right) = \mathbf{T}^T\tilde{\mathbf{w}}\left(i\right) \quad \bar{\mathbf{U}}\left(i\right) = \mathbf{U}\left(i\right)\mathbf{T} \qquad \bar{\mathbf{G}} = \mathbf{T}^T\mathbf{G}\mathbf{T}
$$

$$
\bar{\boldsymbol{\Sigma}} = \mathbf{T}^T\boldsymbol{\Sigma}\mathbf{T} \qquad \bar{\boldsymbol{\Sigma}}' = \mathbf{T}^T\boldsymbol{\Sigma}'\mathbf{T} \quad \bar{\mathbf{D}}\left(i\right) = \mathbf{T}^T\mathbf{D}\left(i\right)\mathbf{T} = \mathbf{D}\left(i\right).
$$

Using these relations (3.19) and (3.22) can be rewritten, respectively, as

$$
\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i+1\right)\right\|_{\bar{\boldsymbol{\Sigma}}}^2\right] = \mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i\right)\right\|_{\bar{\boldsymbol{\Sigma}}'}^2\right] + \mathrm{E}\left[\mathbf{v}^T\left(i\right)\bar{\mathbf{Y}}^T\left(i\right)\bar{\boldsymbol{\Sigma}}\bar{\mathbf{Y}}\left(i\right)\mathbf{v}\left(i\right)\right], \qquad (3.23)
$$

and

$$\begin{aligned}\bar{\boldsymbol{\Sigma}}' &= \bar{\mathbf{G}}^T\bar{\boldsymbol{\Sigma}}\bar{\mathbf{G}} - \bar{\mathbf{G}}^T\bar{\boldsymbol{\Sigma}}\bar{\mathbf{G}}\mathrm{E}\left[\mathbf{D}\left(i\right)\right]\mathrm{E}\left[\bar{\mathbf{U}}^T\left(i\right)\bar{\mathbf{U}}\left(i\right)\right] \\ &\quad -\mathrm{E}\left[\bar{\mathbf{U}}^T\left(i\right)\bar{\mathbf{U}}\left(i\right)\right]\mathrm{E}\left[\mathbf{D}\left(i\right)\right]\bar{\mathbf{G}}^T\bar{\boldsymbol{\Sigma}}\bar{\mathbf{G}} \\ &\quad +\mathrm{E}\left[\bar{\mathbf{U}}^T\left(i\right)\bar{\mathbf{Y}}\left(i\right)\bar{\boldsymbol{\Sigma}}\bar{\mathbf{Y}}\left(i\right)\bar{\mathbf{U}}\left(i\right)\right], \end{aligned} \tag{3.24}$$

where $\bar{\mathbf{Y}}\left(i\right) = \bar{\mathbf{G}}\mathbf{D}\left(i\right)\bar{\mathbf{U}}^T\left(i\right)$.

It can be seen that $\mathrm{E}\left[\bar{\mathbf{U}}^T\left(i\right)\bar{\mathbf{U}}\left(i\right)\right] = \boldsymbol{\Lambda}$. Also, using the *bvec* operator [58], we have $\bar{\sigma} = bvec\left\{\bar{\boldsymbol{\Sigma}}\right\}$. Now, let $\mathbf{R_v} = \boldsymbol{\Lambda_v} \odot \mathbf{I}_M$ denote the noise variance matrix for the entire network, where $\odot$ denotes the block Kronecker product [58]. Hence, the second term of the right-hand side of (3.23) is

$$\mathrm{E}\left[\mathbf{v}^T\left(i\right)\bar{\mathbf{Y}}^T\left(i\right)\bar{\boldsymbol{\Sigma}}\bar{\mathbf{Y}}\left(i\right)\mathbf{v}\left(i\right)\right] = \mathbf{b}^T\left(i\right)\bar{\sigma}, \tag{3.25}$$

where $\mathbf{b}\left(i\right) = bvec\left\{\mathbf{G}\mathbf{R_v}\mathrm{E}\left[\mathbf{D}^2\left(i\right)\right]\boldsymbol{\Lambda}\mathbf{G}^T\right\}$.

The fourth-order moment $\mathrm{E}\left[\bar{\mathbf{U}}^T\left(i\right)\bar{\mathbf{Y}}^T\left(i\right)\bar{\boldsymbol{\Sigma}}\bar{\mathbf{Y}}\left(i\right)\bar{\mathbf{U}}\left(i\right)\right]$ remains to be evaluated. Using the step-size independence assumption and the $\odot$ operator, we get

$$bvec\left\{\mathrm{E}\left[\bar{\mathbf{U}}^T\left(i\right)\bar{\mathbf{Y}}^T\left(i\right)\bar{\boldsymbol{\Sigma}}\bar{\mathbf{Y}}\left(i\right)\bar{\mathbf{U}}\left(i\right)\right]\right\} = \left(\mathrm{E}\left[\mathbf{D}\left(i\right)\odot\mathbf{D}\left(i\right)\right]\right)A\left(\mathbf{G}^T\odot\mathbf{G}^T\right)\bar{\sigma}, \tag{3.26}$$

where we have from [8]

$$\mathbf{A} = diag\left\{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_N\right\}, \tag{3.27}$$

and each matrix $\mathbf{A}_k$ is given by

$$\mathbf{A}_k = diag\left\{\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_k, ..., \lambda_k\lambda_k^T + 2\mathbf{\Lambda}_k \otimes \mathbf{\Lambda}_k, ..., \mathbf{\Lambda}_N \otimes \mathbf{\Lambda}_k\right\}. \tag{3.28}$$

The output of the matrix $\mathrm{E}\left[\mathbf{D}\left(i\right) \odot \mathbf{D}\left(i\right)\right]$ can be written as

$$\begin{aligned}
& \left(\mathrm{E}\left[\mathbf{D}\left(i\right) \odot \mathbf{D}\left(i\right)\right]\right)_{kk} \\
= \; & \mathrm{E}\left[diag\left\{\mu_k\left(i\right)\mathbf{I}_M \otimes \mu_1\left(i\right)\mathbf{I}_M, ..., \mu_k\left(i\right)\mathbf{I}_M \otimes \mu_k\left(i\right)\mathbf{I}_M,\right.\right. \\
& \left.\left. ..., \mu_k\left(i\right)\mathbf{I}_M \otimes \mu_N\left(i\right)\mathbf{I}_M\right\}\right] \\
= \; & \mathrm{E}\left[diag\left\{\mu_k\left(i\right)\mu_1\left(i\right)\mathbf{I}_{M^2}, ..., \mu_k^2\left(i\right)\mathbf{I}_{M^2}, ..., \mu_k\left(i\right)\mu_N\left(i\right)\mathbf{I}_{M^2}\right\}\right] \\
= \; & diag\left\{\mathrm{E}\left[\mu_k\left(i\right)\right]\mathrm{E}\left[\mu_1\left(i\right)\right]\mathbf{I}_{M^2}, ..., \mathrm{E}\left[\mu_k^2\left(i\right)\right]\mathbf{I}_{M^2},\right. \\
& \left. ..., \mathrm{E}\left[\mu_k\left(i\right)\right]\mathrm{E}\left[\mu_N\left(i\right)\right]\mathbf{I}_{M^2}\right\}
\end{aligned} \tag{3.29}$$

Now applying the *bvec* operator on the weighting matrix $\bar{\mathbf{\Sigma}}'$, we get

$$\begin{aligned}
bvec\left\{\bar{\mathbf{\Sigma}}'\right\} &= \bar{\sigma}' = [\mathbf{I}_{M^2N^2} - (\mathbf{I}_{MN} \odot \mathbf{\Lambda}\mathrm{E}\left[\mathbf{D}\left(i\right)\right]) - (\mathbf{\Lambda}\mathrm{E}\left[\mathbf{D}\left(i\right)\right] \odot \mathbf{I}_{MN}) \\
&\quad + (\mathrm{E}\left[\mathbf{D}\left(i\right) \odot \mathbf{D}\left(i\right)\right])\, A]\left(\mathbf{G}^T \odot \mathbf{G}^T\right)\bar{\sigma} \\
&= \mathbf{F}\left(i\right)\bar{\sigma},
\end{aligned} \tag{3.30}$$

where

$$\begin{aligned}
\mathbf{F}\left(i\right) &= [\mathbf{I}_{M^2N^2} - (\mathbf{I}_{MN} \odot \Lambda\mathrm{E}\left[\mathbf{D}\left(i\right)\right]) - (\Lambda\mathrm{E}\left[\mathbf{D}\left(i\right)\right] \odot \mathbf{I}_{MN}) \\
&\quad + (\mathrm{E}\left[\mathbf{D}\left(i\right) \odot \mathbf{D}\left(i\right)\right])\, A]\left(\mathbf{G}^T \odot \mathbf{G}^T\right).
\end{aligned} \tag{3.31}$$

Then (3.19) will look like the following:

$$\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i+1\right)\right\|_{\bar{\sigma}}^{2}\right]=\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i\right)\right\|_{\mathbf{F}(i)\bar{\sigma}}^{2}\right]+\mathbf{b}^{T}\left(i\right)\bar{\sigma}, \tag{3.32}$$

and hence, the transient behavior of the network is characterized by (4.40).

Learning Behavior

In this section, the learning behavior of the VSSDLMS algorithm is evaluated. Starting with $\bar{\mathbf{w}}_0=\mathbf{w}^{(o)}$ and $\mathbf{D}_0=\mu_0\mathbf{I}_{MN}$, we have for iteration $i+1$

$$\begin{aligned}
\mathcal{E}\left(i-1\right) &= diag\left\{\left(\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i-1\right)\right\|_{\lambda}^{2}\right]+\sigma_{v,1}^{2}\right)\mathbf{I}_{M},...,\left(\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i-1\right)\right\|_{\lambda}^{2}\right]+\sigma_{v,N}^{2}\right)\mathbf{I}_{M}\right\}\\
\mathrm{E}\left[\mathbf{D}\left(i\right)\right] &= \alpha\mathrm{E}\left[\mathbf{D}\left(i-1\right)\right]+\gamma\mathcal{E}\left(i-1\right)\\
\mathrm{E}\left[\mathbf{D}^{2}\left(i\right)\right] &= \alpha^{2}\mathrm{E}\left[\mathbf{D}^{2}\left(i-1\right)\right]+2\alpha\gamma\mathcal{E}\left(i-1\right)+\gamma^{2}\mathcal{E}^{2}\left(i-1\right)\\
\mathbf{F}\left(i\right) &= \left[\mathbf{I}_{M^{2}N^{2}}-\left(\mathbf{I}_{MN}\odot\Lambda\mathrm{E}\left[\mathbf{D}\left(i\right)\right]\right)-\left(\Lambda\mathrm{E}\left[\mathbf{D}\left(i\right)\right]\odot\mathbf{I}_{MN}\right)\right.\\
&\quad\left.+\left(\mathrm{E}\left[\mathbf{D}\left(i\right)\odot\mathbf{D}\left(i\right)\right]\right)A\right]\left(\mathbf{G}^{T}\odot\mathbf{G}^{T}\right)\\
\mathbf{b}\left(i\right) &= bvec\left\{\mathbf{G}\mathbf{R}_{\mathbf{v}}\mathrm{E}\left[\mathbf{D}^{2}\left(i\right)\right]\mathbf{\Lambda}\mathbf{G}^{T}\right\},
\end{aligned}$$

then incorporating the above relations in (4.40) gives

$$\begin{aligned}
\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i+1\right)\right\|_{\bar{\sigma}}^{2}\right] &= \mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i\right)\right\|_{\mathbf{F}(i)\bar{\sigma}}^{2}\right]+\mathbf{b}^{T}\left(i\right)\bar{\sigma}\\
&= \left\|\bar{\mathbf{w}}^{(o)}\right\|_{\left(\prod_{m=0}^{i}\mathbf{F}(m)\right)\bar{\sigma}}^{2}\\
&+ \left[\sum_{m=0}^{i-1}\mathbf{b}^{T}\left(m\right)\left(\prod_{n=m+1}^{i}\mathbf{F}\left(n\right)\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}\right]\bar{\sigma}.
\end{aligned} \tag{3.33}$$

Now, subtracting the results of iteration i from from those of iteration $i+1$ and simplifying we get

$$\begin{aligned} \mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i+1\right)\right\|_{\bar{\sigma}}^{2}\right] &= \mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i\right)\right\|_{\bar{\sigma}}^{2}\right] + \left\|\bar{\mathbf{w}}^{(o)}\right\|_{\mathbf{F}'(i)(\mathbf{F}(i)-\mathbf{I}_{MN})\bar{\sigma}}^{2} \\ &\quad + \left[\mathbf{F}''\left(i\right)\left(\mathbf{F}\left(i\right)-\mathbf{I}_{MN}\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}\right]\bar{\sigma}. \end{aligned} \tag{3.34}$$

where

$$\mathbf{F}'\left(i\right) = \prod_{m=0}^{i-1}\mathbf{F}\left(m\right), \tag{3.35}$$

$$\mathbf{F}''\left(i\right) = \sum_{m=0}^{i-2}\mathbf{b}^{T}\left(m\right)\left(\prod_{n=m+1}^{i-1}\mathbf{F}\left(n\right)\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}, \tag{3.36}$$

which can be defined iteratively as

$$\mathbf{F}'\left(i+1\right) = \mathbf{F}'\left(i\right)\mathbf{F}\left(i\right), \tag{3.37}$$

$$\mathbf{F}''\left(i+1\right) = \mathbf{F}''\left(i\right)\mathbf{F}\left(i\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}. \tag{3.38}$$

In order to evaluate the Mean-Square Deviation (MSD) and Excess Mean-Square Error (EMSE), we need to define the corresponding weighting matrix for each of them. Taking $\bar{\sigma} = (1/N)\,bvec\{\mathbf{I}_{MN}\} = \mathbf{q}_{\eta}$ and $\eta\left(i\right) = (1/N)\,\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i\right)\right\|^{2}\right]$ for the MSD we get

$$\eta\left(i\right) = \eta\left(i-1\right) + \left\|\bar{\mathbf{w}}^{(o)}\right\|_{\mathbf{F}'(i)(\mathbf{F}(i)-\mathbf{I}_{MN})\mathbf{q}_{\eta}}^{2} + \left[\mathbf{F}''\left(i\right)\left(\mathbf{F}\left(i\right)-\mathbf{I}_{MN}\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}\right]\mathbf{q}_{\eta}. \tag{3.39}$$

Similarly, taking $\bar{\sigma} = (1/N)\, bvec\{\mathbf{\Lambda}\} = \lambda_\zeta$ and $\zeta(i) = (1/N)\,\mathrm{E}\left[\|\bar{\mathbf{w}}(i)\|^2_{\mathbf{\Lambda}}\right]$, the EMSE behavior is governed by

$$\zeta(i) = \zeta(i-1) + \left\|\bar{\mathbf{w}}^{(o)}\right\|^2_{\mathbf{F}'(i)(\mathbf{F}(i)-\mathbf{I}_{MN})\lambda_\zeta} + \left[\mathbf{F}''(i)\left(\mathbf{F}(i)-\mathbf{I}_{MN}\right) + \mathbf{b}^T(i)\,\mathbf{I}_{MN}\right]\lambda_\zeta. \tag{3.40}$$

3.4.3 Steady-State Analysis

From (3.11), it is seen that the step-size for each node is independent of data from other nodes. Even though the connectivity matrix, $\mathbf{G}$, does not permit the weighting matrix, $\mathbf{F}(i)$, to be evaluated separately for each node, this is not the case for the step-size of any node. Therefore, taking the approach of [50], we first find the misadjustment, given by

$$\mathcal{M}_k = \frac{1 - \left[1 - 2\frac{(3-\alpha)\gamma\sigma^2_{v,k}}{1-\alpha^2}\mathrm{tr}\left(\mathbf{\Lambda_k}\right)\right]^{1/2}}{1 + \left[1 - 2\frac{(3-\alpha)\gamma\sigma^2_{v,k}}{1-\alpha^2}\mathrm{tr}\left(\mathbf{\Lambda_k}\right)\right]^{1/2}}, \tag{3.41}$$

which leads to the steady-state values for the step-size and its square for each node

$$\mu_{ss,k} = \frac{\gamma\sigma^2_{v,k}\left(1+\mathcal{M}_k\right)}{1-\alpha}, \tag{3.42}$$

$$\mu^2_{ss,k} = \frac{2\alpha\gamma\mu_{ss,k}\sigma^2_{v,k}\left(1+\mathcal{M}_k\right) + \gamma^2\sigma^4_{v,k}\left(1+\mathcal{M}_k\right)^2}{1-\alpha^2}. \tag{3.43}$$

Incorporating these steady-state relations in (4.41) to get the steady-state weighting matrix as

$$\begin{aligned}\mathbf{F}_{ss} &= [\mathbf{I}_{M^2N^2} - (\mathbf{I}_{MN} \odot \Lambda \mathrm{E}[\mathbf{D}_{ss}]) - (\Lambda \mathrm{E}[\mathbf{D}_{ss}] \odot \mathbf{I}_{MN}) \\ &\quad + (\mathrm{E}[\mathbf{D}_{ss} \odot \mathbf{D}_{ss}])\, A]\left(\mathbf{G}^T \odot \mathbf{G}^T\right), \end{aligned} \tag{3.44}$$

where $\mathbf{D}_{ss} = diag\{\mu_{ss,k}\mathbf{I}_M\}$.

Thus, the steady-state mean-square behavior is given by

$$\mathrm{E}\left[\|\bar{\mathbf{w}}_{ss}\|^2_{\bar{\sigma}}\right] = \mathrm{E}\left[\|\bar{\mathbf{w}}_{ss}\|^2_{\mathbf{F}_{ss}\bar{\sigma}}\right] + \mathbf{b}_{ss}^T\bar{\sigma}, \tag{3.45}$$

where $\mathbf{b}_{ss} = \mathbf{G}\mathbf{R}_{\mathbf{v}}\mathbf{D}_{ss}^2\mathbf{\Lambda}\mathbf{G}^T$ and $\mathbf{D}_{ss}^2 = diag\left\{\mu_{ss,k}^2\mathbf{I}_M\right\}$. Now solving (4.54), we get

$$\mathrm{E}\left[\|\bar{\mathbf{w}}_{ss}\|^2_{\bar{\sigma}}\right] = \mathbf{b}_{ss}^T\left[\mathbf{I}_{M^2N^2} - \mathbf{F}_{ss}\right]^{-1}\bar{\sigma}. \tag{3.46}$$

This equation gives the steady-state performance measure for the entire network. In order to solve for steady-state values of MSD and EMSE, we take $\bar{\sigma} = \mathbf{q}_\eta$ and $\bar{\sigma} = \lambda_\zeta$, respectively, as in (4.48) and (4.49). This gives us the steady-state values for MSD and EMSE as follows

$$\eta_{ss} = \mathbf{b}_{ss}^T\left[\mathbf{I}_{M^2N^2} - \mathbf{F}_{ss}\right]^{-1}\mathbf{q}_\eta, \tag{3.47}$$

$$\zeta_{ss} = \mathbf{b}_{ss}^T\left[\mathbf{I}_{M^2N^2} - \mathbf{F}_{ss}\right]^{-1}\lambda_\zeta. \tag{3.48}$$

3.5 Numerical Results

In this section, several simulation scenarios are considered and discussed to assess the performance of the proposed VSSDLMS algorithm. Results have been conducted for different average signal-to-noise ratio (SNR) values. The performance measure is the mean square deviation (MSD).

3.5.1 Comparison of the VSSLMS algorithms

First, the discussed VSSLMS algorithms are compared in a WSN environment and this comparison is reported in Fig. 3.1. As can be depicted from Fig. 3.1 the algorithm in [50] performs the best and therefore this algorithm is chosen for the proposed VSSDLMS algorithm.

3.5.2 Sensitivity Analysis

Now we perform a sensitivity analysis for the VSSDLMS algorithm. Since the VSSDLMS algorithm depends upon the choice of α and γ, these values are varied to check the performance of the algorithm. As can be seen from Fig. 3.2 the performance of the VSSDLMS algorithm degrades as α gets larger. Similarly, performance of the proposed algorithm improves as γ increases as depicted in Fig. 3.3. Based on this investigation, the choice of α and γ is made.

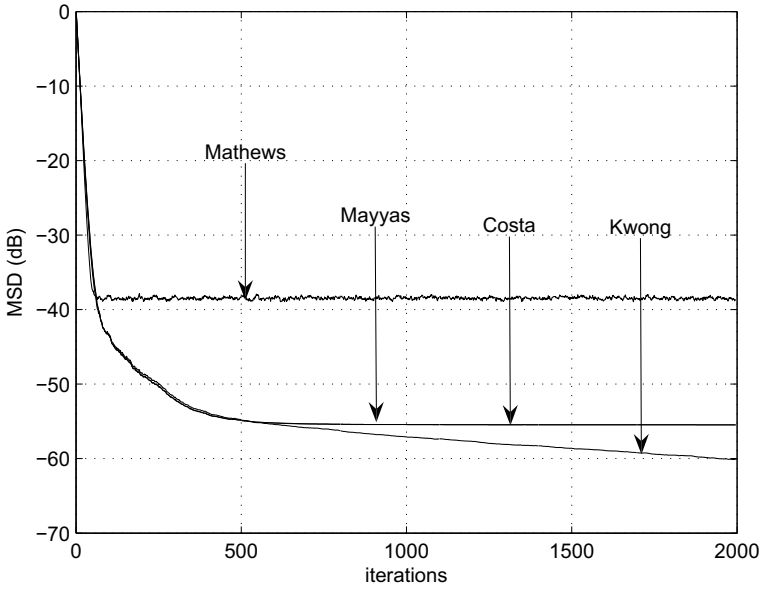

Figure 3.1: MSD for various VSSLMS algorithms applied to diffusion.

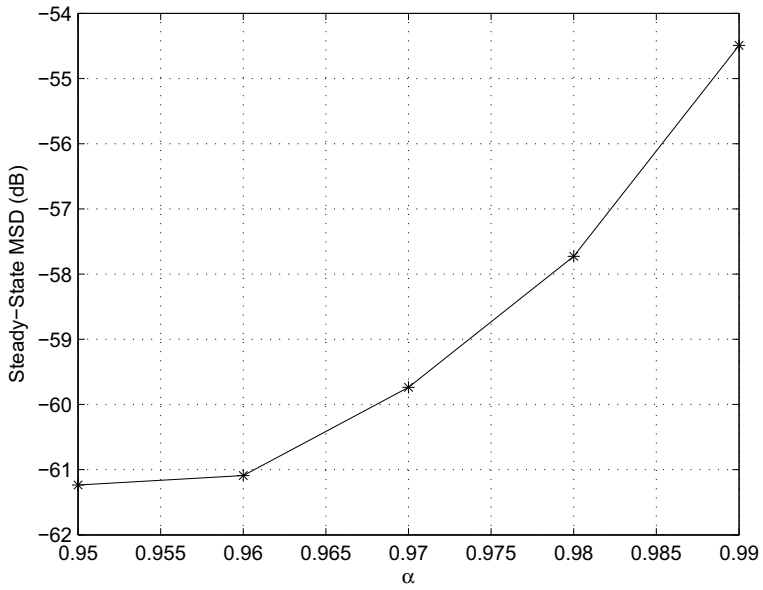

Figure 3.2: Steady-state MSD values for varying values of α.

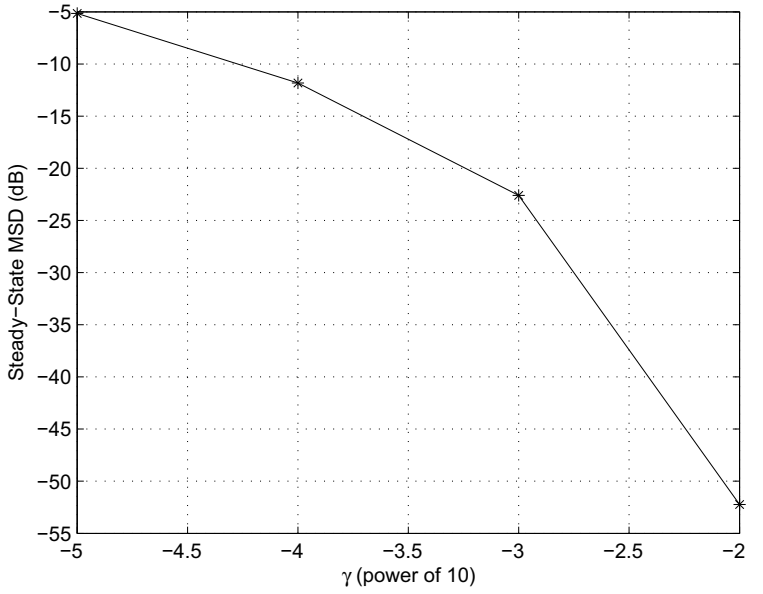

Figure 3.3: Steady-state MSD values for varying values of γ.

3.5.3 Performance of the proposed algorithm

Next, the proposed algorithm is compared with existing algorithms, which are the no cooperation case (NCLMS), the distributed LMS (DSLMS) [11], the DLMS [8], the DLMS with adaptive combiners (DLMSAC) [16] and the diffusion RLS (DRLS) [37]. The length of the unknown vector is taken as $M = 4$. The size of the network is $N = 20$. The sensors are randomly placed in an area of one unit square. The input regressor vector is assumed to be white Gaussian with auto-correlation matrix having the same variance for all nodes. Results are shown for two different values of SNR and communication range 0.3. Figure 3.4 reports the performance behavior of the different algorithms at an SNR of 10 dB. As can be seen from this figure the performance of the proposed VSSDLMS algorithm comes after that of the DRLS algorithm. The improvement in performance of the VSSDLMS algorithm is more for an SNR of 20 dB as depicted in Fig. 3.5. In both of these figures, when compared with other algorithms of similar complexity, the improvement in performance of the VSSDLMS algorithm is very significant. Similar performance for the steady-state behavior is obtained by the proposed VSSDLMS algorithm at SNR of 10 and 20 dB as shown, respectively, in Fig. 3.6 and Fig. 3.7. The DRLS algorithm performs better as expected but the proposed algorithm is clearly better than the remaining algorithms, both in convergence speed as well as steady-state error. Also, diffusion results in effecting the step-size variation of neighboring nodes and as a result the steady-state MSD for all nodes is nearly the same for all cases. This is in contrast with other algorithms for which

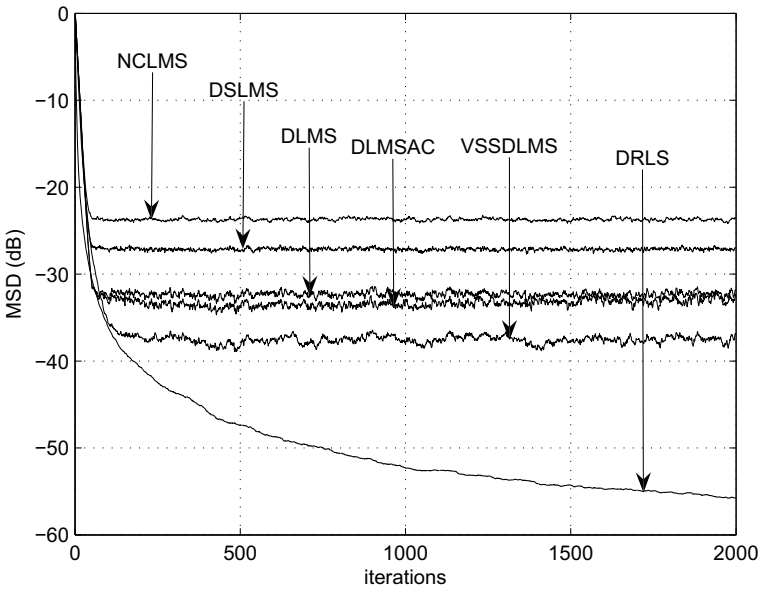

Figure 3.4: MSD for 20 nodes at SNR 10 dB.

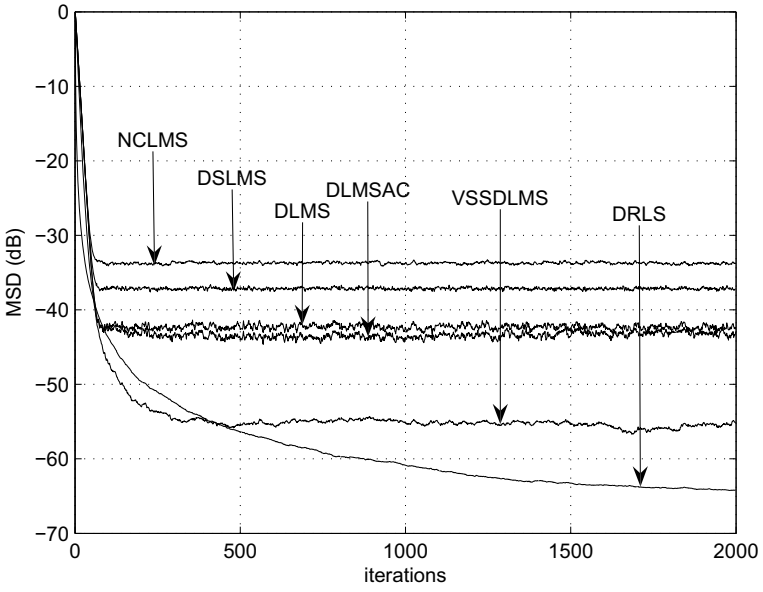

Figure 3.5: MSD for 20 nodes at SNR 20 dB.

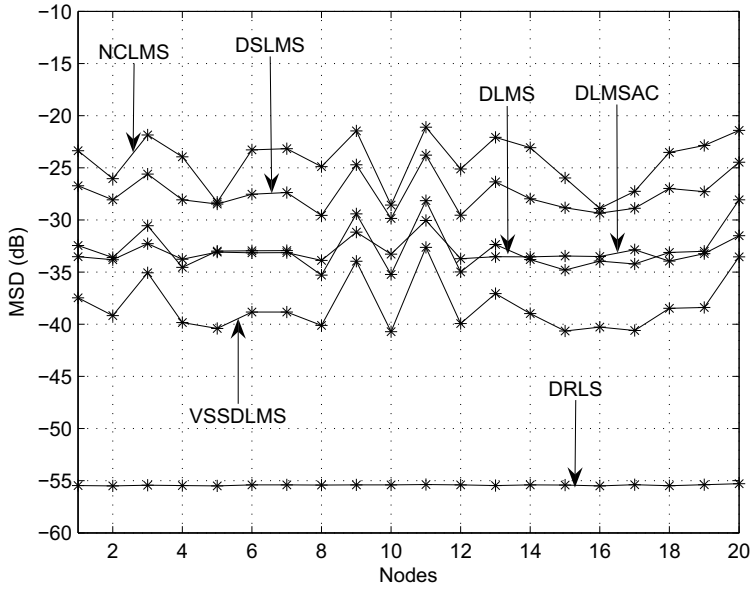

Figure 3.6: MSD at steady-state for 20 nodes at SNR 10 dB.

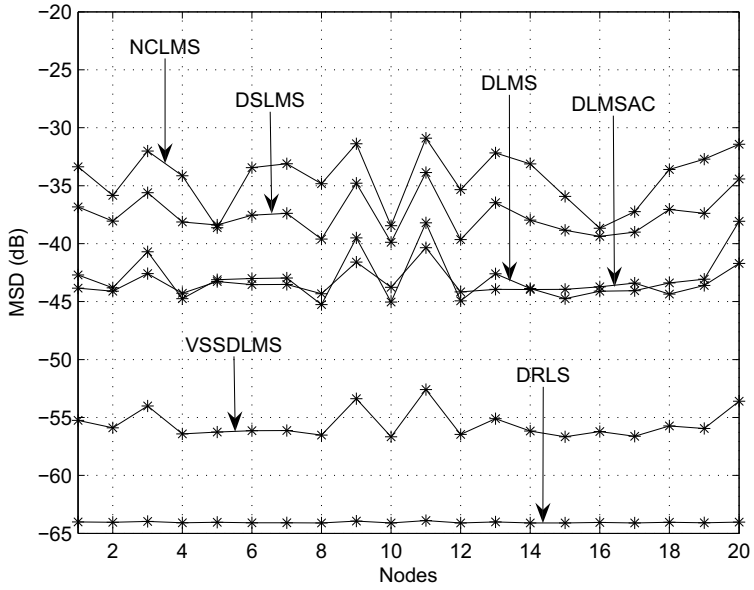

Figure 3.7: MSD at steady-state for 20 nodes at SNR 20 dB.

the steady-state MSD is effected by the SNR at each node, even when the SNR is high.

3.5.4 Theoretical analysis results

Next, the theoretical analysis of the proposed algorithm as compared to the simulation results is reported in Figs. 3.8 and 3.9. As can be seen from these figures, the simulation analysis are corroborating the theoretical findings very well. This is done for a network of 15 nodes with $M = 2$ and range 0.35. Two values for α are chosen, $\alpha = 0.95$ and $\alpha = 0.995$ whereas $\gamma = 0.001$.

3.5.5 Effect of network size

The effect on the performance of the proposed algorithm when the size of the network varies is reported in Figs. 3.10 and 3.11. As can be seen, an increase in network size improves performance. The trend is almost linear so it is safe to assume that performance improves linearly with increase in network size. However, when the number of nodes increase, the connectivity of the network also increases and so the cost of communication and computations also increases. Therefore, the size of the network has to be moderately large for an adequate performance, depending on the application. Furthermore, the trends shown in Figs. 3.10 and 3.11 show a vast improvement in performance over the previous algorithms.

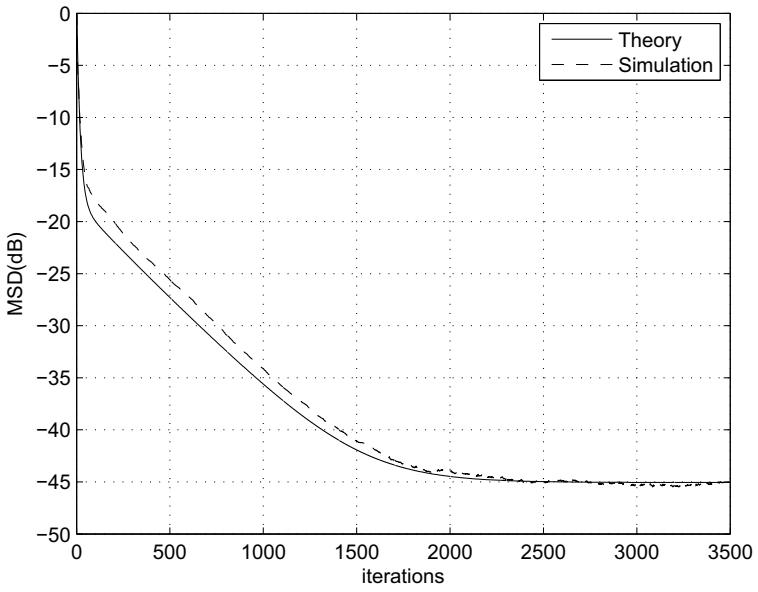

Figure 3.8: MSD for theory and simulation with $\alpha = 0.95$ and $\gamma = 0.001$.

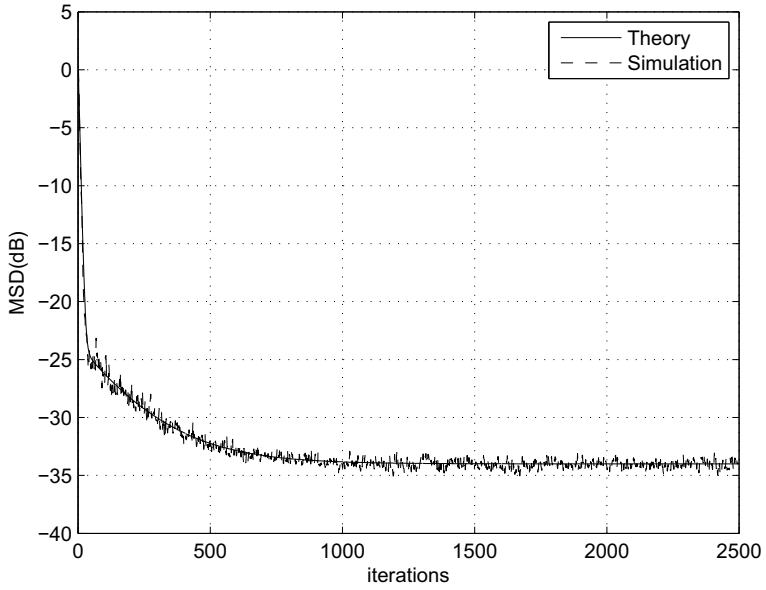

Figure 3.9: MSD for theory and simulation with $\alpha = 0.995$ and $\gamma = 0.001$.

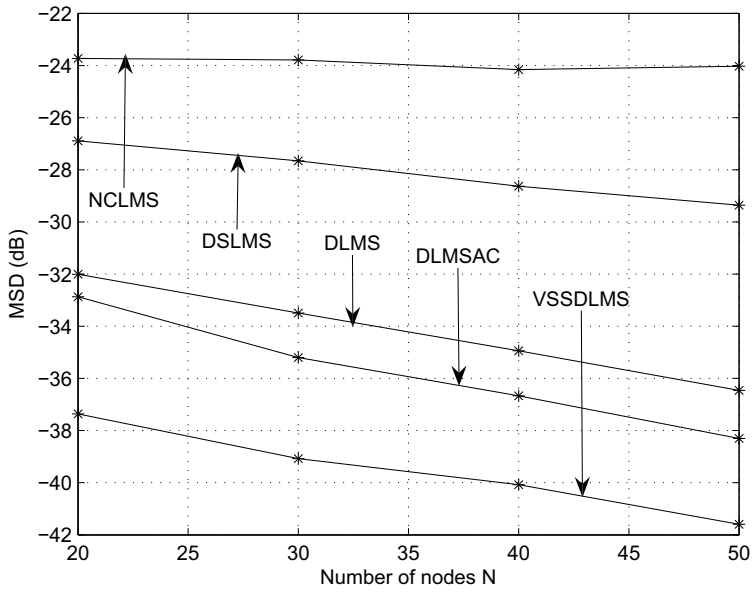

Figure 3.10: Steady-state MSD for varying N at SNR 10 dB.

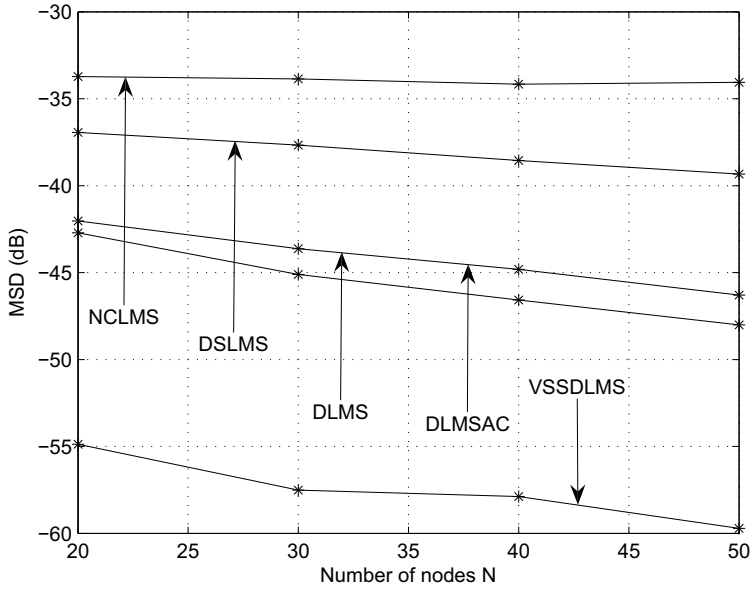

Figure 3.11: Steady-state MSD for varying N at SNR 20 dB.

3.5.6 Effect of node malfunction

An important aspect of working with sensor nodes is the possibility of a node switching off. In such a case the network may be required to adapt itself. The diffusion scheme is robust to such a change and this scenario has been considered here and results are shown in Figs. 3.12 and 3.13. A network of 50 nodes is chosen so that enough nodes can be switched off in order to study the performance of the proposed algorithm in this scenario. Two cases are considered, one where 15 nodes are switched off and another where 30 nodes are switched off. Results are shown in Figs. 3.12 and 3.13 for SNR of 10 dB and 20 dB, respectively. The nodes to be switched off are chosen at random. It is clear that malfunctioning of the nodes does effect the performance of the network. However, even with more than half the nodes switched off, the network still performs very well as the remaining network remains intact. The degradation would be slightly more severe if the malfunctioning nodes are those with most neighbors as that would reduce cooperation significantly. However, the network is still able to perform by adjusting itself to the change.

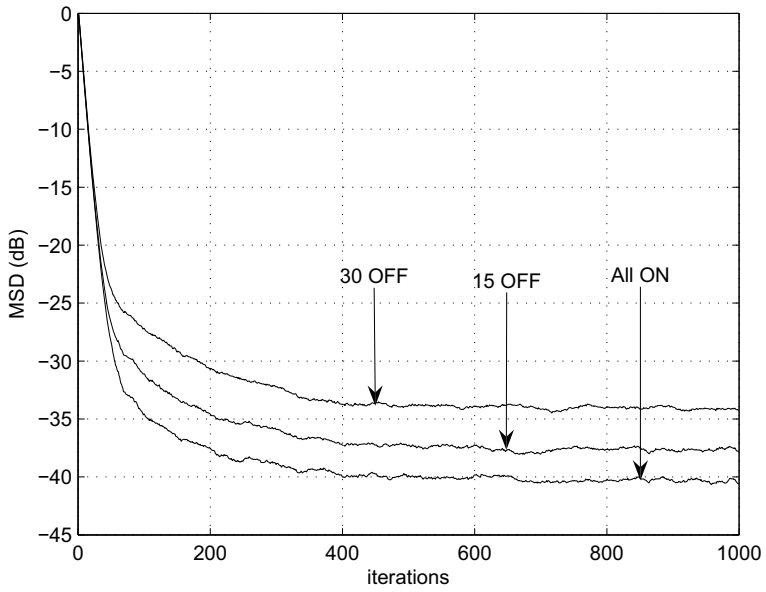

Figure 3.12: Node malfunction performance at SNR 10 dB.

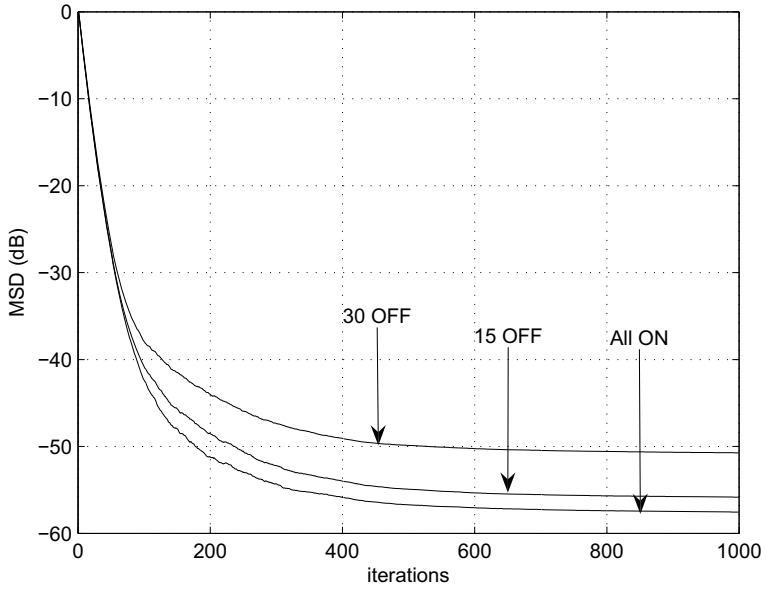

Figure 3.13: Node malfunction performance at SNR 20 dB.

3.5.7 Performance of generalized algorithm

It was shown earlier that the algorithm could be improved by sharing the complete data between neighbor nodes if the extra time delay and energy requirements can be compensated for. A comparison between the two versions of the proposed algorithm is shown in Figs. 3.14 and 3.15. As can be seen from the figures, the generalized algorithm performs much better in comparison with the originally proposed algorithm. The performance improves by almost 10 dB, which provides a reasonable trade-off between performance and cost for relatively small networks and the generalized solution can be considered in applications where cost effectiveness can be compensated for.

3.5.8 Steady-state performance

Finally, the comparison of the theoretical and simulated steady-state values for MSD and EMSE for two different input regressor auto-correlation matrices is given in Table 3.2. As can be seen from this table, close agreement between theory and simulations is observed.

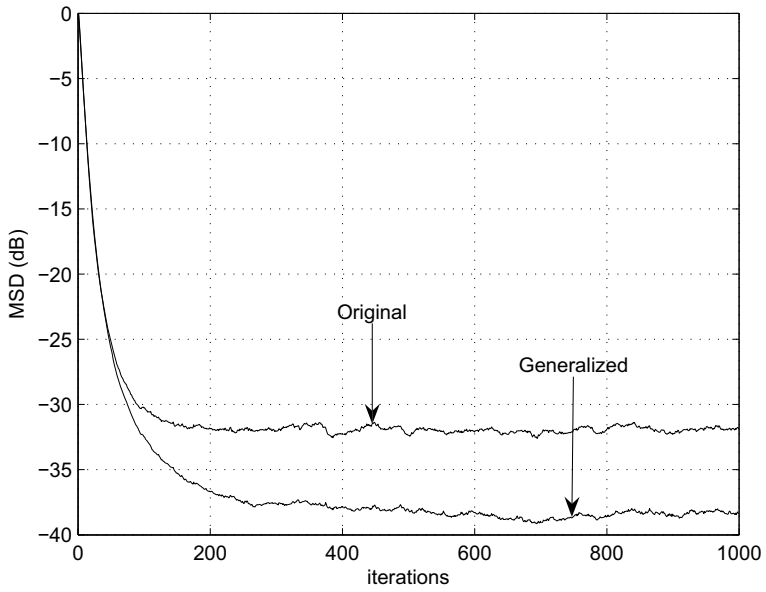

Figure 3.14: Proposed algorithms at SNR 10 dB.

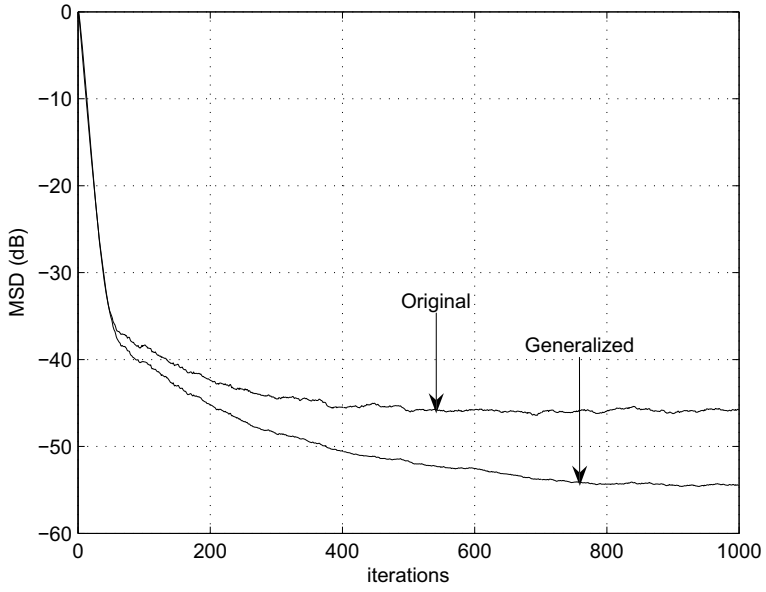

Figure 3.15: Proposed algorithms at SNR 20 dB.

$\mathbf{\Lambda}$	**MSD equation (4.56)**	**MSD simulations**	**EMSE equation (4.57)**	**EMSE simulations**
I_{MN}	-63.7800	-63.2838	-63.7800	-63.2814
$diag\left\{\sigma^2_{\mathbf{u},k}\mathrm{I}_M\right\}$	-63.3310	-63.5882	-58.4950	-58.8067

Table 3.2: Steady-state values for MSD and EMSE

3.6 Conclusion

The variable step-size diffusion LMS (VSSDLMS) algorithm is proposed in this chapter. Several popular VSSLMS algorithms are investigated. Based on performance, the algorithm chosen for the proposed VSSDLMS algorithm is that from [50]. Next, complete transient and steady state analyses are carried out. The independence assumption is used along with the Gaussian data assumption in order to find a closed form solution. Simulation results show that despite the use of assumptions, the theoretical results are corroborated by simulation results. A sensitivity analysis has been carried out to choose appropriate parameters that control how the step size should be varied. Simulation results show a comparison of the proposed algorithm with previous algorithms. It is found that the proposed algorithm shows remarkable improvement in performance and provides an excellent trade-off with computational cost. The proposed algorithm is then found to be robust even when several nodes are turned off. Finally, a steady-state comparison between theoretical and simulated results is tabulated and the results are found to corroborate each other.

CHAPTER 4

NOISE-CONSTRAINED DIFFUSION LEAST MEAN SQUARE ALGORITHM

4.1 Introduction

In the previous chapter a VSSLMS algorithm was directly incorporated into the diffusion scheme to come up with the VSSDLMS algorithm. Inspired by the work in [59] and motivated by the constraint-based approach used by [11], [12], the current chapter derives a new algorithm using the noise constraint. Here, first the derivation of the noise constraint based algorithm is given. Second, detailed convergence and steady-state analyses are carried out, including analyses for the case where there is mismatch in the noise variance estimate. Finally, extensive simulations are carried out to test the robustness of the proposed algorithm under

different scenarios, especially the mismatch scenario. Moreover, the simulation results are found to corroborate the theoretical findings very well.

4.2 The Proposed Algorithm

The global cost function for the entire network is given in Chapter 2 as

$$J(\mathbf{w}) = \sum_{k=1}^{N} \mathrm{E}\left[|d_k - \mathbf{u}_k \mathbf{w}|^2\right] \tag{4.1}$$

From (4.1), we can write the local cost function for each node k as

$$J_k(\mathbf{w}) = \mathrm{E}\left[|d_k - \mathbf{u}_k \mathbf{w}|^2\right], \tag{4.2}$$

where letting $\mathrm{E}\left[\mathbf{u}_k^* \mathbf{u}_k\right] = \mathbf{R}_{\mathbf{u},k}$ and solving gives

$$J_k(\mathbf{w}) = \|\mathbf{w} - \mathbf{w}_k\|^2_{\mathbf{R}_{\mathbf{u},k}} + \mathrm{MMSE}, \tag{4.3}$$

where MMSE represents the noise floor and does not include $\mathbf{w}$ and can, therefore, be ignored. Incorporating (4.3) into (4.1), the global cost function can be written as

$$\begin{aligned} J(\mathbf{w}) &= J_k(\mathbf{w}) + \sum_{l \neq k}^{N} J_l(\mathbf{w}) \\ &= \mathrm{E}\left[|d_k - \mathbf{u}_k \mathbf{w}|^2\right] + \sum_{l \neq k}^{N} \|\mathbf{w} - \mathbf{w}_l\|^2_{\mathbf{R}_{\mathbf{u},l}} \end{aligned} \tag{4.4}$$

This model assumes that any node k has access to data across the entire network. However, this is not a practical assumption as node k has access only to its neighbors. As a result, the cost function is approximated with data from neighbors being shared at each node. The resulting weighting matrix for the second term in (4.4) changes from $\mathbf{R}_{\mathbf{u},l}$ to a constant weighting factor b_{lk}, where the subscript lk denotes the connection between node k with its neighbor node l. The cost function, thus, becomes

$$\begin{aligned} J(\mathbf{w}) &= \mathrm{E}\left[|d_k - \mathbf{u}_k\mathbf{w}|^2\right] + \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}\left\|\mathbf{w}-\mathbf{w}_l\right\|^2 \\ &= J_k(\mathbf{w}) + \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}\left\|\mathbf{w}-\mathbf{w}_l\right\|^2. \end{aligned} \tag{4.5}$$

Assuming that the additive noise variance, $\sigma_{v,k}^2$, is known, the cost function can be modified using Lagrange multipliers as follows:

$$\begin{aligned} \min_{\mathbf{w}} J'(\mathbf{w}) &= J_k(\mathbf{w}) + \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}\left\|\mathbf{w}-\mathbf{w}_l\right\|^2 \\ &\quad +\gamma\beta_k\left(J_k(\mathbf{w}) - \sigma_{v,k}^2\right) - \gamma\beta_k^2, \end{aligned} \tag{4.6}$$

where the last term is a correction term added to avoid any spurious behavior. For node k, this cost function will be slightly modified as follows:

$$\begin{aligned} \min_{\mathbf{w}_k} J'(\mathbf{w}_k) &= J_k(\mathbf{w}_k) + \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}\left\|\mathbf{w}_k-\mathbf{w}_l\right\|^2 \\ &\quad +\gamma\beta_k\left(J_k(\mathbf{w}_k) - \sigma_{v,k}^2\right) - \gamma\beta_k^2, \end{aligned} \tag{4.7}$$

The solution for (4.7) can be obtained from the Robbins-Munro algorithm [60]

$$\mathbf{w}_k\left(i+1\right) = \mathbf{w}_k\left(i\right) - \mu_k \frac{\partial J'_k\left(\mathbf{w}_k\right)}{\partial \mathbf{w}_k}, \tag{4.8}$$

$$\beta_k\left(i+1\right) = \beta_k\left(i\right) + \alpha \frac{\partial J'_k\left(\mathbf{w}_k\right)}{\partial \beta_k}. \tag{4.9}$$

4.2.1 Steepest Descent Solution

Solution of the first partial derivative is given by

$$\begin{aligned} \frac{\partial J_k\left(\mathbf{w}_k\right)}{\partial \mathbf{w}_k} &= \left(1+\gamma\beta_k\right)\left(\mathbf{R}_{\mathbf{u},k}\mathbf{w}_k - \mathbf{r}_{d\mathbf{u},k}\right) \\ &+ \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}\left(\mathbf{w}_k - \mathbf{w_l}\right), \end{aligned} \tag{4.10}$$

where $\mathbf{R}_{\mathbf{u},k} = \mathrm{E}\left[\mathbf{u}_k^T\mathbf{u}_k\right]$ is the auto-correlation of the regressor vector $\mathbf{u}_k$ and $\mathbf{r}_{d\mathbf{u},k} = \mathrm{E}\left[d\mathbf{u}_k^T\right]$ is the cross-correlation between the regressor vector and the measured data.

Similarly, the solution of the 2nd partial derivative is

$$\frac{\partial J'_k\left(\mathbf{w}_k\right)}{\partial \beta_k} = \gamma\left(\mathrm{E}\left[\left|d_k - \mathbf{u}_k\mathbf{w}\right|^2\right] - \sigma^2_{v,k}\right) - 2\gamma\beta_k, \tag{4.11}$$

which results in

$$\beta_k\left(i+1\right) = \beta_k\left(i\right) + \alpha\gamma\left(\mathrm{E}\left[\left|d_k - \mathbf{u}_k\mathbf{w}\right|^2\right] - \sigma^2_{v,k}\right) - 2\alpha\gamma\beta_k\left(i\right). \tag{4.12}$$

If we replace $\alpha\gamma$ by $\alpha/2$ and then insert the solutions to the partial derivatives

into the algorithm, we get the resulting steepest descent solution

$$\begin{aligned}
\mathbf{w}_k(i+1) &= \mathbf{w}_k(i) + \mu_k(1+\gamma\beta_k(i))(\mathbf{R}_{\mathbf{u},k}\mathbf{w}_k(i) - \mathbf{R}_{d\mathbf{u},k}), \\
&+ \upsilon_k \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}(\mathbf{w}_l(i) - \mathbf{w}_k(i)) \qquad (4.13) \\
\beta_k(i+1) &= (1-\alpha)\beta_k(i) \\
&+ \frac{\alpha}{2}\left(\mathrm{E}\left[|d_k - \mathbf{u}_k\mathbf{w}|^2\right] - \sigma_{v,k}^2\right). \qquad (4.14)
\end{aligned}$$

Now, (4.13) can be written as a two-step process

$$\begin{aligned}
\mathbf{\Psi}_k(i+1) &= \mathbf{w}_k(i) + \mu_k(1+\gamma\beta_k(i)) \\
&\quad .(\mathbf{R}_{\mathbf{u},k}\mathbf{w}_k - \mathbf{R}_{d\mathbf{u},k}), \qquad (4.15) \\
\mathbf{w}_k(i+1) &= \mathbf{\Psi}_k(i+1) + \upsilon_k \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}(\mathbf{\Psi}_l(i+1) - \mathbf{\Psi}_k(i+1)) \\
&= \mathbf{\Psi}_k(i+1)(1-\upsilon_k + b_{kk}\upsilon_k) + \upsilon_k \sum_{\substack{l\in\mathcal{N}_k \\ l\neq k}} b_{lk}\mathbf{\Psi}_l(i+1) \\
&= \sum_{l\in N_k} c_{lk}\mathbf{\Psi}_l(i+1), \qquad (4.16)
\end{aligned}$$

where

$$c_{lk} = \begin{cases} 1-\upsilon_k + \upsilon_k b_{kk}, & l = k \\ \upsilon_k b_{lk}, & l \neq k \end{cases}$$

Combining (4.15), (4.16) with (4.14) results in the steepest descent solution to the noise-constrained diffusion problem.

4.2.2 Least Mean Square Solution

The steepest descent solution requires complete statistical knowledge of the data. For a practical adaptive solution, we simply replace $\mathbf{R}_{\mathbf{u},k}$, $\mathbf{R}_{d\mathbf{u},k}$ and $\mathrm{E}\left[\left|d_k - \mathbf{u}_k\mathbf{w}\right|^2\right]$ by their instantaneous values. Noting that $e_k(i) = d_k(i) - \mathbf{u}_k(i)\,\mathbf{w}_k(i)$, we get

$$\begin{aligned}
\mathbf{\Psi}_k(i+1) &= \mathbf{w}_k(i) + \mu_k\left(1+\gamma\beta_k(i)\right)\mathbf{u}_k^T(i)\,e_k(i), && (4.17)\\
\mathbf{w}_k(i+1) &= \sum_{l\in N_k} c_{lk}\mathbf{\Psi}_l(i+1), && (4.18)\\
\beta_k(i+1) &= (1-\alpha)\,\beta_k(i) + \frac{\alpha}{2}\left(e_k^2(i) - \sigma_{v,k}^2\right), && (4.19)
\end{aligned}$$

where

$$c_{lk} = \begin{cases} 1 - \upsilon_k + \upsilon_k b_{kk}, & l = k \\ \upsilon_k b_{lk}, & l \neq k \end{cases}$$

So equations (4.17)-(4.21) form the proposed Noise-Constrained Diffusion LMS (NCDLMS) algorithm [18].

4.2.3 Generalized solution

As in the previous chapter, a generalized algorithm can be derived using sharing of complete data. Following the same derivation steps as for the NCDLMS algorithm,

the generalized NCDLMS algorithm can be derived as

$$\begin{aligned}
\Psi_{k,gen}(i+1) &= \mathbf{w}_{k,gen}(i) + \mu_{k,gen}(1+\gamma\beta_{k,gen}(i)) \\
&\quad .\sum_{l\in\mathcal{N}_k} s_{lk}\mathbf{u}_l^T(i)(d_l(i) - \mathbf{u}_l(i)\mathbf{w}_{l,gen}(i)), \\
\mathbf{w}_{k,gen}(i+1) &= \sum_{l\in\mathcal{N}_k} c_{lk}\mathbf{\Psi}_{l,gen}(i+1) \\
\beta_{k,gen}(i+1) &= (1-\alpha)\beta_{k,gen}(i) + \frac{\alpha}{2}\left(e^2_{k,gen}(i) - \sigma^2_{v,k}\right),
\end{aligned} \tag{4.20}$$

where $e_{k,gen}(i) = \sum_{l\in\mathcal{N}_k} s_{lk}(d_l(i) - \mathbf{u}_l(i)\mathbf{w}_{l,gen}(i))$ and s_{lk} is the combiner weight for the sensor data being shared by node l with node k. As in the previous chapter, the generalized algorithm is considered as a special case and the cost comparison carried out in the previous chapter holds true for this case as well.

4.3 Performance Analysis

In order to perform the analysis, the whole network needs to be looked at because any node k is being affected by its neighbors and the neighbors in turn are affected by their respective neighbors. Therefore, we introduce new terms to study the performance of the network in a global manner. The local variables are transformed into global variables as in the previous chapter as follows:

$$\mathbf{w}(i) = col\{\mathbf{w}_1(i), ..., \mathbf{w}_N(i)\}, \quad \mathbf{\Psi}(i) = col\{\mathbf{\Psi}_1(i), ..., \mathbf{\Psi}_N(i)\},$$

$$\mathbf{U}(i) = diag\{\mathbf{u}_1(i), ..., \mathbf{u}_N(i)\}, \quad \mathbf{D} = diag\{\mu_1\mathbf{I}_M, ..., \mu_N\mathbf{I}_M\},$$

$$\mathbf{d}(i) = col\{d_1(i), ..., d_N(i)\}, \quad \mathbf{v}(i) = col\{v_1(i), ..., v_N(i)\}.$$

Following the same procedure as in the previous chapter, the set of equations for the entire networks is given as

$$\begin{aligned}
\boldsymbol{\Psi}(i+1) &= \mathbf{w}(i) + \mathbf{D}\left(\mathbf{I}_{MN} + \gamma\mathbf{B}(i)\right)\mathbf{U}^T(i)\left(\mathbf{d}(i) - \mathbf{U}(i)\,\mathbf{w}(i)\right), \quad (4.21)\\
\mathbf{w}(i+1) &= \mathbf{G}\boldsymbol{\Psi}(i+1), \quad (4.22)\\
\mathbf{B}(i+1) &= (1-\alpha)\,\mathbf{B}(i) + \frac{\alpha}{2}\left(\mathcal{E}(i) - \mathbf{S}\right), \quad (4.23)
\end{aligned}$$

where $\mathbf{G} = \mathbf{C} \otimes \mathbf{I}_M$, $\mathbf{C}$ is the $N \times N$ weighting matrix, $\otimes$ is the Block Kronecker operator, $\mathbf{B}_i = diag\left\{\beta_1\mathbf{I}_M, ..., \beta_N\mathbf{I}_M\right\}$ is the diagonal update matrix for the Lagrange multipliers, $\mathcal{E}(i) = diag\left\{e_1^2(i)\,\mathbf{I}_M, ..., e_N^2(i)\,\mathbf{I}_M\right\}$ is the diagonal matrix for instantaneous error and $\mathbf{S} = diag\left\{\sigma_1^2\mathbf{I}_M, ..., \sigma_N^2\mathbf{I}_M\right\}$ is the diagonal matrix containing the estimated noise variances for all nodes. Here it is assumed that the noise variances have been estimated exactly. The noise variance estimate mismatch will also be considered later.

4.3.1 Mean Analysis

As before, the global weight-error vector is given as

$$\tilde{\mathbf{w}}(i) = \mathbf{w}^{(o)} - \mathbf{w}(i) \quad (4.24)$$

Similarly, using $\mathbf{G}\mathbf{w}^{(\mathbf{o})} \triangleq \mathbf{w}^{(\mathbf{o})}$ in (4.21) and (4.22), we get

$$\begin{aligned}\tilde{\mathbf{w}}(i+1) &= \mathbf{G}\Psi(i+1) \\ &= \mathbf{G}\left(\mathbf{I}_{MN} - \mathcal{D}(i)\mathbf{U}^T(i)\mathbf{U}(i)\right)\tilde{\mathbf{w}}(i) - \mathbf{G}\mathcal{D}(i)\mathbf{U}^T(i)\mathbf{v}(i) \quad (4.25)\end{aligned}$$

Taking the expectation on both sides of the above equation gives

$$\mathrm{E}[\tilde{\mathbf{w}}(i+1)] = \mathbf{G}\left(\mathbf{I}_{MN} - \mathcal{D}(i)\mathbf{R}_{\mathbf{U}}\right)\mathrm{E}[\tilde{\mathbf{w}}(i)] \quad (4.26)$$

where $\mathbf{R}_{\mathbf{U}} = \mathrm{E}\left[\mathbf{U}^T\mathbf{U}\right]$ is the regressor auto-correlation matrix for the entire network and the matrix $\mathcal{D}(i) = \mathbf{D}\left(\mathbf{I}_{MN} + \gamma\mathbf{B}(i)\right)$ is assumed to be independent of the regressor matrix as it depends only on the values from the previous $i-1$ iterations. Also, since the measurement noise is spatially uncorrelated, the expectation of the second part of the right-hand side of equation (4.25) is zero.

From [15], we see that the diffusion algorithm is stable if the combination weights are restricted to within the unit circle. However, in this case stability is also dependent on the Lagrange multipliers. In this case, the algorithm will be stable if, for each node

$$\prod_{i=1}^{n}\left(\mathbf{I}_M - \mu_k\left(1 + \gamma\mathrm{E}[\beta(i)]\right)\mathbf{R}_{\mathbf{u},k}\right) \to 0, \quad as \;\; n \to \infty \quad (4.27)$$

which holds true if

$$0 < \mu_k < \frac{2}{\left(1 + \gamma \mathrm{E}\left[\beta\left(i\right)\right]\right) \lambda_{\max}\left(\mathbf{R}_{\mathbf{u},k}\right)}, \quad 1 \leq k \leq N, \tag{4.28}$$

where $\lambda_{\max}\left(\mathbf{R}_{\mathbf{u},k}\right)$ is the maximum eigenvalue of the auto-correlation matrix $\mathbf{R}_{\mathbf{u},k}$. The step-size limit is dependent on the convergence of the Lagrangian multiplier. The step-size can be kept under control with the proper selection of the parameters α and γ. If these parameters are chosen carefully, the multiplier value converges to a steady-state value which is equal to half that of the steady-state excess mean square error (EMSE). Since this value is small, the limit on the step-size can be safely approximated as follows:

$$0 < \mu_k < \frac{2}{\lambda_{\max}\left(\mathbf{R}_{\mathbf{u},k}\right)}. \tag{4.29}$$

Effect Of Noise Variance Estimate Mismatch

The previous analysis assumed perfect noise variance estimation. However, in a practical system it is not always possible to have an exact estimate and a mismatch can occur. The analysis in this case may be altered slightly in order to include the effect of the mismatch. The Lagrangian does not converge as in (4.28) because of

the mismatch. Taking the expectation of (4.21), we have

$$\begin{aligned}
\mathrm{E}\left[\beta_k\left(i+1\right)\right] &= \left(1-\alpha\right)\mathrm{E}\left[\beta_k\left(i\right)\right] + \frac{\alpha}{2}\mathrm{E}\left[e_k^2\left(i\right) - \hat{\sigma}_{v,k}^2\right] \\
&= \left(1-\alpha\right)\mathrm{E}\left[\beta_k\left(i\right)\right] + \frac{\alpha}{2}\left(EMSE\left(i\right) + \sigma_{v,k}^2 - \hat{\sigma}_{v,k}^2\right) \\
&= \left(1-\alpha\right)\mathrm{E}\left[\beta_{k,i-1}\right] + \frac{\alpha}{2}\left(EMSE\left(i\right) + \tilde{\sigma}_{v,k}^2\right), \qquad (4.30)
\end{aligned}$$

where $\hat{\sigma}_{v,k}^2$ is the imperfect estimate of the noise variance for node k, $\tilde{\sigma}_{v,k}^2$ is the noise variance mismatch and $EMSE$ is the excess mean square error. Thus, at steady-state the Lagrange multiplier becomes

$$\beta_{k,ss} = \frac{1}{2}\left(EMSE_{ss} + \tilde{\sigma}_{v,k}^2\right), \qquad (4.31)$$

which is simply a summation of the steady-state EMSE and the noise power mismatch. Since the value of EMSE is reasonably small, the bound on the step-size can be approximated by

$$0 < \mu_k < \frac{2}{\left(1 + \tilde{\sigma}_{v,k}^2/2\right)\lambda_{\max}\left(\mathbf{R}_{\mathbf{u},k}\right)}. \qquad (4.32)$$

In case of the zero noise-constrained algorithm (ZNCDLMS) there is no estimate for the noise power and so the limit can be approximated as

$$0 < \mu_k < \frac{2}{\left(1 + \sigma_{v,k}^2/2\right)\lambda_{\max}\left(\mathbf{R}_{\mathbf{u},k}\right)}, \qquad (4.33)$$

where the mismatch of the estimate, $\tilde{\sigma}_{v,k}^2$, is replaced by the actual noise power,

$\sigma^2_{v,k}$.

4.3.2 Mean-Square Analysis

As in the previous chapter, taking the weighted norm of (4.25) [56], [57] and applying the expectation operator yields

$$\mathrm{E}\left[\|\tilde{\mathbf{w}}\,(i+1)\|^2_{\Sigma}\right] = \mathrm{E}\left[\|\tilde{\mathbf{w}}\,(i)\|^2_{\Sigma'}\right] + \mathrm{E}\left[\mathbf{v}^T\,(i)\,\mathbf{L}^T\,(i)\,\Sigma\mathbf{L}\,(i)\,\mathbf{v}\,(i)\right], \tag{4.34}$$

where

$$\begin{aligned}
\mathbf{L}\,(i) &= \mathbf{G}\mathcal{D}\,(i)\,\mathbf{U}^T\,(i) && (4.35)\\
\Sigma' &= \mathbf{G}^T\Sigma\mathbf{G} - \mathbf{G}^T\Sigma\mathbf{L}\,(i)\,\mathbf{U}\,(i) - \mathbf{U}^T\,(i)\,\mathbf{L}^T\,(i)\,\Sigma\mathbf{G}\\
&\quad + \mathbf{U}^T\,(i)\,\mathbf{L}^T\,(i)\,\Sigma\mathbf{L}\,(i)\,\mathbf{U}\,(i)\,. && (4.36)
\end{aligned}$$

Using the data independence assumption [57] and applying the expectation operator directly on (4.36) we have

$$\begin{aligned}
\boldsymbol{\Sigma}' &= \mathbf{G}^T\boldsymbol{\Sigma}\mathbf{G} - \mathbf{G}^T\boldsymbol{\Sigma}\mathbf{G}\mathrm{E}\left[\mathcal{D}\,(i)\right]\mathrm{E}\left[\mathbf{U}^T\,(i)\,\mathbf{U}\,(i)\right]\\
&\quad - \mathrm{E}\left[\mathbf{U}^T\,(i)\,\mathbf{U}\,(i)\right]\mathrm{E}\left[\mathcal{D}\,(i)\right]\mathbf{G}^T\boldsymbol{\Sigma}\mathbf{G}\\
&\quad + \mathrm{E}\left[\mathbf{U}^T\,(i)\,\mathbf{L}^T(i)\boldsymbol{\Sigma}\mathbf{L}\,(i)\,\mathbf{U}\,(i)\right], && (4.37)
\end{aligned}$$

where $\mathrm{E}\left[\mathcal{D}\,(i)\right] = \mathbf{D}\left(\mathbf{I}_{MN} + \gamma\mathrm{E}\left[\mathbf{B}\,(i)\right]\right)$.

As can be seen, (4.34) and (4.37) are similar to (3.19) and (3.22) so the analysis

will follow a similar process as the previous chapter. Following the same process as before, the data is assumed to be Gaussian and (4.34) and (4.37) are transformed as follows:

$$\mathrm{E}\left[\|\bar{\mathbf{w}}(i+1)\|^2_{\bar{\mathbf{\Sigma}}}\right] = \mathrm{E}\left[\|\bar{\mathbf{w}}(i)\|^2_{\bar{\mathbf{\Sigma}}'}\right] + \mathrm{E}\left[\mathbf{v}^T(i)\bar{\mathbf{L}}^T(i)\bar{\mathbf{\Sigma}}\bar{\mathbf{L}}(i)\mathbf{v}(i)\right], \tag{4.38}$$

and

$$\begin{aligned}\bar{\mathbf{\Sigma}}' &= \bar{\mathbf{G}}^T\bar{\mathbf{\Sigma}}\bar{\mathbf{G}} - \bar{\mathbf{G}}^T\bar{\mathbf{\Sigma}}\bar{\mathbf{G}}\mathrm{E}\left[\mathcal{D}(i)\right]\mathrm{E}\left[\bar{\mathbf{U}}^T(i)\bar{\mathbf{U}}(i)\right] \\ &\quad -\mathrm{E}\left[\bar{\mathbf{U}}^T(i)\bar{\mathbf{U}}(i)\right]\mathrm{E}\left[\mathcal{D}(i)\right]\bar{\mathbf{G}}^T\bar{\mathbf{\Sigma}}\bar{\mathbf{G}} \\ &\quad +\mathrm{E}\left[\bar{\mathbf{U}}^T(i)\bar{\mathbf{L}}^T(i)\bar{\mathbf{\Sigma}}\bar{\mathbf{L}}(i)\bar{\mathbf{U}}(i)\right], \end{aligned} \tag{4.39}$$

where $\bar{\mathbf{L}}\left(i\right) = \bar{\mathbf{G}}\mathcal{D}\left(i\right)\bar{\mathbf{U}}^T\left(i\right)$.

Similarly, the expectations can be solved as before and the final solution is given as

$$\mathrm{E}\left[\|\bar{\mathbf{w}}\left(i+1\right)\|^2_{\bar{\sigma}}\right] = \mathrm{E}\left[\|\bar{\mathbf{w}}\left(i\right)\|^2_{\mathbf{F}(i)\bar{\sigma}}\right] + \mathbf{b}^T\left(i\right)\bar{\sigma}, \tag{4.40}$$

where $\mathbf{b}\left(i\right) = bvec\left\{\mathbf{G}\mathbf{R}_{\mathbf{v}}\mathrm{E}\left[\mathcal{D}^2\left(i\right)\right]\mathbf{\Lambda}\mathbf{G}^T\right\}$ and

$$\begin{aligned}\mathbf{F}\left(i\right) &= \left[\mathbf{I}_{M^2N^2} - \left(\mathbf{I}_{MN}\odot\Lambda\mathrm{E}\left[\mathcal{D}\left(i\right)\right]\right) - \left(\Lambda\mathrm{E}\left[\mathcal{D}\left(i\right)\right]\odot\mathbf{I}_{MN}\right)\right. \\ &\quad \left. + \left(\mathrm{E}\left[\mathcal{D}\left(i\right)\odot\mathcal{D}\left(i\right)\right]\right)\mathbf{A}\right]\left(\mathbf{G}^T\odot\mathbf{G}^T\right). \end{aligned} \tag{4.41}$$

and hence, the transient behavior of the network is characterized by (4.40).

Learning Behavior

In this section, the learning behavior of the NCDLMS algorithm is evaluated. Starting with $\bar{\mathbf{w}}_0 = \mathbf{w}^{(o)}$ and $\mathcal{D}_0 = \mu_0 \mathbf{I}_{MN}$, we have for iteration $i+1$

$$
\begin{aligned}
\mathcal{E}(i-1) &= diag\left\{\left(\mathrm{E}\left[\left\|\bar{\mathbf{w}}(i-1)\right\|_{\lambda}^{2}\right] + \tilde{\sigma}_{v,1}^{2}\right)\mathbf{I}_M, ..., \left(\mathrm{E}\left[\left\|\bar{\mathbf{w}}(i-1)\right\|_{\lambda}^{2}\right] + \tilde{\sigma}_{v,N}^{2}\right)\mathbf{I}_M\right\} \\
&= diag\left\{\left(\mathrm{E}\left[\left\|\bar{\mathbf{w}}(i-1)\right\|_{\lambda}^{2}\right] + (1-a)\,\sigma_{v,1}^{2}\right)\mathbf{I}_M, \right. \\
&\quad \left. ..., \left(\mathrm{E}\left[\left\|\bar{\mathbf{w}}(i-1)\right\|_{\lambda}^{2}\right] + (1-a)\sigma_{v,N}^{2}\right)\mathbf{I}_M\right\} \\
\mathrm{E}\left[\mathbf{B}(i)\right] &= (1-\alpha)\,\mathrm{E}\left[\mathbf{B}(i-1)\right] + \frac{\alpha}{2}\mathcal{E}(i-1) \\
\mathrm{E}\left[\mathbf{B}^2(i)\right] &= (1-\alpha)^2\,\mathrm{E}\left[\mathbf{B}^2(i-1)\right] + \alpha\,(1-\alpha)\,\mathrm{E}\left[\mathbf{B}(i-1)\right]\mathcal{E}(i-1) + \frac{\alpha^2}{4}\left(\mathcal{E}(i-1)\right)^2 \\
\mathrm{E}\left[\mathcal{D}(i)\right] &= \mathbf{D}\left(\mathbf{I}_{MN} + \gamma \mathrm{E}\left[\mathbf{B}(i)\right]\right) \\
\mathrm{E}\left[\mathcal{D}^2(i)\right] &= \mathbf{D}^2\left(\mathbf{I}_{MN} + 2\gamma \mathrm{E}\left[\mathbf{B}(i)\right] + \gamma^2 \mathrm{E}\left[\mathbf{B}^2(i)\right]\right) \\
\mathbf{F}(i) &= \left[\mathbf{I}_{M^2N^2} - \left(\mathbf{I}_{MN} \odot \Lambda \mathrm{E}\left[\mathcal{D}(i)\right]\right) - \left(\Lambda \mathrm{E}\left[\mathcal{D}(i)\right] \odot \mathbf{I}_{MN}\right)\right. \\
&\quad \left. + \left(\mathrm{E}\left[\mathcal{D}(i) \odot \mathcal{D}(i)\right]\right)\mathbf{A}\right]\left(\mathbf{G}^T \odot \mathbf{G}^T\right) \\
\mathbf{b}(i) &= bvec\left\{\mathbf{G}\mathbf{R}_{\mathbf{v}}\mathrm{E}\left[\mathcal{D}^2(i)\right]\mathbf{\Lambda}\mathbf{G}^T\right\},
\end{aligned}
$$

where $a = \hat{\sigma}_{v,k}^2 / \sigma_{v,k}^2$ is the ratio between the estimated and actual noise power at node k. For a perfect estimate, this would result in $a = 1$, for a mismatch, $0 < a < 1$ and for the case of ZNCDLMS, $a = 0$. Incorporating the above relations

in (4.40) gives

$$\begin{aligned}
\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i+1\right)\right\|_{\bar{\sigma}}^{2}\right] &= \mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i\right)\right\|_{\mathbf{F}(i)\bar{\sigma}}^{2}\right]+\mathbf{b}^{T}\left(i\right)\bar{\sigma} \\
&= \left\|\bar{\mathbf{w}}^{(o)}\right\|_{\left(\prod_{m=0}^{i}\mathbf{F}(m)\right)\bar{\sigma}}^{2} \\
&\quad+\left[\sum_{m=0}^{i-1}\mathbf{b}^{T}\left(m\right)\left(\prod_{n=m+1}^{i}\mathbf{F}\left(n\right)\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}\right]\bar{\sigma}. \quad (4.42)
\end{aligned}$$

Now, subtracting the results of iteration i from from those of iteration $i+1$ and simplifying we get

$$\begin{aligned}
\mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i+1\right)\right\|_{\bar{\sigma}}^{2}\right] &= \mathrm{E}\left[\left\|\bar{\mathbf{w}}\left(i\right)\right\|_{\bar{\sigma}}^{2}\right]+\left\|\bar{\mathbf{w}}^{(o)}\right\|_{\mathbf{F}'(i)(\mathbf{F}(i)-\mathbf{I}_{MN})\bar{\sigma}}^{2} \\
&\quad+\left[\mathbf{F}''\left(i\right)\left(\mathbf{F}\left(i\right)-\mathbf{I}_{MN}\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}\right]\bar{\sigma}. \quad (4.43)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{F}'\left(i\right) &= \prod_{m=0}^{i-1}\mathbf{F}\left(m\right), & (4.44)\\
\mathbf{F}''\left(i\right) &= \sum_{m=0}^{i-2}\mathbf{b}^{T}\left(m\right)\left(\prod_{n=m+1}^{i-1}\mathbf{F}\left(n\right)\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}, & (4.45)
\end{aligned}$$

which can be defined iteratively as

$$\begin{aligned}
\mathbf{F}'\left(i+1\right) &= \mathbf{F}'\left(i\right)\mathbf{F}\left(i\right), & (4.46)\\
\mathbf{F}''\left(i+1\right) &= \mathbf{F}''\left(i\right)\mathbf{F}\left(i\right)+\mathbf{b}^{T}\left(i\right)\mathbf{I}_{MN}. & (4.47)
\end{aligned}$$

In order to evaluate the Mean-Square Deviation (MSD) and Excess Mean-

Square Error (EMSE), we need to define the corresponding weighting matrix for each of them. Taking $\bar{\sigma} = (1/N)\, bvec\{\mathbf{I}_{MN}\} = \mathbf{q}_\eta$ and $\eta(i) = (1/N)\,\mathrm{E}\left[\|\bar{\mathbf{w}}(i)\|^2\right]$ for the MSD we get

$$\eta(i) = \eta(i-1) + \left\|\bar{\mathbf{w}}^{(o)}\right\|^2_{\mathbf{F}'(i)(\mathbf{F}(i)-\mathbf{I}_{MN})\mathbf{q}_\eta} + \left[\mathbf{F}''(i)\left(\mathbf{F}(i)-\mathbf{I}_{MN}\right) + \mathbf{b}^T(i)\,\mathbf{I}_{MN}\right]\mathbf{q}_\eta. \tag{4.48}$$

Similarly, taking $\bar{\sigma} = (1/N)\, bvec\{\mathbf{\Lambda}\} = \lambda_\zeta$ and $\zeta(i) = (1/N)\,\mathrm{E}\left[\|\bar{\mathbf{w}}(i)\|^2_{\mathbf{\Lambda}}\right]$, the EMSE behavior is governed by

$$\zeta(i) = \zeta(i-1) + \left\|\bar{\mathbf{w}}^{(o)}\right\|^2_{\mathbf{F}'(i)(\mathbf{F}(i)-\mathbf{I}_{MN})\lambda_\zeta} + \left[\mathbf{F}''(i)\left(\mathbf{F}(i)-\mathbf{I}_{MN}\right) + \mathbf{b}^T(i)\,\mathbf{I}_{MN}\right]\lambda_\zeta. \tag{4.49}$$

4.3.3 Steady-State Analysis

From (4.23), it is seen that the Lagrangian multiplier update for each node is independent of data from other nodes. Even though the connectivity matrix, $\mathbf{G}$, does not permit the weighting matrix, $\mathbf{F}(i)$, to be evaluated separately for each node, this is not the case for the step-size of any node. Therefore, taking the approach of [59], we first find the misadjustment, given by

$$\mathcal{M}_k = \frac{\mu_k Tr\{\mathbf{R}_{\mathbf{u},k}\}}{2}\left(1 + \frac{\gamma\sigma^2_{v,k}(1-a)}{2} + \frac{\gamma^2\alpha\sigma^2_{v,k}}{2(2-\alpha)\left(1+\gamma\sigma^2_{v,k}(1-a)/2\right)}\right), \tag{4.50}$$

which leads to the steady-state values for the Lagrange multiplier update and its square for each node

$$\begin{aligned}
\beta_{k,ss} &= \frac{1}{2}\left(EMSE_{ss} + (1-a)\sigma_{v,k}^2\right) \\
&= \frac{1}{2}\left(\mathcal{M}_k\sigma_{v,k}^2 + (1-a)\,\sigma_{v,k}^2\right) \\
&= \frac{\sigma_{v,k}^2}{2}\left(\mathcal{M}_k + 1 - a\right), \qquad (4.51) \\
\beta_{k,ss}^2 &= \frac{\alpha}{(1-\alpha)}\beta_{k,ss}\sigma_{v,k}^2\left(\mathcal{M}_k + 1 - a\right) + \frac{\sigma_{v,k}^4}{4\,(1-\alpha)^2}\left(\mathcal{M}_k + 1 - a\right)^2. \qquad (4.52)
\end{aligned}$$

Incorporating these relations in (4.41) to get the steady-state weighting matrix as

$$\begin{aligned}
\mathbf{F}_{ss} &= \left[\mathbf{I}_{M^2N^2} - \left(\mathbf{I}_{MN} \odot \Lambda \mathrm{E}\left[\mathcal{D}_{ss}\right]\right) - \left(\Lambda \mathrm{E}\left[\mathcal{D}_{ss}\right] \odot \mathbf{I}_{MN}\right)\right. \\
&\quad \left. + \left(\mathrm{E}\left[\mathcal{D}_{ss} \odot \mathcal{D}_{ss}\right]\right)\mathbf{A}\right]\left(\mathbf{G}^T \odot \mathbf{G}^T\right), \qquad (4.53)
\end{aligned}$$

where $\mathcal{D}_{ss} = diag\left\{\mu_k(1+\gamma\beta_{k,ss})\mathbf{I}_M\right\}$.

Thus, the steady-state mean-square behavior is given by

$$\mathrm{E}\left[\|\bar{\mathbf{w}}_{ss}\|_{\bar{\sigma}}^2\right] = \mathrm{E}\left[\|\bar{\mathbf{w}}_{ss}\|_{\mathbf{F}_{ss}\bar{\sigma}}^2\right] + \mathbf{b}_{ss}^T\bar{\sigma}, \qquad (4.54)$$

where $\mathbf{b}_{ss} = \mathbf{R}_{\mathbf{v}}\mathcal{D}_{ss}^2\mathbf{\Lambda}$ and $\mathcal{D}_{ss}^2 = diag\left\{\mu_k^2(1+\gamma\beta_{k,ss})^2\mathbf{I}_M\right\}$. Now solving (4.54), we get

$$\mathrm{E}\left[\|\bar{\mathbf{w}}_{ss}\|_{\bar{\sigma}}^2\right] = \mathbf{b}_{ss}^T\left[\mathbf{I}_{M^2N^2} - \mathbf{F}_{ss}\right]^{-1}\bar{\sigma}. \qquad (4.55)$$

This equation gives the steady-state performance measure for the entire network. In order to solve for steady-state values of MSD and EMSE, we take $\bar{\sigma} = \mathbf{q}_\eta$

and $\bar{\sigma} = \lambda_\zeta$, respectively, as in (4.48) and (4.49). This gives us the steady-state values for MSD and EMSE as follows

$$\eta_{ss} = \mathbf{b}_{ss}^T \left[\mathbf{I}_{M^2N^2} - \mathbf{F}_{ss}\right]^{-1} \mathbf{q}_\eta, \tag{4.56}$$

$$\zeta_{ss} = \mathbf{b}_{ss}^T \left[\mathbf{I}_{M^2N^2} - \mathbf{F}_{ss}\right]^{-1} \lambda_\zeta. \tag{4.57}$$

4.4 Numerical Results

In this section, several simulation scenarios are considered and discussed to assess the performance of the proposed NCDLMS algorithm. Results have been conducted for different average signal-to-noise ratio (SNR) values. The performance measure is the mean square deviation (MSD).

4.4.1 Performance of the proposed algorithm

First, the proposed algorithm is compared with existing algorithms, which are the no cooperation case (NCLMS), the distributed LMS (DSLMS) [11], the DLMS [8], the DLMS with adaptive combiners (DLMSAC) [16], the VSSDLMS algorithm [17] and the diffusion RLS (DRLS) [37]. The length of the unknown vector is taken as $M = 4$. The size of the network is $N = 20$. The sensors are randomly placed in an area of one unit square. The input regressor vector is assumed to be white Gaussian with auto-correlation matrix having the same variance for all nodes. Results are shown for two different values of SNR and communication range 0.3. Figure 4.1 reports the performance behavior of the different algorithms at an

SNR of 10 dB. As can be seen from this figure the performance of the proposed NCDLMS algorithm comes after that of the DRLS algorithm. The performance of the NCDLMS algorithm improves better for an SNR of 20 dB as depicted in Fig. 4.2. In both of these figures, when compared with other algorithms of similar complexity, the improvement in performance of the NCDLMS algorithm is very significant. Similar performance for the steady-state behavior is obtained by the proposed NCDLMS algorithm at SNR of 10 and 20 dB as shown, respectively, in Fig. 4.3 and Fig. 4.4. The DRLS algorithm performs better as expected but the proposed algorithm is clearly better than the remaining algorithms, both in convergence speed as well as steady-state error. Also, since noise variance has been estimated, diffusion effects the step-size variation of neighboring nodes and the combination of these two factors results in the steady-state MSD for all nodes being nearly the same for all cases. This is in contrast with other algorithms for which the steady-state MSD is effected by the SNR at each node, even when the SNR is high, as noise variance is unknown for those algorithms.

4.4.2 Effect of noise variance estimate mismatch

Next, the robustness of the proposed NCDLMS algorithm is shown when there is a mismatch in the noise variance estimate. The performance is compared with that of the VSSDLMS algorithm. Figure 4.5 shows the comparison for a SNR of 10 dB. As can be seen from the figure, the performance degrades as the mismatch increases but the performance is still better than the VSSDLMS algorithm. The

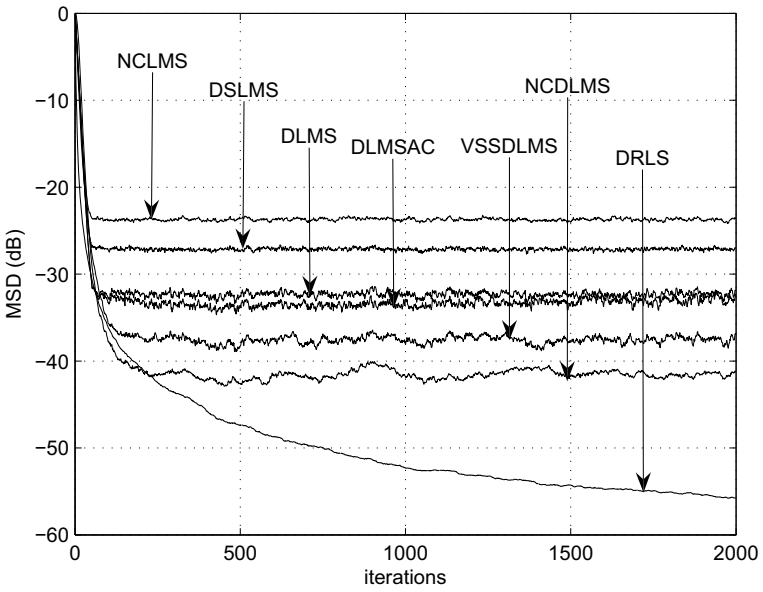

Figure 4.1: MSD for 20 nodes at SNR 10 dB.

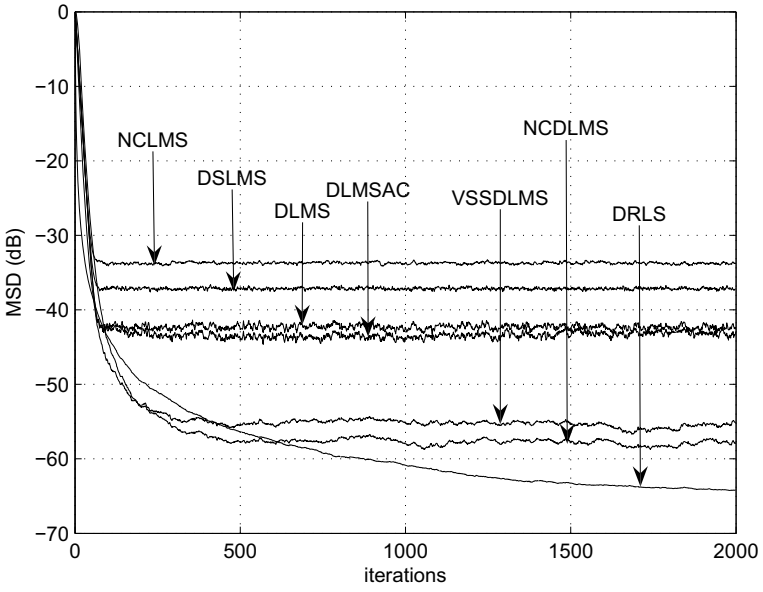

Figure 4.2: MSD for 20 nodes at SNR 20 dB.

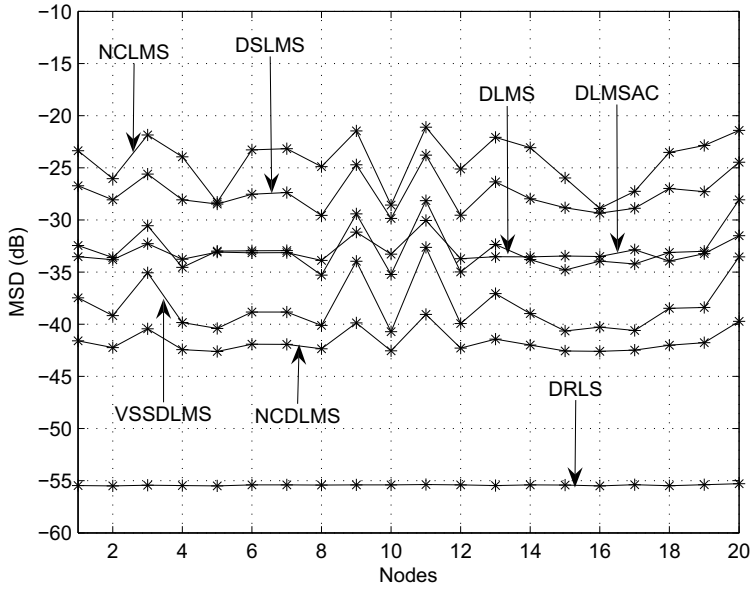

Figure 4.3: MSD at steady-state for 20 nodes at SNR 10 dB.

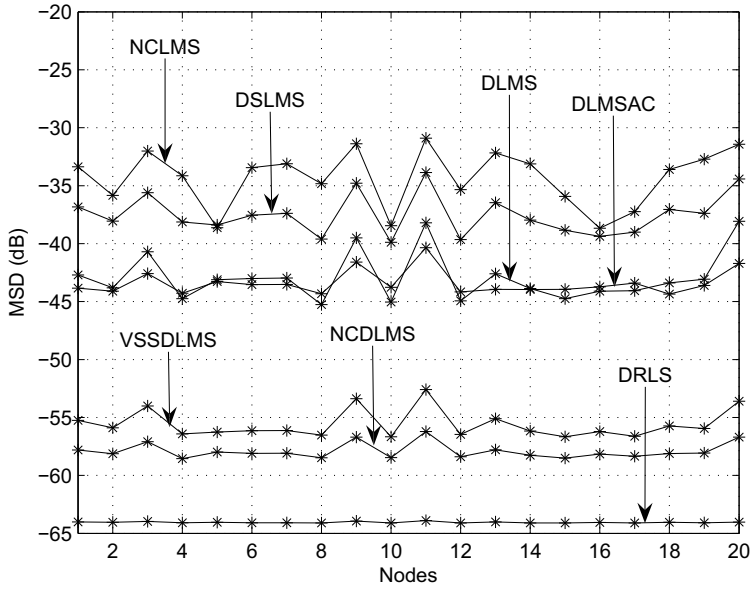

Figure 4.4: MSD at steady-state for 20 nodes at SNR 20 dB.

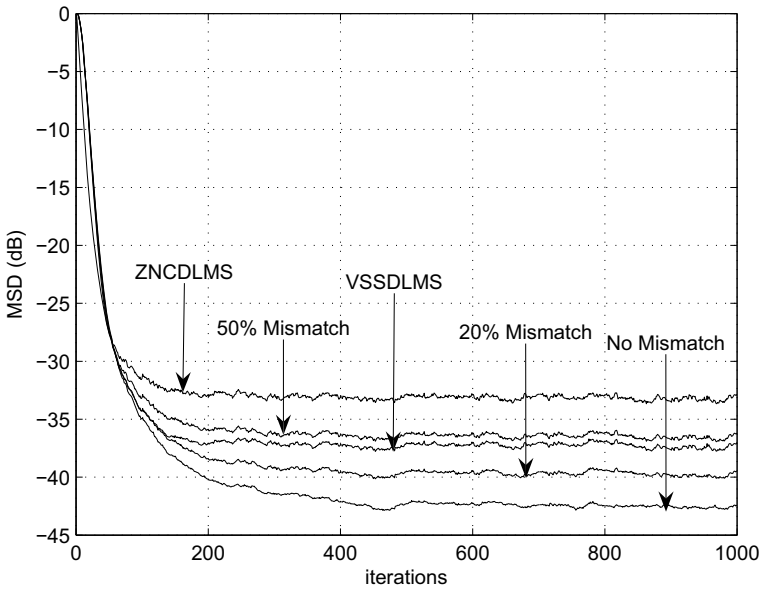

Figure 4.5: Mismatch at SNR 10 dB.

ZNCDLMS algorithm performs slightly worse than the VSSDLMS algorithm but its complexity is comparable to that of the VSSDLMS algorithm and the performance is justified.

4.4.3 Theoretical analysis results

Next, the theoretical analysis of the proposed algorithm as compared to the simulation results is reported in Figs. 4.6 and 4.7 for perfect noise variance estimation and Figs. 4.8 and 4.9 for a 50% mismatch in noise variance estimation. As can be seen from these figures, the simulation analysis are corroborating the theoretical findings very well. The value for $\alpha = 0.01$ whereas $\gamma = 20$ and the results are shown for SNR of 10 dB and 20 dB.

4.4.4 Effect of network size

The effect on the performance of the proposed algorithm when the size of the network varies is reported in Figs. 4.10 and 4.11. An increase in network size improves performance. As can be seen, the improvement is approximately linear. However, the computational complexity greatly increases for large networks as the connectivity also increases. Furthermore, the trends shown in Figs. 4.10 and 4.11 show a vast improvement in performance over the previous algorithms.

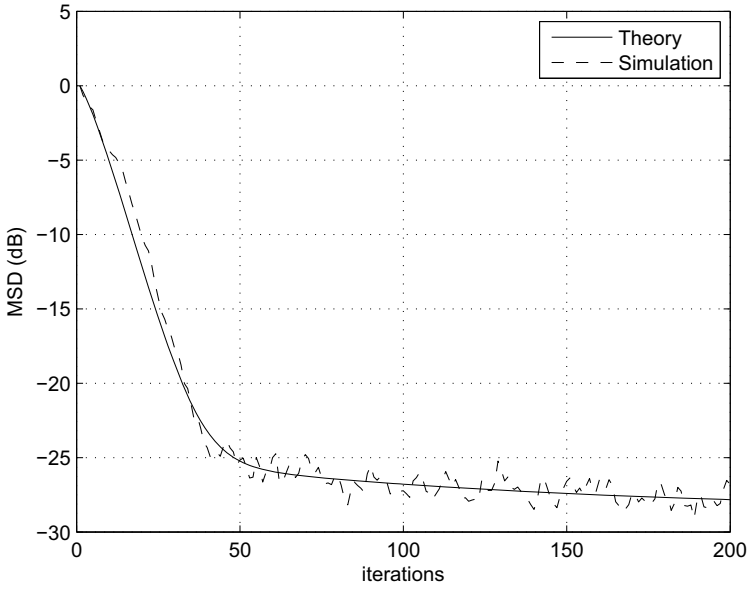

Figure 4.6: MSD for theory and simulation at SNR 10 dB.

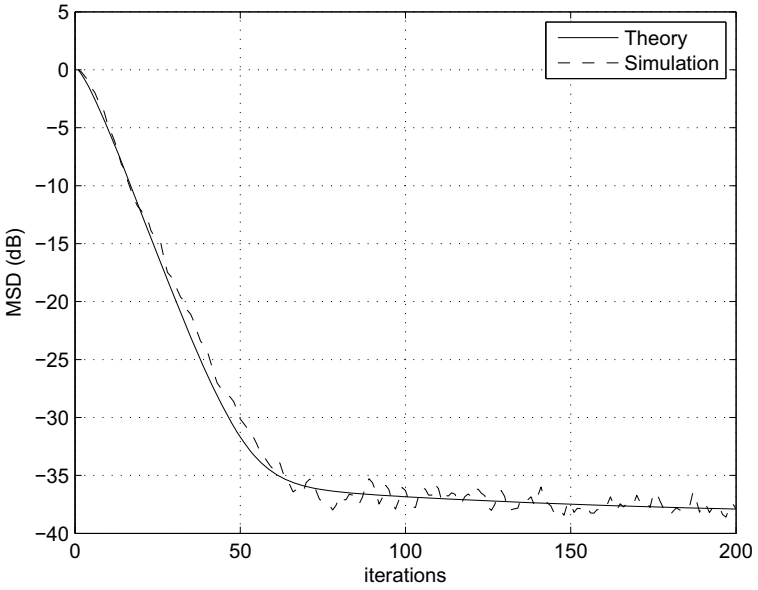

Figure 4.7: MSD for theory and simulation at SNR 20 dB.

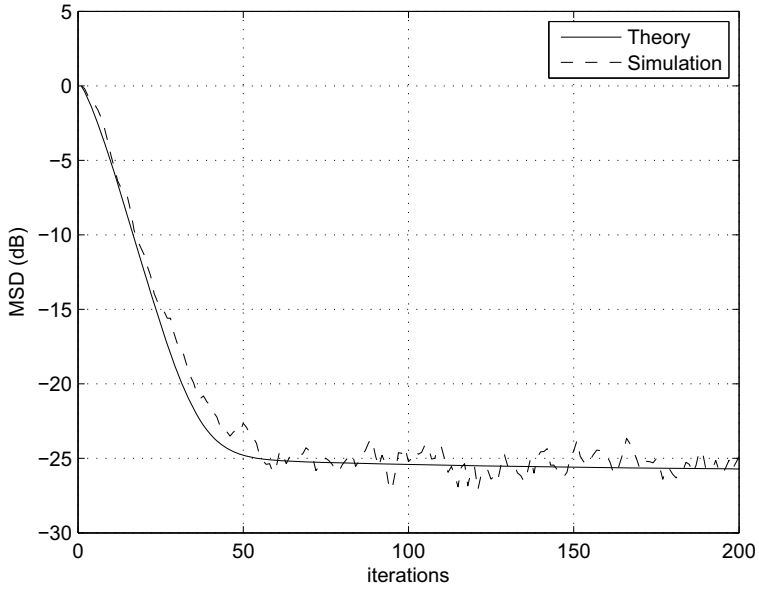

Figure 4.8: MSD for theory and simulation at SNR 10 dB for 50% mismatch.

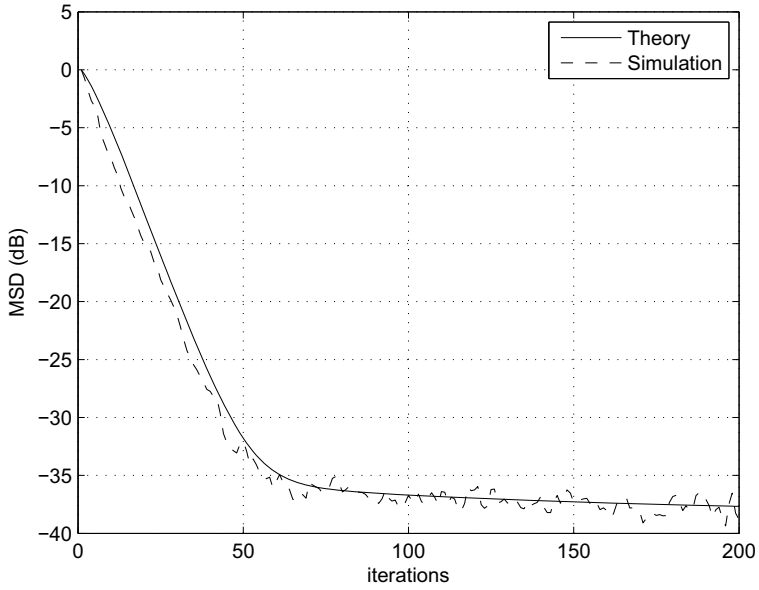

Figure 4.9: MSD for theory and simulation at SNR 20 dB for 50% mismatch.

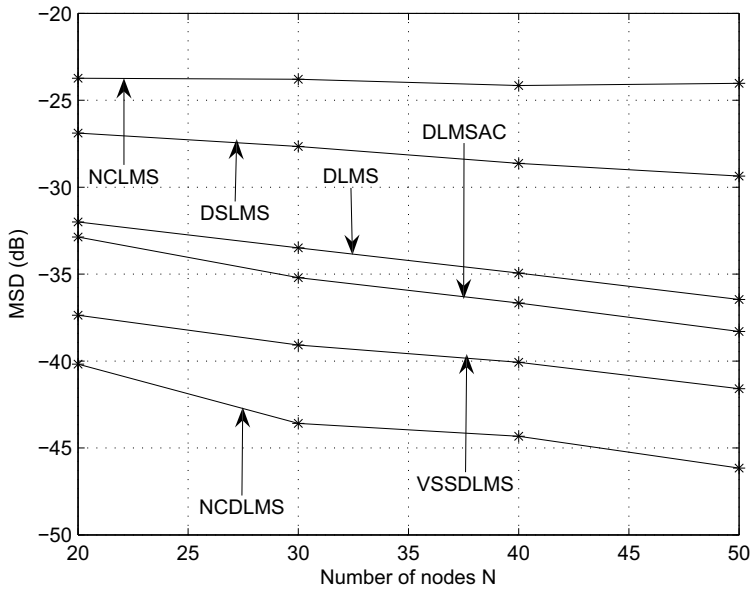

Figure 4.10: Steady-state MSD for varying N at SNR 10 dB.

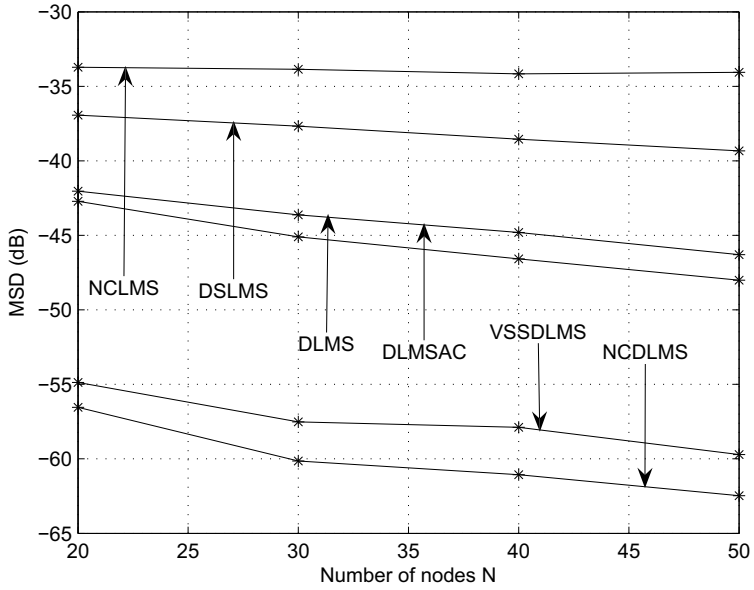

Figure 4.11: Steady-state MSD for varying N at SNR 20 dB.

4.4.5 Effect of node malfunction

As before the robustness of the proposed algorithm is assessed when several nodes switch off. A network of 50 nodes is chosen. Two cases are considered, one where 15 nodes are switched off and another where 30 nodes are switched off. Results are shown in Figs. 4.12 and 4.13 for SNR of 10 dB and 20 dB, respectively. Again, the nodes to be switched off are chosen at random. As seen in the previous chapter, even with more than half the nodes switched off, the network still performs very well as the remaining network remains intact. The degradation would be slightly more severe if the malfunctioning nodes are those with most neighbors as that would reduce cooperation significantly. However, the network is still able to perform by adjusting itself to the change.

4.4.6 Performance of generalized algorithm

A generalized version of the algorithm was derived where sensor nodes can share all available data given enough resources. A comparison between the two versions of the proposed algorithm is shown in Figs. 4.14 and 4.15. As can be seen from the figures, the generalized algorithm performs much better in comparison with the originally proposed algorithm. However, as explained earlier, the generalized algorithm can only be utilized in cost effective applications.

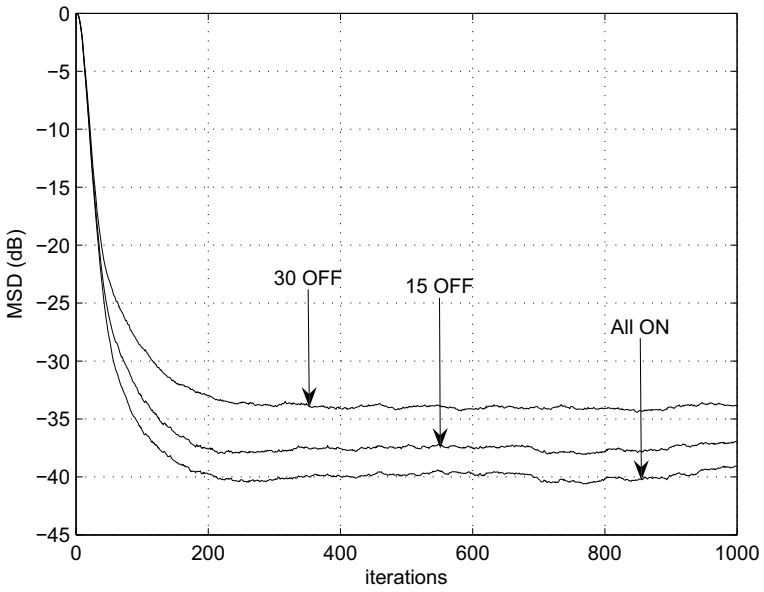

Figure 4.12: Node malfunction performance at SNR 10 dB.

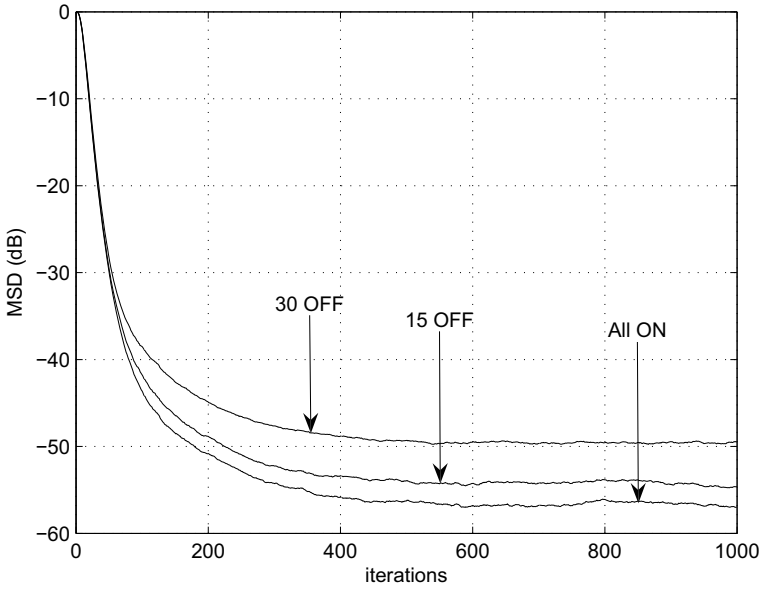

Figure 4.13: Node malfunction performance at SNR 20 dB.

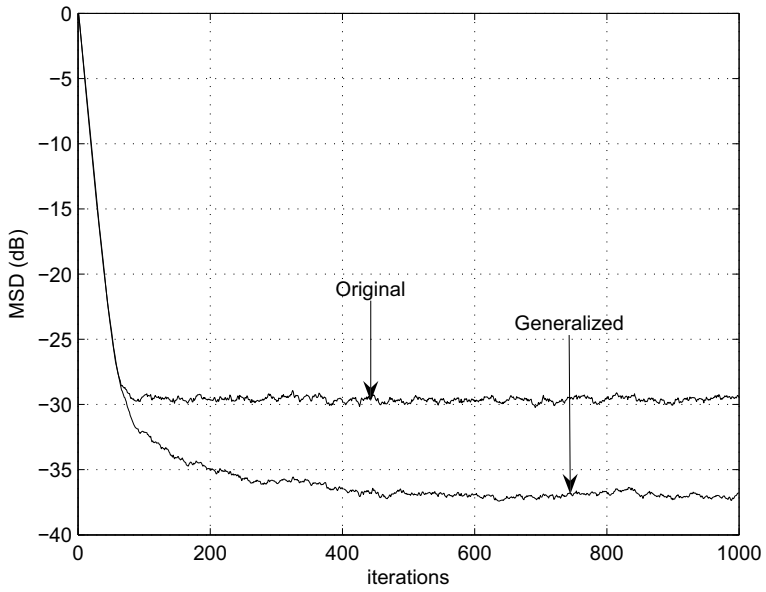

Figure 4.14: Proposed algorithms at SNR 10 dB.

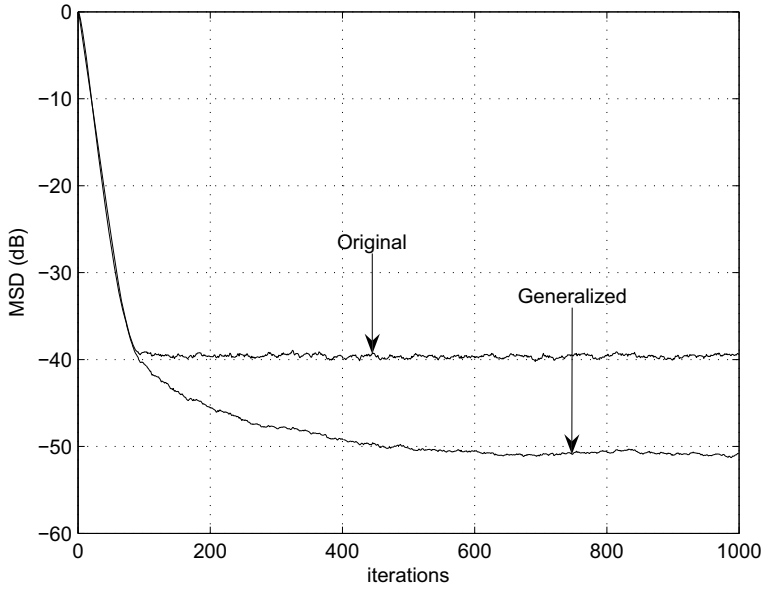

Figure 4.15: Proposed algorithms at SNR 20 dB.

4.4.7 Steady-state performance

Finally, we look at the stability analysis of the algorithm. Here, the auto-correlation matrix, $\mathbf{R}_{\mathbf{u},k}$, is taken to be an identity matrix. Table 4.1 gives results for steady-state MSD for the network when the value of μ_k is varied, for $k = 3$, $\gamma_{\text{NC}} = 0.1$, $\alpha_{\text{NC}} = 0.01$, and SNR = 20 dB. From this table, it can be seen that the simulations corroborate the theoretical finding for the steady-state MSD. Moreover, the bound in (4.28) holds true.

μ_k	γ_{NC}	**SS-MSD simulations**	**SS-MSD equation (4.56)**
1.9	0.1	-15.3	-15.7
1.75	0.1	-18.4	-18.6
1.5	0.1	-21.5	-21.5
1	0.1	-25.8	-26

Table 4.1: Comparison of MSD, from simulations and theory.

4.5 Conclusion

The noise constrained diffusion LMS (NCDLMS) algorithm is proposed in this chapter. Complete derivation of the algorithm using the ATC diffusion technique is given for estimation in a wireless sensor network. A generalized form of the algorithm is suggested as a special case. Complete convergence and steady-state analyses are carried out including the effect of noise variance estimate mismatch. Simulations are carried out to assess the performance of the proposed algorithm under different scenarios. The algorithm is found to outperform all existing algorithms including the VSSDLMS algorithm suggested in the previous chapter. A comparison of the performance of the algorithm is shown for various degrees of noise variance estimate mismatch, including the case of zero NCDLMS (ZNCDLMS) algorithm, which is found to have similar complexity and performance as the VSSDLMS algorithm. Theoretical results are compared with simulation results for exact estimate as well as the mismatch case and the results are found to be corroborating each other. The performance of the proposed algorithm is then studied under different scenarios and the algorithm is found to be robust under all scenarios. Finally, a table lists results for the steady-state analysis and theoretical results corroborate simulation results. Furthermore, the step-size limits defined by (4.28) are also corroborated by simulation results.

CHAPTER 5

BLIND DIFFUSION ALGORITHMS

5.1 Introduction

The previous two chapters introduced algorithms that assume that the input regressor data, $\mathbf{u}_{k,i}$, is available at the sensors. If this information is not available, then the said problem becomes a blind estimation problem. Blind algorithms have been a topic of interest ever since Sato devised a blind equalization approach in [61]. Since then several algorithms have been derived for blind estimation [62], [63], [64]. The work in [64] summarizes the second order statistics based approaches, also known as subspace methods, for blind identification. These include multichannel as well as single channel blind estimation methods. The work in [65] is one of the most popular blind estimation techniques for a single-input-single-output (SISO) model. The work in [66] shows that the technique of [65] can be

improved using only two blocks of data. The authors in [67] use the idea from [66] to devise an algorithm that uses only 2 blocks of data and does indeed show improvement over the algorithm of [65], which uses N blocks. However, the computational complexity of this new algorithm is greater as well. The authors in [68] improve both these algorithms by generalizing them. In [69], a Cholesky factorization based least squares solution is suggested that simplifies the work of [65], [67] and [68]. The performance is shown to be only slightly degraded but taking more number of blocks shows that even though the computational complexity reduces remarkably, the performance is not as good as the previous works. However, in systems where less complexity is required and performance can be compromised to an extent, this algorithm provides a good substitute. In this work, blind block recursive least squares algorithms are derived from the works in [65] and [69] and then implemented in a distributed WSN environment using the diffusion approach suggested by [36].

5.1.1 Singular Value Decomposition Based Blind Algorithm

The work in [65] uses redundant filterbank precoding to construct data blocks that have trailing zeros. These data blocks are then collected at the receiver and used for blind channel identification. In this work, however, there is no precoding required. The trailing zeros will still be used though, for estimation purposes. Let the unknown vector be of size $(L \times 1)$. Suppose the input vector is a $(P \times 1)$

vector with $P - M$ trailing zeros

$$\mathbf{s}_i = \{s_0(i), s_1(i), ..., s_{M-1}(i), 0, ..., 0\}^T, \tag{5.1}$$

where P and M are related through $P = M + L - 1$. The unknown vector can be written in the form of a convolution matrix given by

$$\mathbf{W} = \begin{bmatrix} w(0) & 0 & \cdots & 0 \\ w(1) & w(0) & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ w(L-1) & \cdots & w(0) & 0 \\ 0 & w(L-1) & \ddots & w(0) \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & w(L-1) \end{bmatrix}, \tag{5.2}$$

where $\mathbf{w}^o = [w(0), w(1), ..., w(L-1)]$ is the unknown vector. The output data block can now be written as

$$\mathbf{d}_i = \mathbf{W}\mathbf{s}_i + \mathbf{v}_i, \tag{5.3}$$

where $\mathbf{v}$ is added white Gaussian noise. The output blocks are collected together to form a matrix

$$\mathbf{D}_N = (\mathbf{d}_0, \mathbf{d}_1, ..., \mathbf{d}_{N-1}), \tag{5.4}$$

where N is greater than the minimum number of data blocks required for the input blocks to have a full rank. The singular value decomposition (SVD) of the auto-

correlation of $\mathbf{D}_N$ gives a set of null eigenvectors. Thus the eigendecomposition

$$\mathbf{D}_N\mathbf{D}_N^T = \begin{pmatrix} \bar{\mathbf{U}} & \tilde{\mathbf{U}} \end{pmatrix} \begin{pmatrix} \Sigma_{M\times M} & \mathbf{0}_{M\times(L-1)} \\ \mathbf{0}_{(L-1)\times M} & \mathbf{0}_{(L-1)\times(L-1)} \end{pmatrix} \begin{pmatrix} \bar{\mathbf{U}}^T \\ \tilde{\mathbf{U}}^T \end{pmatrix}, \tag{5.5}$$

gives the $(P \times L-1)$ matrix $\tilde{\mathbf{U}}$ whose columns form the null space for $\mathbf{D}_N$, which implies

$$\tilde{\mathbf{U}}^T\mathbf{W} = \mathbf{0}, \tag{5.6}$$

which can also be written as

$$\tilde{\mathbf{u}}_l^T\mathbf{W} = \mathbf{0}^T, \tag{5.7}$$

where $l = 1, ..., L-1$. This equation denotes convolution since $\mathbf{W}$ is essentially a convolution matrix. Since convolution is commutative, equation (5.7) can also be written as

$$\mathbf{w}^T\mathcal{U} := \mathbf{w}^T\left(\mathcal{U}_1 ... \mathcal{U}_{L-1}\right) = \mathbf{0}^T, \tag{5.8}$$

where $\mathcal{U}_l$ is an $(L \times M)$ Hankel matrix given by

$$\mathcal{U}_l = \begin{bmatrix} \tilde{\mathbf{u}}_l(0) & \tilde{\mathbf{u}}_l(1) & \cdots & \tilde{\mathbf{u}}_l(P-L-1) \\ \tilde{\mathbf{u}}_l(1) & \tilde{\mathbf{u}}_l(2) & \cdots & \tilde{\mathbf{u}}_l(P-L) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{\mathbf{u}}_l(L-1) & \tilde{\mathbf{u}}_l(L) & \cdots & \tilde{\mathbf{u}}_l(P-1) \end{bmatrix}. \tag{5.9}$$

The final estimate is given by the unique solution (up to a constant factor) for equation (5.8).

5.1.2 Cholesky Factorization Based Blind Algorithm

The work in [69] describes a method based on the Cholesky factorization to blindly estimate the channel. Again, the output equation can be written as

$$\mathbf{d}_i = \mathbf{W}\mathbf{s}_i^T + \mathbf{v}_i. \tag{5.10}$$

Taking the auto-correlation of $\mathbf{d_i}$ in equation (5.10) and assuming the input data regressors are white Gaussian with variance $\sigma_{\mathbf{u}}^2$, we get

$$\begin{aligned} \mathbf{R_d} &= \mathrm{E}\left[\mathbf{d}_i\mathbf{d}_i^T\right] \\ &= \sigma_{\mathbf{s}}^2\mathbf{W}\mathbf{W}^T + \sigma_{\mathbf{v}}^2\mathrm{I}. \end{aligned} \tag{5.11}$$

Now if the second order statistics of both the input regressor data as well as the additive noise are known then the correlation matrix for the unknown vector can be written as

$$\begin{aligned} \mathbf{R_w} &= \mathbf{W}\mathbf{W}^T \\ &= \left(\mathbf{R_d} - \sigma_{\mathbf{v}}^2\mathrm{I}\right)/\sigma_{\mathbf{s}}^2. \end{aligned} \tag{5.12}$$

However, this information is not always known and cannot be easily estimated as well, particularly the information about the input regressor data. Therefore, the correlation matrix of the unknown vector has to be approximated by the correlation matrix of the received/sensed data. Now the algorithm in [69] takes

the Cholesky factor of this matrix and then provides a least squares solution for the estimate of the unknown vector.

The method given in [69] is summarized here. Since the correlation matrix is not available at the receiver, an approximate matrix is calculated using K blocks of data. So the correlation matrix is given by

$$\hat{\mathbf{R}}_{\mathbf{d}} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{d}_i \mathbf{d}_i^T. \tag{5.13}$$

As the second order statistics are not known, the noise variance is estimated and then subtracted from the correlation matrix. Thus, we have

$$\begin{aligned} \hat{\mathbf{R}}_{\mathbf{w}} &= \hat{\mathbf{R}}_{\mathbf{d}} - \hat{\sigma}_v^2 \mathrm{I}_K \\ &= \frac{1}{K}\sum_{i=1}^{K} \mathbf{d}_i \mathbf{d}_i^T - \hat{\sigma}_v^2 \mathrm{I}_K, \end{aligned} \tag{5.14}$$

where the noise variance is estimated using the null space of the estimated correlation matrix, $\hat{\mathbf{R}}_{\mathbf{d}}$. If the number of data blocks considered are not enough then the resulting matrix could be singular. Taking the Cholesky factor of this matrix gives us the upper triangular matrix $\hat{\mathbf{G}}$

$$\hat{\mathbf{G}} = chol\left\{\hat{\mathbf{R}}_{\mathbf{w}}\right\}, \tag{5.15}$$

Next we use the *vec* operator to get a $M^2 \times 1$ vector $\hat{\mathbf{g}}$

$$\hat{\mathbf{g}} = vec\left\{\hat{\mathbf{G}}\right\}. \tag{5.16}$$

Its given that the vectors $\mathbf{g}$ and $\mathbf{w}^o$ are related through

$$\mathbf{g} = \mathbf{Q}\mathbf{w}^o, \tag{5.17}$$

where $\mathbf{Q}$ is a $M^2 \times M$ selection matrix given by $\mathbf{Q} = \left[\mathbf{J}_1^T \mathbf{J}_2^T ... \mathbf{J}_M^T\right]^T$, and the $M \times M$ matrices $\mathbf{J}_k$ are defined as

$$\mathbf{J}_k = \begin{cases} 1, & if \;\; s+t = k-1 \\ 0, & otherwise \end{cases} \tag{5.18}$$

where $\{s, t, k\} = 0, ..., M-1$. So the least squares solution is given as

$$\hat{\mathbf{w}} = \left(\mathbf{Q}^T\mathbf{Q}\right)^{-1}\mathbf{Q}^T\hat{\mathbf{g}}. \tag{5.19}$$

The work in [69] also gives a method to estimate the noise variance. Results given in [69] show that this method works just fine. However, it is not so easy to find a correct estimate of noise at low SNR and subtracting it from the auto-correlation matrix may not yield a positive definite matrix, which means that Cholesky factorization may not be possible. Without the use of the noise estimate, however, the estimate is poor. The main advantage of this method is very low computational complexity. Whereas the method of [65] requires singular value decomposition of the auto-correlation matrix followed by formation of Hankel matrices using the null eigenvectors and then finding a unique solution to an overdetermined set of linear equation, this method simply evaluates the Cholesky

factor upper triangular matrix of the auto-correlation matrix and directly finds the estimate from this matrix. Computational complexity is, thus, greatly reduced. However, performance is also degraded.

Both these methods require that several blocks of data be stored before estimation can be performed. Although the least squares approximation gives a good estimate, a sensor network requires a recursive algorithm in order to cooperate and enhance performance. Therefore, the first step would be to make both these algorithms recursive in order to utilize them in a WSN setup.

5.1.3 Blind Block Recursive SVD Algorithm

Here we show how the algorithm from [65] can be made into a blind block recursive algorithm. Since the algorithm requires a complete block of data, we base our iterative process on blocks as well. So instead of the matrix $\mathbf{D}$, we have the block data vector $\mathbf{d}$. The auto-correlation matrix for the first data block is defined in as

$$\hat{\mathbf{R}}_{\mathbf{d}}(1) = \mathbf{d}_1 \mathbf{d}_1^T. \tag{5.20}$$

The matrix is expanded for two blocks in the original algorithm as

$$\begin{aligned} \hat{\mathbf{R}}_{\mathbf{d}}(2) &= \mathbf{D}_2 \mathbf{D}_2^T \\ &= \begin{bmatrix} \mathbf{d}_1 & \mathbf{d}_2 \end{bmatrix} \begin{bmatrix} \mathbf{d}_1^T \\ \\ \mathbf{d}_2^T \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \mathbf{d}_1\mathbf{d}_1^T + \mathbf{d}_2\mathbf{d}_2^T \\
&= \hat{\mathbf{R}}_{\mathbf{d}}(1) + \mathbf{d}_2\mathbf{d}_2^T. \qquad (5.21)
\end{aligned}
$$

From this a generalization can be written as

$$\hat{\mathbf{R}}_{\mathbf{d}}(i) = \hat{\mathbf{R}}_{\mathbf{d}}(i-1) + \mathbf{d}_i\mathbf{d}_i^T. \qquad (5.22)$$

The first few iterations may not give a good estimate and the error may even seem to be increasing as the matrix will be rank deficient. However, with sufficient data blocks, the rank becomes full and the estimate then begins to improve. The next step is to get the eigendecomposition for this matrix. Applying the SVD on $\mathbf{R_d}$ we get the eigenvector matrix $\mathbf{U}$, from which we get the $(L-1 \times M)$ matrix $\tilde{\mathbf{U}}$ forming the null space of the autocorrelation matrix, from which we can form the L Hankel matrices of size $(L \times M+1)$ that are concatenated to give the matrix $\mathcal{U}(\rangle)$ from which we can finally get the estimate $\tilde{\mathbf{w}}(i)$

$$SVD\{\mathbf{R}_{\mathbf{d}}(i)\} \Rightarrow \mathbf{U}(i) \Rightarrow \tilde{\mathbf{U}}(i) \Rightarrow \mathcal{U}(i) \Rightarrow \tilde{\mathbf{w}}_i. \qquad (5.23)$$

The update for the estimate of the unknown vector is then given by

$$\hat{\mathbf{w}}_i = \lambda\hat{\mathbf{w}}_{i-1} + (1-\lambda)\tilde{\mathbf{w}}_i. \qquad (5.24)$$

It can be seen that the recursive algorithm does not become computationally less complex. However, it does require lesser memory and the result improves with

increase in the number of data blocks. The performance almost matches that of the batch processing least squares algorithm of [65].

Table 5.1 gives the Blind Block Recursive SVD (BBRS) algorithm. The forgetting factor is fixed in this case. If the forgetting factor value is changed to $\lambda_i = 1 - \frac{1}{i}$, the algorithm becomes Variable Forgetting Factor BBRS (VFFBBRS) algorithm. However, simulations show that the VFFBBRS algorithm converges slower. Results show that if the forgetting factor is small, the algorithm converges faster even though it gives a higher error floor at steady-state. When varied, the forgetting factor increases with time so the convergence slows down even though it eventually reaches a very low steady-state error floor.

5.1.4 Blind Block Recursive Cholesky Algorithm

In this section, we show how the algorithm of [69] can be converted into a blind block recursive solution. Equation (5.14) can be rewritten as

$$\begin{aligned}
\hat{\mathbf{R}}_{\mathbf{w}}(i) &= \frac{1}{i}\sum_{n=1}^{i}\mathbf{d}_n\mathbf{d}_n^T - \hat{\sigma}_v^2\mathrm{I}_K \\
&= \frac{1}{i}\mathbf{d}_i\mathbf{d}_i^T + \frac{1}{i}\sum_{n=1}^{i-1}\mathbf{d}_n\mathbf{d}_n^T - \hat{\sigma}_v^2\mathrm{I}_K \\
&= \frac{1}{i}\left(\mathbf{d}_i\mathbf{d}_i^T - \hat{\sigma}^2\mathrm{I}_K\right) + \frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1) \qquad (5.25)
\end{aligned}$$

Blind Block Recursive SVD Algorithm

Step 1. Form auto-correlation matrix for iteration i from equation (5.22).

Step 2. Get $\mathbf{U}(i)$ from SVD of $\hat{\mathbf{R}}_{\mathbf{d}}(i)$.

Step 3. Form $\tilde{\mathbf{U}}(i)$ from the null eigenvectors of $\mathbf{U}(i)$.

Step 4. Form Hankel matrices of size $(L \times M-1)$ from individual vectors of $\tilde{\mathbf{U}}(i)$.

Step 5. Form $\mathcal{U}(i)$ by concatenating the Hankel matrices.

Step 6. The null eigenvector from the SVD of $\mathcal{U}(i)$ is the estimate $\tilde{\mathbf{w}}_i$.

Step 7. Use $\tilde{\mathbf{w}}_i$ in equation (5.24) to get the update $\hat{\mathbf{w}}_i$.

Table 5.1: Summary of the Blind Block Recursive SVD (BBRS) algorithm.

Similarly, we have

$$\hat{\mathbf{G}}(i) = chol\left\{\hat{\mathbf{R}}_{\mathbf{w}}(i)\right\} \tag{5.26}$$

$$\hat{\mathbf{g}}_i = vec\left\{\hat{\mathbf{G}}(i)\right\}. \tag{5.27}$$

Letting $\mathbf{Q}_A = \left(\mathbf{Q}^T\mathbf{Q}\right)^{-1}\mathbf{Q}^T$, we have

$$\hat{\mathbf{w}}_i = \mathbf{Q}_A\hat{\mathbf{g}}_i. \tag{5.28}$$

This equation is still not recursive. So we expand $\hat{\mathbf{g}}_i$ in terms of equation (5.25) and use the fact that the *vec* operator is linear to get

$$\begin{aligned}
\hat{\mathbf{w}}_i &= \mathbf{Q}_A\hat{\mathbf{g}}_i \\
&= \mathbf{Q}_A vec\left\{\hat{\mathbf{G}}(i)\right\} \\
&= \mathbf{Q}_A vec\left\{chol\left\{\hat{\mathbf{R}}_{\mathbf{w}}(i)\right\}\right\} \\
&= \mathbf{Q}_A vec\left\{chol\left\{\frac{1}{i}\mathbf{d}_i\mathbf{d}_i^T + \frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\}\right\},
\end{aligned} \tag{5.29}$$

where we now add and subtract $chol\left\{\frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\}$ from the right hand side of the equation to get

$$\begin{aligned}
\hat{\mathbf{w}}_i = \mathbf{Q}_A vec\Bigg\{&chol\left\{\frac{1}{i}\mathbf{d}_i\mathbf{d}_i^T + \frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\} \\
&-chol\left\{\frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\} + chol\left\{\frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\}\Bigg\}
\end{aligned} \tag{5.30}$$

$$= \mathbf{Q}_A vec\left\{chol\left\{\frac{1}{i}\mathbf{d}_i\mathbf{d}_i^T + \frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\} - chol\left\{\frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\}\right\}$$
$$+\mathbf{Q}_A vec\left\{chol\left\{\frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\}\right\}. \quad (5.31)$$

Recognizing that $chol\left\{\frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\} = \frac{i-1}{i}\hat{\mathbf{g}}(i-1)$ and $\mathbf{Q}_A vec\left\{chol\left\{\frac{i-1}{i}\hat{\mathbf{R}}_{\mathbf{w}}(i-1)\right\}\right\} = \frac{i-1}{i}\mathbf{Q}_A\hat{\mathbf{g}}_{i-1} = \frac{i-1}{i}\hat{\mathbf{w}}_{i-1}$, we get

$$\begin{aligned}\hat{\mathbf{w}}_i &= \mathbf{Q}_A\left(\hat{\mathbf{g}}_i - \frac{i-1}{i}\hat{\mathbf{g}}_{i-1}\right) + \frac{i-1}{i}\mathbf{Q}_A\hat{\mathbf{g}}_{i-1} \\ &= \mathbf{Q}_A\left(\hat{\mathbf{g}}_i - \frac{i-1}{i}\hat{\mathbf{g}}_{i-1}\right) + \frac{i-1}{i}\hat{\mathbf{w}}_{i-1}. \quad (5.32)\end{aligned}$$

Letting $\lambda_i = 1 - \frac{1}{i}$, the blind block recursive Cholesky algorithm is summarised in Table 5.2. The table defines the Blind Block Recursive Cholesky algorithm with variable forgetting factor (VFFBBRC). If the forgetting factor is fixed then the algorithm can simply be called Blind Block Recursive Cholesky (BBRC) algorithm. Simulation results show that the VFFBBRC algorithm converges to the least squares solution obtained through the algorithm given in [69]. The BBRC algorithm can also achieve the same result if the value of the forgetting factor is extremely close to 1. However, the convergence speed of the BBRC algorithm is slow compared to that of the VFFBBRC algorithm even though it requires lesser memory and is therefore computationally less complex. There are two issues with the recursive algorithm. Firstly, the Cholesky factorization cannot be applied until at least M blocks of data have been received as the correlation matrix needs to be positive definite in order for the factorization to be possible. The second issue involves the noise variance of the additive noise. In [69] it is shown that if

the noise variance can be estimated, the estimate of the unknown vector will improve. However, using the noise variance in the recursive algorithm can make the resulting matrix to have zero or negative eigenvalues and so the Cholesky factorization may not be possible until a considerable number of data blocks have been received. The performance does degrade if noise variance cannot be subtracted from the auto-correlation matrix but this algorithm is still computationally less complex compared to the SVD approach. One approach is to estimate the noise variance after a certain number of blocks have been received and then use that value for the remainder of the iterations. The table below summarises the algorithm.

5.1.5 Diffusion Blind Block Recursive Algorithms

In order to incorporate the above defined algorithms in a WSN setup, we simply use the ATC scheme for diffusion and incorporating the algorithms directly, we come up with the Diffusion BBRS (DBBRS) and Diffusion BBRC (DBBRC) algorithms. Reforming the algorithms from tables 5.1 and 5.2, the new algorithms can be summarised as in tables 5.3 and 5.4. The subscript k denotes the node number, N_k is the set of neighbors of node k, $\hat{\mathbf{h}}_k$ is the intermediate estimate for node k, c_{lk} is the combination weight for the estimate coming from node l to node k.

Blind Block Recursive Cholesky Algorithm

Step 1. Let forgetting factor be defined as $\lambda_i = 1 - \frac{1}{i}$.

Step 2. Form auto-correlation matrix for iteration i using λ_i in equation (5.25) to get

$$\hat{\mathbf{R}}_{\mathbf{w}}(i) = (1 - \lambda(i)) \left(\mathbf{d}_i \mathbf{d}_i^T - \hat{\sigma}_v^2 \mathrm{I}_K\right) + \lambda(i) \hat{\mathbf{R}}_{\mathbf{w}}(i-1)$$

Step 3. Get $\hat{\mathbf{G}}(i)$ as the Cholesky factor of $\hat{\mathbf{R}}_{\mathbf{w}}(i)$.

Step 4. Apply the *vec* operator to get $\hat{\mathbf{g}}_i$.

Step 5. Use λ_i in equation (5.32) to get the final update

$$\hat{\mathbf{w}}_i = \mathbf{Q}_A (\hat{\mathbf{g}}_i - \lambda_i \hat{\mathbf{g}}_{i-1}) + \lambda_i \hat{\mathbf{w}}_{i-1}.$$

Table 5.2: Summary of Blind Block Recursive Cholesky (BBRC) algorithm

Diffusion Blind Block Recursive SVD Algorithm

Step 1. Form auto-correlation matrix for iteration i from equation (5.22) for each node k.

$$\hat{\mathbf{R}}_{\mathbf{d},k}(i) = \mathbf{d}_{k,i}\mathbf{d}_{k,i}^T + \hat{\mathbf{R}}_{\mathbf{d},k}(i-1)$$

Step 2. Get $\mathbf{U}_k(i)$ from SVD of $\hat{\mathbf{R}}_{\mathbf{d},k}(i)$.

Step 3. Form $\tilde{\mathbf{U}}_k(i)$ from the null eigenvectors of $\mathbf{U}_k(i)$.

Step 4. Form Hankel matrices of size $(L \times M-1)$ from individual vectors of $\tilde{\mathbf{U}}_k(i)$.

Step 5. Form $\mathcal{U}_k(i)$ by concatenating the Hankel matrices.

Step 6. The null eigenvector from the SVD of $\mathcal{U}_k(i)$ is the estimate $\tilde{\mathbf{w}}_{k,i}$.

Step 7. Use $\tilde{\mathbf{w}}_{k,i}$ in equation (5.24) to get the intermediate update $\hat{\mathbf{h}}_{k,i}$.

$$\hat{\mathbf{h}}_{k,i} = \lambda\hat{\mathbf{w}}_{k,i-1} + (1-\lambda)\tilde{\mathbf{w}}_{k,i}$$

Step 8. Combine estimates from neighbors of node k to get $\hat{\mathbf{w}}_{k,i}$.

$$\hat{\mathbf{w}}_{k,i} = \sum_{l\varepsilon N_k} c_{lk}\hat{\mathbf{h}}_{l,i}$$

Table 5.3: Summary of DBBRS algorithm.

Diffusion Blind Block Recursive Cholesky Algorithm

Step 1. Let forgetting factor be defined as $\lambda_{k,i} = 1 - \frac{1}{i}$.

Step 2. Form auto-correlation matrix for iteration k from

$$\hat{\mathbf{R}}_{\mathbf{w},k}(i) = (1 - \lambda_{k,i}) \left(\mathbf{d}_{k,i} \mathbf{d}_{k,i}^T - \hat{\sigma}_{v,k}^2 \mathrm{I}_K \right) + \lambda_{k,i} \hat{\mathbf{R}}_{\mathbf{w},k}(i-1)$$

Step 3. Get $\hat{\mathbf{G}}_k(i)$ as the Cholesky factor of $\hat{\mathbf{R}}_{\mathbf{w},k}(i)$.

Step 4. Apply the *vec* operator to get $\hat{\mathbf{g}}_{k,i}$.

Step 5. The intermediate update is then given as

$$\hat{\mathbf{h}}_{k,i} = \mathbf{Q}_A (\hat{\mathbf{g}}_{k,i} - \lambda_{k,i} \hat{\mathbf{g}}_{k,i-1}) + \lambda_{k,i} \hat{\mathbf{w}}_{k,i-1}.$$

Step 6. The final update is the weighted sum of the estimates of all neighbors of node k

$$\hat{\mathbf{w}}_{k,i} = \sum_{l \varepsilon N_k} c_{lk} \hat{\mathbf{h}}_{l,i}$$

Table 5.4: Summary of DBBRC algorithm

5.2 Computational complexity of the proposed algorithms

In order to fully understand the variation in performance of the two algorithms it is necessary to look at the computational complexity as it tells us how much an algorithm gains in terms of computations as it loses in terms of performance. We first look at the complexity of the original algorithms and then move on to the recursive versions.

5.2.1 Blind SVD Algorithm

The length of the unknown vector is M and the data block size is K. A total number of N data blocks are required for estimation where $N \geq K$. This means that we have a data block matrix of size $K \times N$. The correlation matrix formed using this matrix will thus have the size $K \times K$ and this function requires $K^2\,(2N-1)$ calculations (including both multiplications and additions). The next step is Singular Value Decomposition (SVD), which is done using the QR decomposition algorithm. This algorithm requires a total of $[\frac{4}{3}K^3 + \frac{3}{2}K^2 + \frac{19}{6}K - 6]$ calculations. Then the null eigenvectors are separated and each eigenvector is used to form a Hankel matrix and all the Hankel matrices are stacked together to form a matrix of size $M \times (K-M)(M-1)$. The unique null eigenvector of this new matrix gives the estimate of the unknown vector. To find the eigenvector requires another $[\left(2K + \frac{7}{3}\right) M^3 - 2M^4 + (1-4K)\,\frac{M^2}{2} + \frac{19}{6}M - 6]$ calculations. So the overall

calculations required for the algorithm can be given as

$$\begin{aligned} T_{C,SVD} &= \frac{4}{3}K^3 + \left(2N + \frac{1}{2}\right)K^2 + \frac{19}{6}K + \left(2K + \frac{7}{3}\right)M^3 \\ &\quad -2M^4 + (1 - 4K)\frac{M^2}{2} + \frac{19}{6}M - 12. \end{aligned} \tag{5.33}$$

5.2.2 Blind Cholesky Algorithm

Like the SVD algorithm, here also the unknown vector length is M and the data block size is K. The total number of data blocks are taken as N in order to have uniformity in comparison. The correlation process is the same except for the final averaging step which results in an extra division so the total calculations become $K^2(2N-1)+1$. The next step is to estimate the noise variance, which requires the SVD decomposition and therefore another $[\frac{4}{3}K^3+\frac{3}{2}K^2+\frac{19}{6}K-6]$ calculations. The minimum number of calculations required to estimate the noise variance is 1 division. The noise variance is then subtracted from the diagonal of the correlation matrix, resulting in another K calculations. After that the Cholesky factorization is performed, which requires $[\frac{1}{3}(M^3+3M^2+M)]$ calculations. Finally the last step is to get the estimate of the unknown vector through the pseudo-inverse and this step requires another $[M(2M^2-1)]$ calculations. Thus the total number of calculations required are given as

$$T_{C,Chol} = \frac{4}{3}K^3 + \left(2N + \frac{1}{2}\right)K^2 + \frac{19}{6}K - 4 + \frac{1}{3}\left(7M^3 + 3M^2 - M\right). \tag{5.34}$$

5.2.3 Blind Block Recursive SVD Algorithm

Moving on to the recursive algorithms, we can notice that there is only a slight change in the overall algorithm but it reduces the calculations by nearly half. Since the correlation matrix is only being updated at each iteration, the calculations required for the first step are now only $2K^2$ instead of $K^2(2N-1)$. However, an extra $M+2$ calculations are required for the final step. The overall calculations are thus given as

$$\begin{aligned} T_{C,RS} &= \frac{4}{3}K^3 + \frac{7}{2}K^2 + \frac{19}{6}K + \left(2K + \frac{7}{3}\right)M^3 - 2M^4 \\ &\quad + (1-4K)\frac{M^2}{2} + \frac{25}{6}M - 10. \end{aligned} \tag{5.35}$$

5.2.4 Blind Block Recursive Cholesky Algorithm

Similarly, the number of calculations for the first step for this algorithm is reduced to $2K^2+2$ instead of $K^2(2N-1)+1$. The final step includes an extra K^2+M+2 calculations. Thus the total number of calculations is now given as

$$T_{C,RC} = \frac{4}{3}K^3 + \frac{7}{2}K^2 + \frac{19}{6}K + \frac{1}{3}\left(7M^3 + 3M^2 + 2M\right). \tag{5.36}$$

However, it should be noted that the estimation of the noise variance need not be repeated at each iteration. After a few iterations, the number of which can be fixed *apriori*, the noise variance can be estimated and then this same value can be used in the remaining iterations instead of estimating it repeatedly. The number

of calculations, thus, reduces to

$$T_{C,RC} = 2K^2 + \frac{1}{3}\left(7M^3 + 3M^2 + 2M\right) + 4. \tag{5.37}$$

5.2.5 Complexity Comparison

Here we compare all the algorithms for specific scenarios. The value for M is fixed to 4. The value for K is varied whereas the value for N is varied between 10 and 20 for the least squares algorithms. The number of calculations for the recursive algorithms are shown for one iteration only. The last algorithm is the recursive Cholesky algorithm where the noise variance is calculated only once, after a select number of iterations have occurred, and then it is kept constant. Tables 5.5 and 5.6 summarize the results.

Table 5.5 lists the number of computations for the original algorithms, showing that the Cholesky based method requires lesser computations and so the trade off between performance and complexity is justified. If the number of blocks is small then the Cholesky based method may even perform better than the SVD based method as shown in [69]. Here it is assumed that the exact length of the unknown vector is known. Generally, an upper bound of this value is known and that value is used, resulting in an increase in calculations. Since the assumption stands true for both algorithms here, the comparison is fair.

M = 4		**N = 10**		**N = 20**	
	K = 8	**K = 10**	**K = 8**	**K = 10**	**K = 20**
SVD	2434	4021	3714	6021	28496
Chol	2180	3575	3460	5575	27090

Table 5.5: Computations for original least squares algorithms.

M = 4	**K = 8**	**K = 10**	**K = 20**
RS	1352	2327	13702
RC	1100	1883	12298
RCNV	300	372	972

Table 5.6: Computations for recursive algorithms.

Table 5.6 lists the computations per iteration for the recursive algorithms. RS gives the number of computations for the recursive SVD algorithm and RC is for the recursive Cholesky algorithm. RCNV lists the number of computations when the noise variance is estimated only once. This shows how the complexity of the algorithm can be reduced greatly by a small improvisation. Although the performance does suffer slightly, the gain in complexity more than compensates for this loss.

5.3 Simulations and Results

Here we compare results for the newly developed algorithms. Results are shown for a network of 20 nodes, show in fig. 5.1. The forgetting factor is both varied as well as kept fixed in order to study how the performance varies. The two algorithms are compared with each other and then compared under different scenarios to see how each performs. The forgetting factor, data block size and network size are changed one by one while all other variables are kept constant to check how the performance varies for each algorithm.

5.3.1 Performance of the proposed algorithms

Initially, the two algorithms are used to identify an unknown vector of length $M = 4$ in an environment with signal-to-noise ratio (SNR) of 10 and 20 dB. The

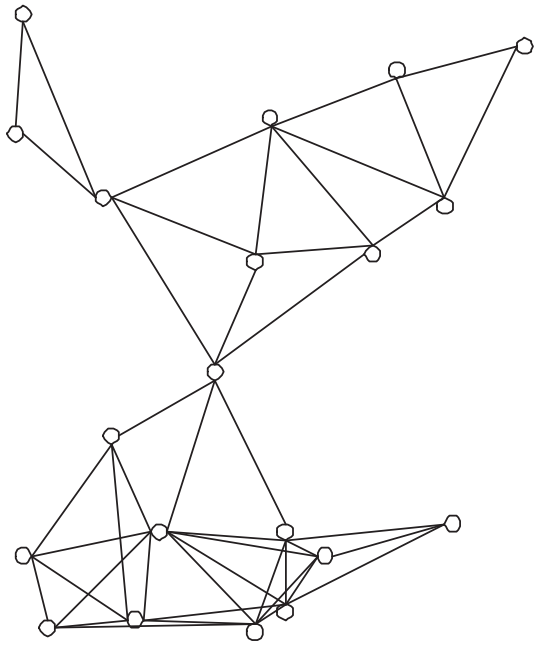

Figure 5.1: Network of 20 nodes.

forgetting factor is fixed at $\lambda = \{0.9, 0.99\}$. The block size is taken as $K = 8$. Results are shown for the two algorithms for both diffusion and no cooperation cases. Figures 5.2-5.5 show these results. As can be seen, the Cholesky algorithm does not perform well when the forgetting factor is small. The performance improves with increase in forgetting factor but the speed of convergence reduces. However, its one main positive attribute remains its low computational complexity. For the SVD algorithm, the performance improves slightly with increase in forgetting factor but at a loss of convergence speed.

5.3.2 Effect of forgetting factor

Next, the performance of each algorithm is separately studied for different values of the forgetting factor. For the fixed case the values taken are $\lambda = \{0.9, 0.95, 0.99\}$ and the results are compared with the variable forgetting factor case. The SNR is chosen as 20 dB and the network size is taken to be 20 nodes. Fig. 5.6 shows the results for the Cholesky factorization based BBRC algorithm. It is seen that the performance improves as the forgetting factor is increased but the speed of convergence slows down. The algorithm performs best when the forgetting factor is variable. The results for the SVD based BBRS algorithm are shown in figs. 5.7-5.9. Fig. 5.7 shows the complete curves. However, there is not much difference in the performance so the figure is zoomed in to see how the algorithm is behaving at the beginning and the end of the simulation. Fig. 5.8 shows the result that is expected. The speed of convergence is fastest for $\lambda = 0.9$ and slowest for $\lambda = 0.99$.

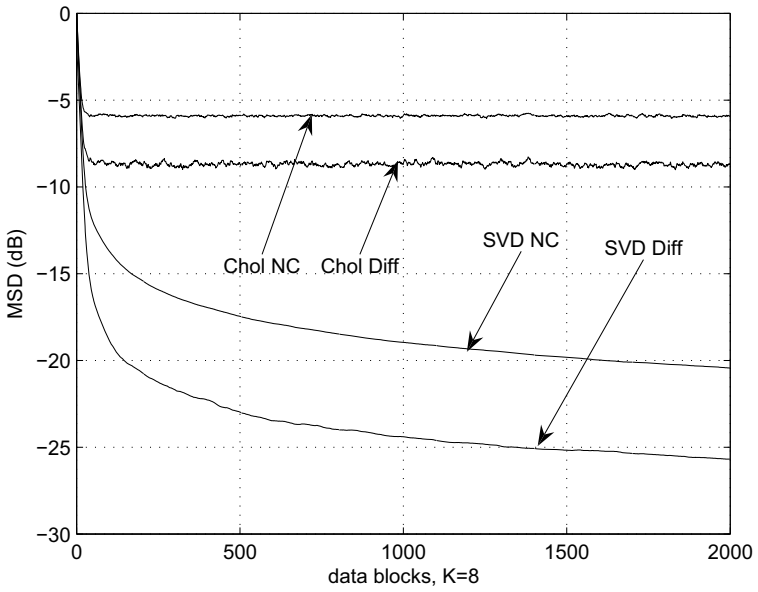

Figure 5.2: MSD at SNR 10 dB and FF 0.9.

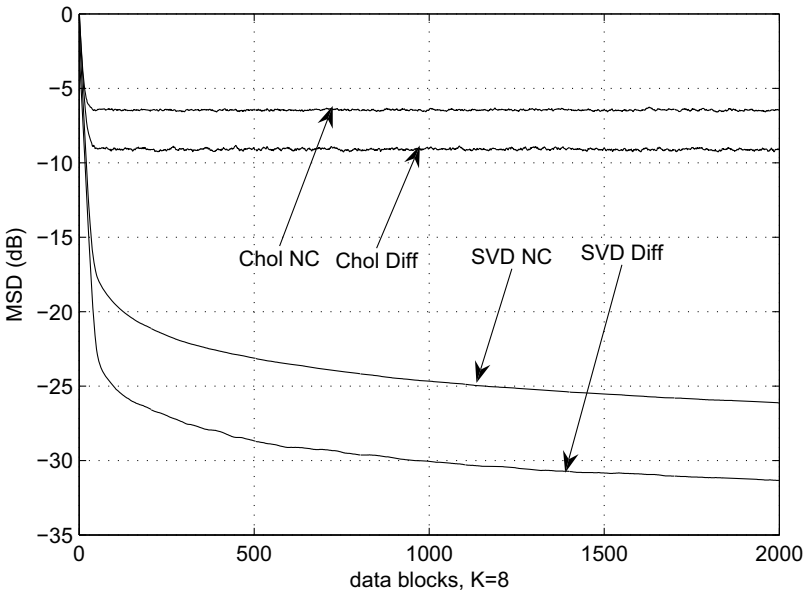

Figure 5.3: MSD at SNR 20 dB and FF 0.9.

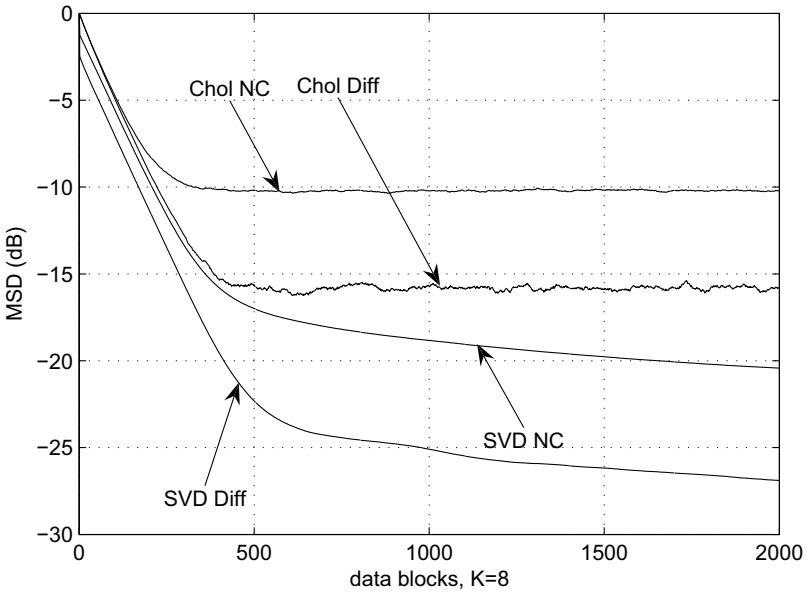

Figure 5.4: MSD at SNR 10 dB and FF 0.99.

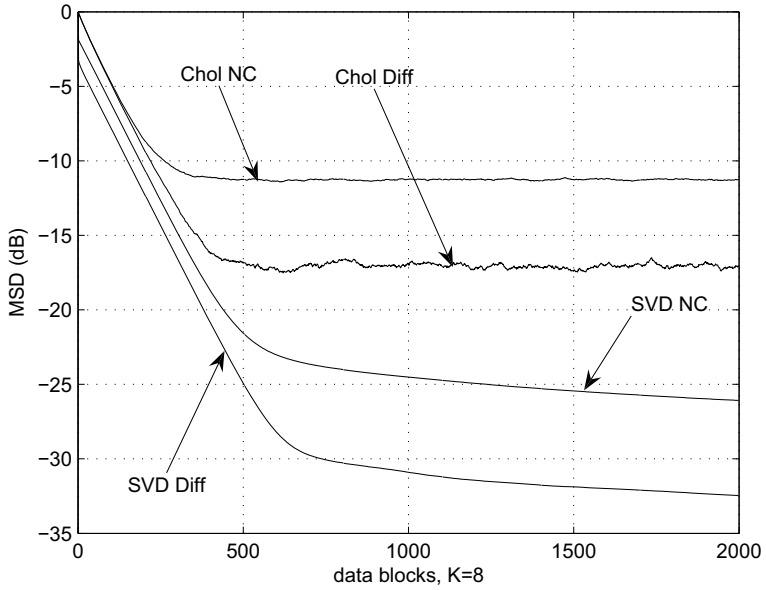

Figure 5.5: MSD at SNR 20 dB and FF 0.99.

For the variable forgetting factor (VFF) case, the speed is fast initially but slows down with time. Fig. 5.9 shows how the curves approach steady-state. It is evident that $\lambda = 0.99$ would give the lowest steady-state error whereas the VFF case would take the longest to reach the steady-state and although the results may be as good as for the case of $\lambda = 0.99$ or even better, the speed of convergence is too slow.

5.3.3 Performance of the proposed algorithms using optimal forgetting factor

From these results it can easily be inferred that the Cholesky factorization based approach yields the best results when the forgetting factor is varied whereas the SVD based algorithm performs best if the forgetting factor is fixed. In order to have a fair performance comparison, the two algorithms need to be compared under conditions in which they both perform the best. Figs. 5.10 and 5.11 give the respective results. As can be seen, at SNR 10 dB, the Cholesky based DBBRC algorithm performs slightly better than the SVD based BBRS algorithm without diffusion whereas both SVD based algorithms outperform the Cholesky based algorithms at SNR 20 dB. However, the BBRC algorithm remains computationally less complex than BBRS algorithm. In the end it is a trade off between complexity and performance while choosing either of the algorithms.

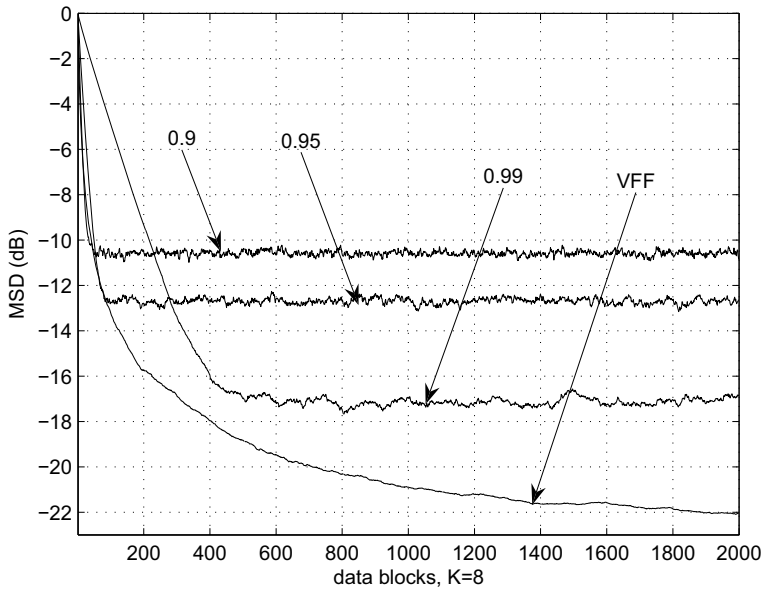

Figure 5.6: MSD at SNR 20 dB for BBRC with different FFs.

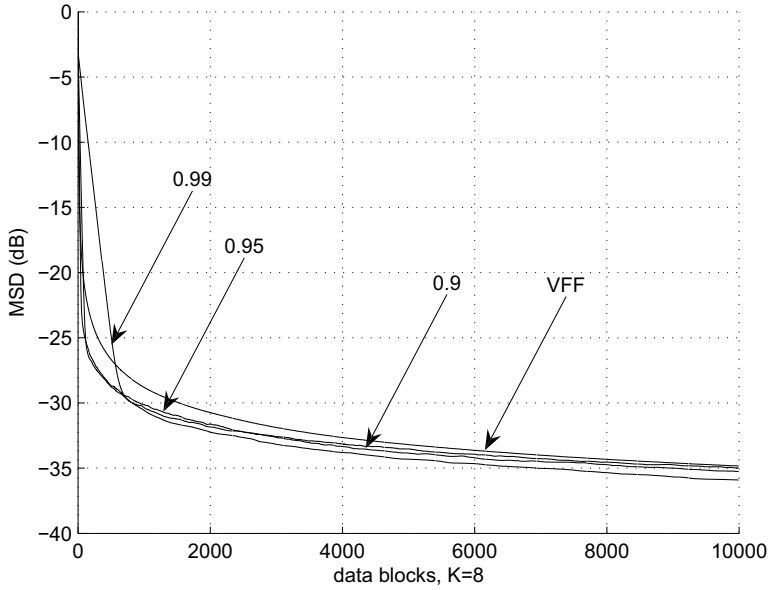

Figure 5.7: MSD at SNR 20 dB for BBRS with different FFs.

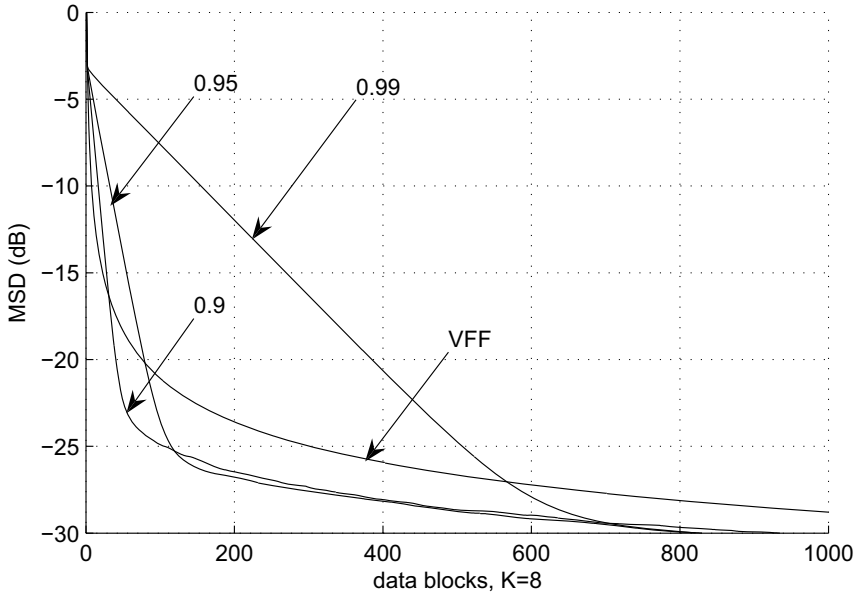

Figure 5.8: MSD at SNR 20 dB for BBRS with different FFs (Transient).

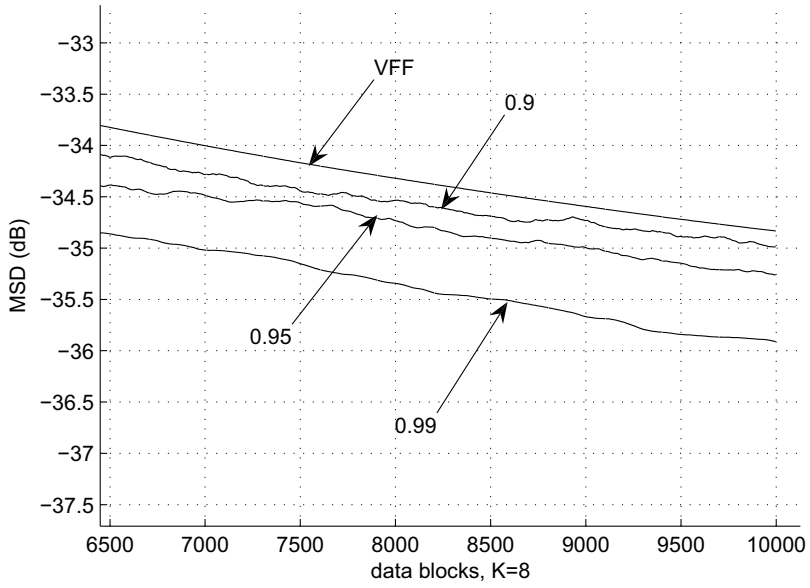

Figure 5.9: MSD at SNR 20 dB for BBRS with different FFs (Near Steady-State).

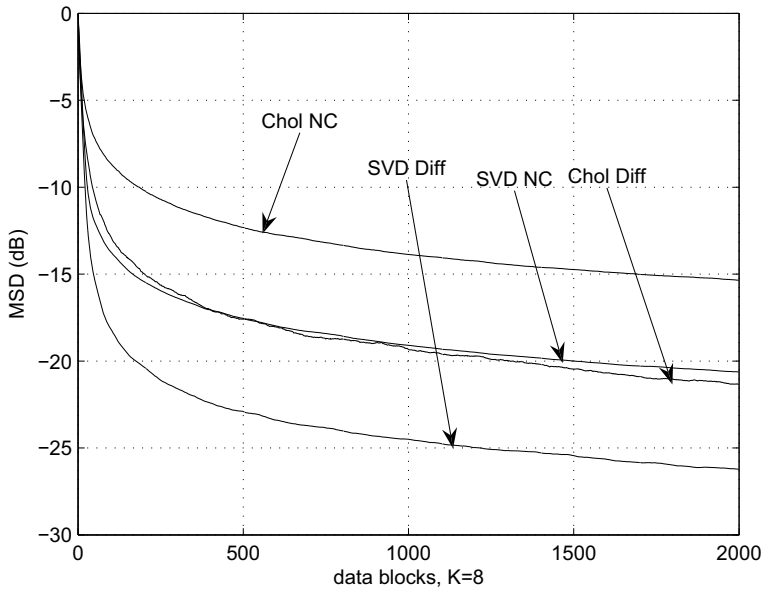

Figure 5.10: MSD at SNR 10 dB under best performance.

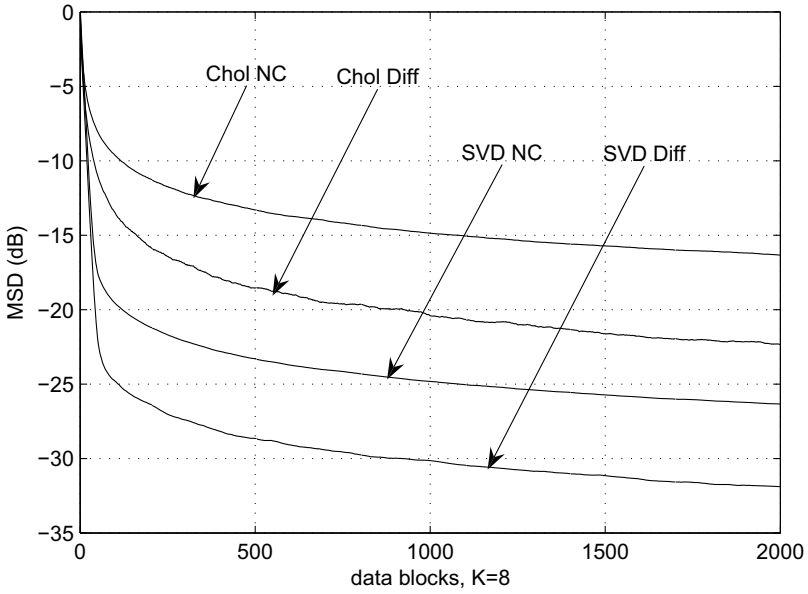

Figure 5.11: MSD at SNR 20 dB under best performance.

5.3.4 Effect of block size

Since it has been stated in the original works that the block size can effect the performance of the algorithm, the performance of the algorithms is also considered for various block sizes. The size is varied as $K = \{5, 8, 10, 15, 20\}$. The SNR is 20 dB. This is done for both algorithms separately. Here it is important to note that as the size of the data block increases, the total amount of data required for the same number of blocks also increases. For example, if a simulation is run for 1000 blocks of data then a block size of $K = 8$ would mean 8000 sensed values whereas a block size of $K = 20$ would mean 20,000 sensed values. Figures 5.12 and 5.13 give results for the BBRC algorithm. The algorithm fails badly for $K = 5$. However, for the remaining block sizes the algorithm performs almost similarly. The convergence speeds are nearly the same (see fig. 5.12) and the performance at steady-state is similar as well with only slight difference (see fig. 5.13). From fig. 5.13 it can be inferred that the best result, in every respect, is achieved when block size is just large enough to achieve a full rank input data matrix ($K = 8$ in this case). Thus, it is essential to estimate a tight upper bound for the size of the unknown vector in order to achieve good performance. Figures 5.14 and 5.15 give the results for the BBRS algorithm. Here, the performance improves gradually with increase in block size. However, the speed of convergence is slow for a large block size (see fig. 5.14) even though a larger block size gives better performance at steady-state (see fig. 5.15). Again it can be inferred that it is best to keep the block size reasonably small in order to achieve a good trade off between

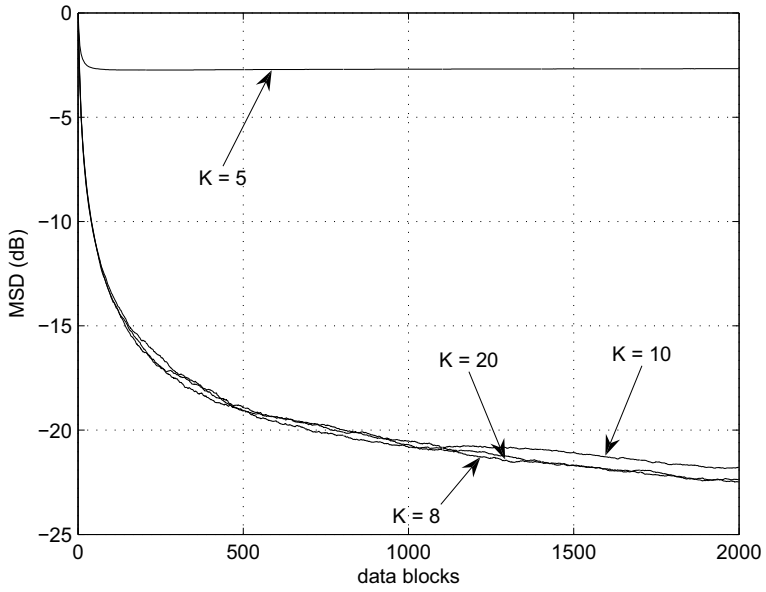

Figure 5.12: MSD at SNR 20 dB for varying K for BBRC.

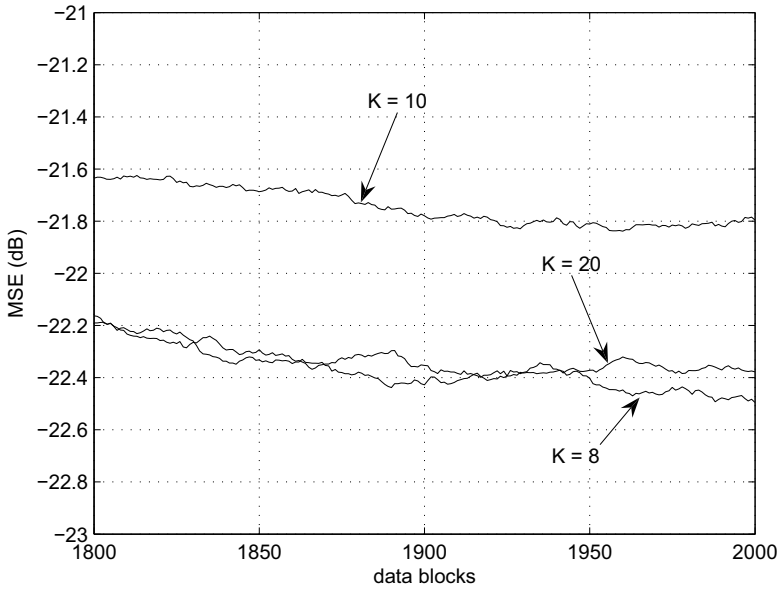

Figure 5.13: MSD at SNR 20 dB for varying K for BBRC (zoomed in at the end).

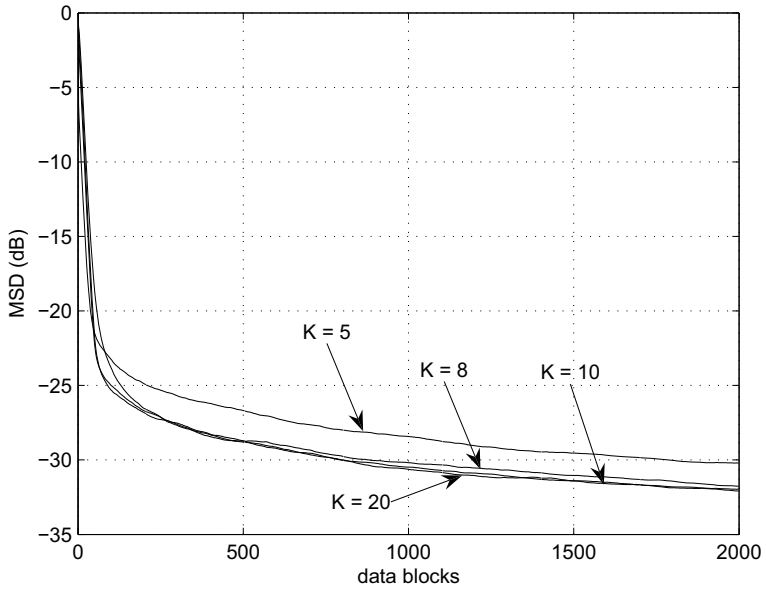

Figure 5.14: MSD at SNR 20 dB for varying K for BBRS.

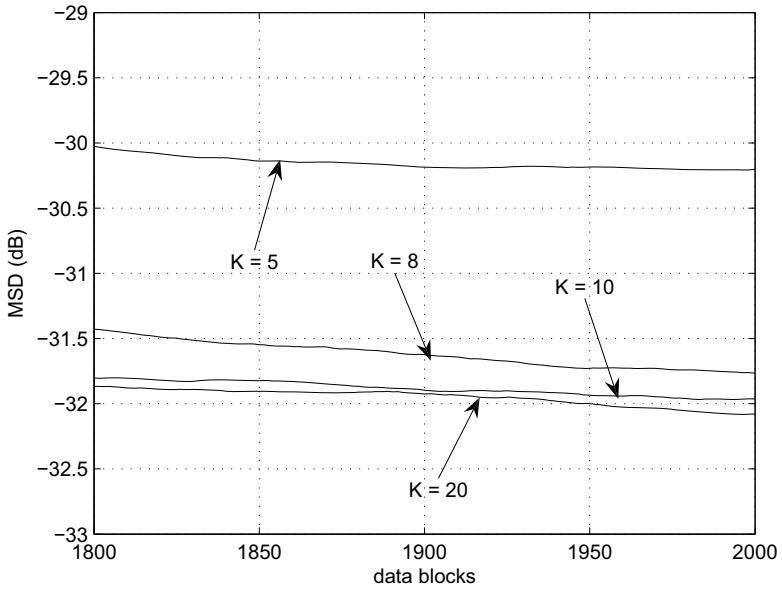

Figure 5.15: MSD at SNR 20 dB for varying K for BBRS (zoomed in at the end).

performance and speed of convergence, especially taking into account the fact that a larger block size would mean sensing more data for the same number of blocks.

5.3.5 Effect of network size

Next it is seen how the size of the network can have an effect on the performance of the algorithms. For this purpose, the size of the network is varied for $N = \{10 - 50\}$ while the forgetting factor is kept fixed at $\lambda = 0.9$ for the BBRS algorithm and variable for the BBRC algorithm. The block size is taken as $K = 8$. This performance comparison is also done for both algorithms separately. The number of neighbors for each node are increased gradually as the size of the network is increased. Figures 5.16 and 5.17 show results for the BBRC algorithm. The performance is not so good when $N = 10$ but the performance improves as N increases. The initial speed of convergence is similar as can be seen in fig. 5.16 but near steady-state, the bigger sized networks show slight improvement in performance, as shown in fig. 5.17. Figures 5.18 and 5.19 show the results for the BBRS algorithm. Here the trend is slightly different. It can be seen that the initial speed of convergence improves with increase in N (see fig. 5.18) but the improvement in performance is slightly lesser near steady-state (see fig. 5.19). Also, the difference in performance is lesser for larger networks, which is as expected.

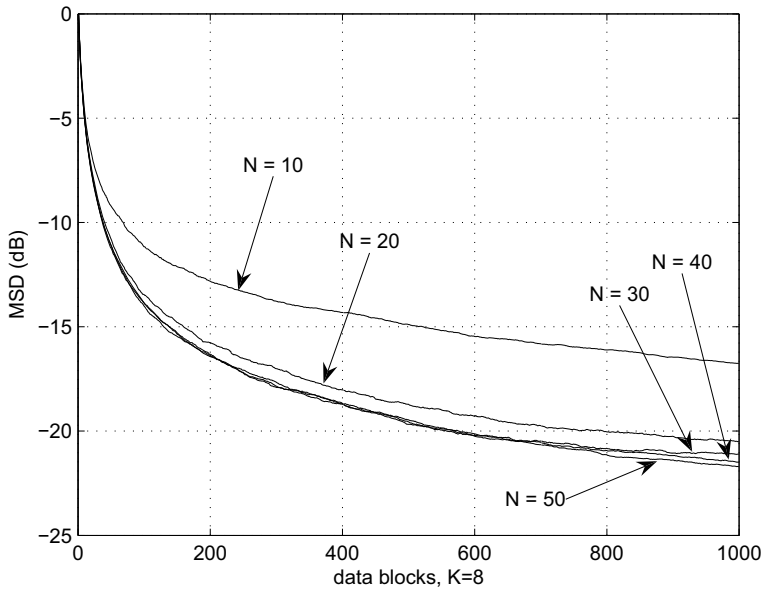

Figure 5.16: MSD at SNR 20 dB for varying network sizes for BBRC.

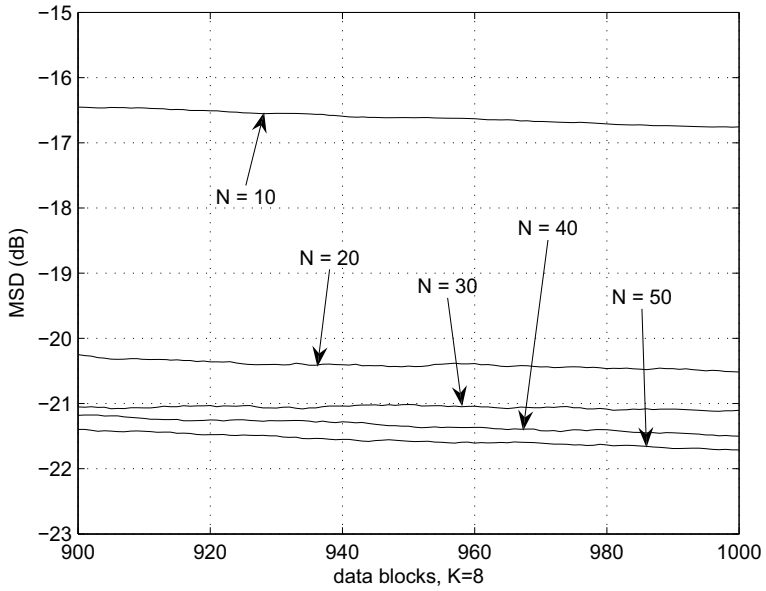

Figure 5.17: MSD at SNR 20 dB for varying network sizes for BBRC (zoomed in at the end).

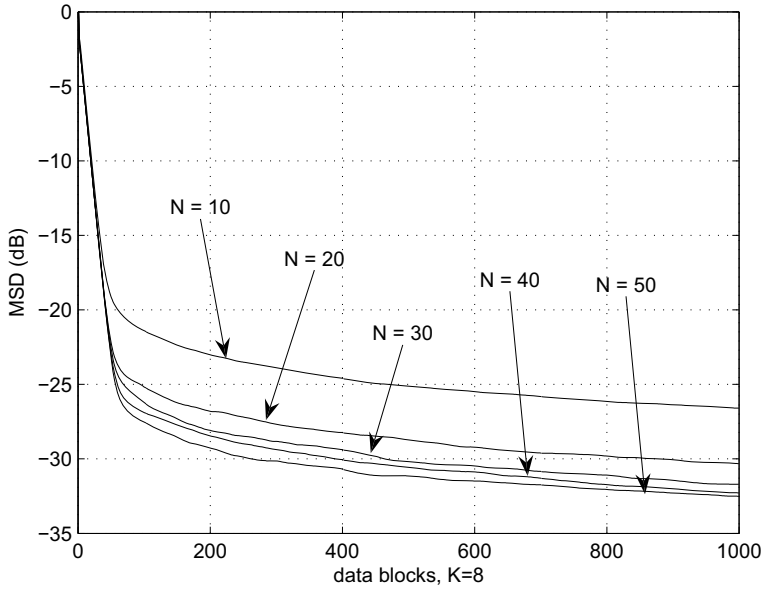

Figure 5.18: MSD at SNR 20 dB for varying network sizes for BBRS.

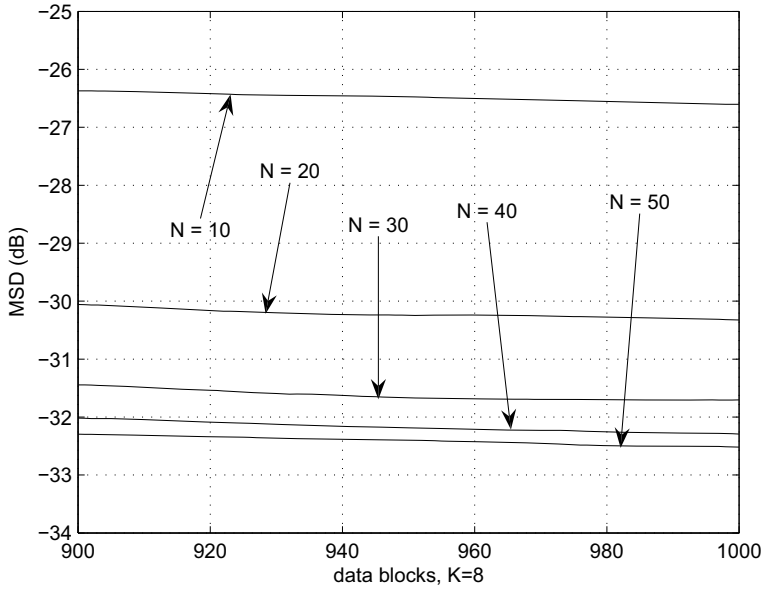

Figure 5.19: MSD at SNR 20 dB for varying network sizes for BBRS (zoomed in at the end).

5.3.6 Effect of node malfunction

Finally, it is shown how the performance can be effected if one or more nodes malfunction. Two different network sizes are chosen under two different SNR values to show how performance gets effected in this scenario. First, a network of 20 nodes is used and 5 nodes are switched off, resulting in a re-calibration of weights. The nodes with the maximum number of neighbors are switched off to see how badly the network performance might be effected. Results are shown for SNR 10 dB and 20 dB both, in figs 5.20 and 5.21 respectively. The network size is then increased to 50 nodes and again a quarter of the nodes are switched off, which is 13 nodes in this case. Results are shown in figs 5.22 and 5.23. The performance gets effected greatly for the BBRC algorithm at SNR 10 dB but there is not much difference in performance at SNR 20 dB and the difference gets even smaller when the network size is increased. The degradation is similar for the SVD based algorithm for all the cases, which shows that the SVD based algorithm is strongly dependent on the connectivity of the nodes. As expected, the overall performance improves with increase in network size. The effect of switched off nodes, however, is similar in both cases when the ratio of switched off nodes to the total nodes is the same.

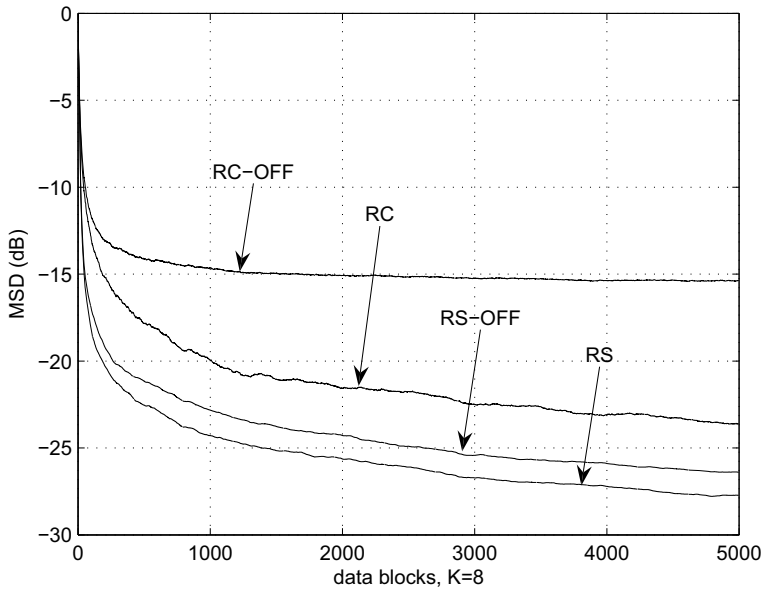

Figure 5.20: MSD at SNR 10 dB for 20 nodes when 5 nodes are off.

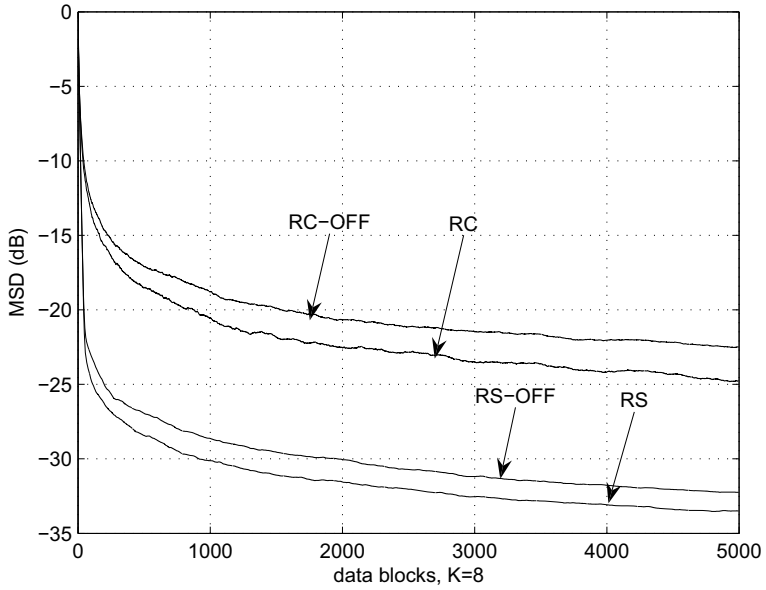

Figure 5.21: MSD at SNR 20 dB for 20 nodes when 5 nodes are off.

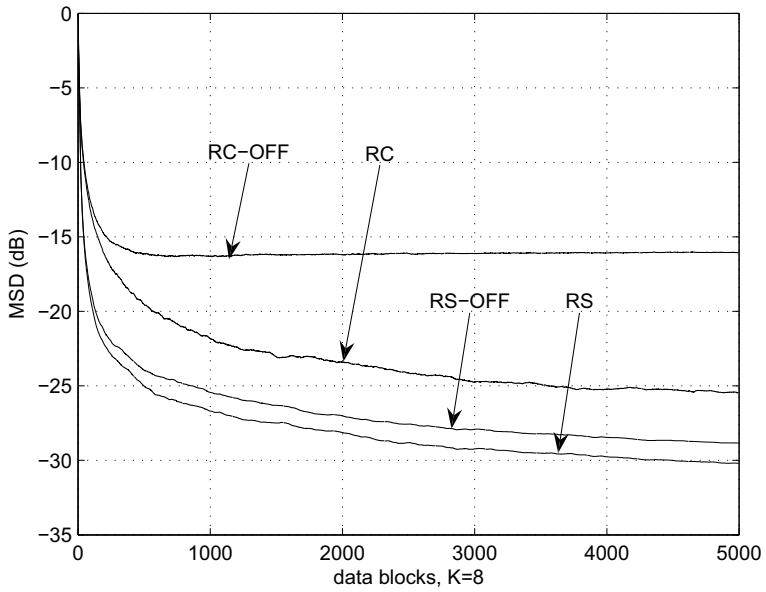

Figure 5.22: MSD at SNR 10 dB for 50 nodes when 13 nodes are off.

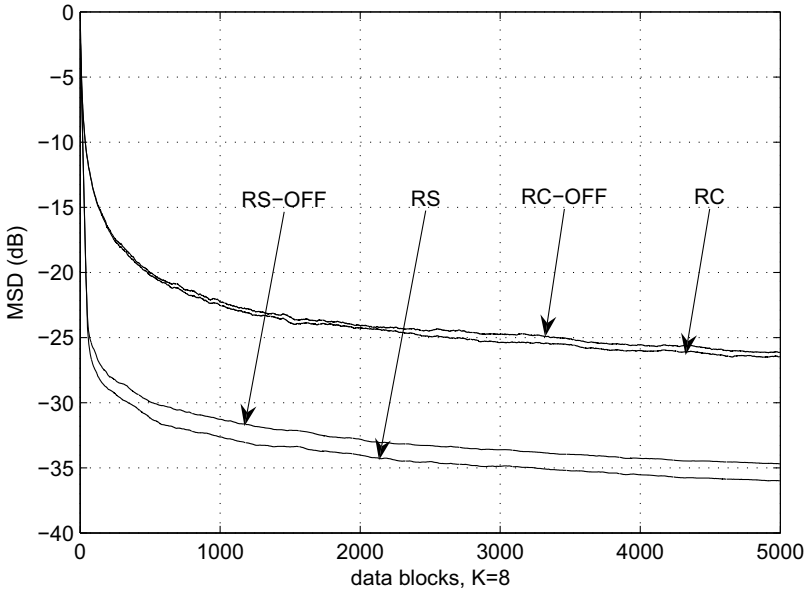

Figure 5.23: MSD at SNR 20 dB for 50 nodes when 13 nodes are off.

5.4 Conclusion

This work develops blind block recursive least squares algorithms based on Cholesky factorization and singular value decomposition (SVD). The algorithms are then used to estimate an unknown vector of interest in a wireless sensor network using cooperation between neighboring sensor nodes. Incorporating the algorithms in the sensor networks creates new diffusion based algorithms, which are shown to perform much better than the no cooperation case. The new algorithms have been tested using both a variable forgetting factor as well as a fixed forgetting factor. The two algorithms are named as the Diffusion Blind Block Recursive Cholesky (DBBRC) algorithm and the Diffusion Blind Block Recursive SVD (DBBRS) algorithm. Simulation results compare the two algorithms under different scenarios. It has been seen that the DBBRS algorithm performs much better but is also computationally very complex. Comparatively, the DBBRC algorithm is computationally less complex but does not perform as well. Also, the DBBRC algorithm performs better when the forgetting factor is variable whereas the DBBRS algorithm gives better results with a fixed forgetting factor. The value of the forgetting factor in the case of DBBRS does not effect the final result a great deal except for the slight variation in convergence speed and steady-state performance. It was also seen that the size of the data block has an effect on the performance of the algorithms. The speed of convergence slows down and this means that with a large block size, the amount of data required would also increase but the performance does not necessarily improve with an increase in

block size and generally, a small block size gives a better performance. Therefore, it is essential to estimate a very low upper bound to the size of the unknown vector so that the data block size is not so large. Next, it was noticed that an increase in the network size improves performance but the improvement gradually decreases for large sized networks. Finally, it was seen that switching off of nodes can slightly degrade the performance of the algorithm. In the case of the Cholesky based algorithm, the degradation can be severe at low SNR but the SVD based algorithm shows only a slight degradation.

CHAPTER 6

CONCLUSIONS AND FUTURE RECOMMENDATIONS

6.1 Conclusions

This dissertation proposes several algorithms for distributed parameter estimation over ad hoc wireless sensor networks. The non-blind algorithms are studied in detail and complete performance analyses are carried out, supported by simulation results to assess the performance of the proposed algorithms under various scenarios. The blind algorithms are formulated recursively and then applied for estimation in a wireless sensor network environment. The main contributions of the dissertation are listed below:

1. The variable step-size diffusion least mean square (VSSDLMS) algorithm is formulated after choosing the most suitable VSSLMS algorithm form the existing literature. A generalized formulation for the algorithm is also sug-

gested. A complete analysis of the algorithm is carried out and a detailed performance study is done for the proposed algorithm.

2. The noise-constrained diffusion least mean square (NCDLMS) algorithm is derived for estimation in a wireless sensor network. A generalized formulation is then suggested for the algorithm. Complete performance analysis is carried out including the effect of noise variance estimate mismatch. The bounds on performance have been tested and the robustness of the proposed algorithm is tested under different scenarios.

3. Two blind block estimation algorithms are formulated in the recursive sense inspired form existing algorithms and then applied to estimation in a wireless sensor network. Computational complexity of the algorithms is studied. Then the performance is studied through simulations and the effect of different variables on performance is shown.

6.2 Future Recommendations

Based on the results achieved in this work, several recommendations for future work are being suggested.

1. This work studies only time invariant environments. Time varying environments should be studied for the algorithms proposed in this work.

2. The proposed algorithms should be applied to applications such as radar detection and tracking, medical imaging, and spectrum sensing in order to

study their effectiveness.

3. The analysis of the non-blind algorithms has been done for Gaussian data only. The analysis can be extended for non-Gaussian data to see if it is possible to find a closed form solution.

4. The work in this dissertation is done using real-valued uncorrelated data only. The work should be extended to complex-valued and correlated data sets.

5. The analysis of the non-blind algorithms should be studied without using the independence assumptions.

6. A study of the parameters controlling the non-blind algorithms should be carried out to ascertain optimal performance measures.

7. Semi-blind algorithms should be studied in order to see if a trade-off between non-blind and blind algorithms can be reached for a practically implementable solution.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] I.F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges," *Ad Hoc Networks Journal*, vol. 2, pp. 351–367, Oct 2004.

[3] I.F. Akyildiz, D. Pompili, and T. Melodia, "Underwater Acoustic Sensor Networks: Research Challenges," *Ad Hoc Networks Journal*, vol. 3, pp. 257–279, Mar 2005.

[4] I.F. Akyildiz and E. Stuntebeck, "Wireless Underground Sensor Networks: Research Challenges," *Ad Hoc Networks Journal*, vol. 4, pp. 669–686, Nov 2006.

[5] I.F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative Spectrum Sensing in Cognitive Radio Networks: A Survey," *Physical Communication*, vol. 4, pp. 40–62, Mar 2011.

[6] Ning Xu, "A survey of sensor network applications," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[7] C.G. Lopes and A.H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4064–4077, 2007.

[8] C.G. Lopes and A.H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

[9] I.D. Schizas, Alejandro Ribeiro, and G.B. Giannakis, "Consensus in ad hoc WSNs with noisy linksPart I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[10] I.D. Schizas, Georgios B. Giannakis, Stergios I. Roumeliotis, and Alejandro Ribeiro, "Consensus in Ad Hoc WSNs With Noisy LinksPart II: Distributed Estimation and Smoothing of Random Signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1650–1666, Apr. 2008.

[11] G. Mateos, I.D. Schizas, and G.B. Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, 2009.

[12] G. Mateos, Ioannis D. Schizas, and Georgios B. Giannakis, "Performance Analysis of the Consensus-Based Distributed LMS Algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1–20, 2009.

[13] F.S. Cattivelli and A.H. Sayed, "Multi-level diffusion adaptive networks," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. 2009, pp. 2789–2792, IEEE.

[14] Cassio G. Lopes and A.H. Sayed, "Randomized incremental protocols over adaptive networks," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 3514–3517, 2010.

[15] F.S. Cattivelli and a.H. Sayed, "Diffusion LMS Strategies for Distributed Estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[16] N Takahashi, I Yamada, and A.H. Sayed, "Diffusion Least-Mean Squares With Adaptive Combiners: Formulation and Performance Analysis," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4795–4810, Sept. 2010.

[17] M.O. Bin Saeed, A. Zerguine, and S.A. Zummo, "Variable step-size least mean square algorithms over adaptive networks," in *10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA)*, 2010, number ISSPA, pp. 381–384.

[18] M.O. Bin Saeed, A. Zerguine, and S.A. Zummo, "Noise Constrained Diffusion Least Mean Squares over adaptive networks," in *IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. 2010, number i, pp. 288–292, IEEE.

[19] R. Olfati-Saber, J. Alex Fax, and Richard M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[20] A. Jadbabaie and A.S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[21] R. Olfati-Saber and R.M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[22] L Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, Sept. 2004.

[23] S Barbarossa, G Scutari, and T Battisti, "Distributed signal subspace projection algorithms with maximum convergence rate for sensor networks with topological constraints," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. 2009, pp. 2893–2896, IEEE.

[24] S. Barbarossa, G. Scutari, and T. Battisti, "Cooperative sensing for cognitive radio using decentralized projection algorithms," in *IEEE 10th Workshop on Signal Processing Advances in Wireless Communications, SPAWC*. 2009, pp. 116–120, IEEE.

[25] D.P. Spanos, R. Olfati-Saber, and R.M. Murray, "Dynamic consensus on mobile networks," in *IFAC world congress*, 2005.

[26] R. Olfati-Saber and J.S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *IEEE Conference on Decision and Control and European Control Conference. CDC-ECC'05.* 2005, number 1, pp. 6698–6703, IEEE.

[27] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proceedings of the 5th international conference on Information processing in sensor networks*. 2006, pp. 168–176, ACM.

[28] F.S. Cattivelli, C.G. Lopes, and A.H. Sayed, "Diffusion strategies for distributed Kalman filtering: formulation and performance analysis," in *Proc. Cognitive Information Processing*. 2008, pp. 36–41, Citeseer.

[29] F.S. Cattivelli and A.H. Sayed, "Diffusion mechanisms for fixedpoint distributed Kalman smoothing," in *Proc. EUSIPCO*, 2008, number 3, pp. 1–4.

[30] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," *IEEE Conference on Decision and Control*, pp. 5492–5498, 2007.

[31] H. Zhu, I.D. Schizas, and G.B. Giannakis, "Power-Efficient Dimensionality Reduction for Distributed Channel-Aware Kalman Tracking Using WSNs," *IEEE Transactions on Signal Processing*, vol. 57, no. 8, pp. 3193–3207, Aug. 2009.

[32] S.S. Ram, A. Nedic, and VV Veeravalli, "Stochastic incremental gradient descent for estimation in sensor networks," in *Asilomar Conference on Signals, Systems and Computers*. Nov. 2007, pp. 582–586, IEEE.

[33] C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[34] Federico S. Cattivelli and A.H. Sayed, "Diffusion LMS algorithms with information exchange," *42nd Asilomar Conference on Signals, Systems and Computers*, , no. 8, pp. 251–255, Oct. 2008.

[35] S. Stankovic, M. Stankovic, and D. Stipanovic, "Decentralized parameter estimation by consensus based stochastic approximation," in *IEEE Conference on Decision and Control (CDC)*. 2007, number 99, pp. 1535–1540, IEEE.

[36] A.H. Sayed and C.G. Lopes, "Distributed recursive least-squares strategies over adaptive networks," in *Asilomar Conference on Signals, Systems and Computers*. 2006, number 1, pp. 233–237, IEEE.

[37] F.S. Cattivelli, C.G. Lopes, and A.H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, 2008.

[38] M.G. Rabbat, R.D. Nowak, and J.a. Bucklew, "Generalized consensus computation in networked systems with erasure links," *IEEE 6th Workshop on Signal Processing Advances in Wireless Communications.*, , no. 7, pp. 1088–1092, 2005.

[39] Dimitri P. Bertsekas and John N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, 1997.

[40] M.G. Rabbat and R.D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 798–808, Apr. 2005.

[41] Dimitri P. Bertsekas, *Nonlinear Programming: 2nd Edition*, Athena Scientific, 1999.

[42] Angelia Nedic and D.P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM Journal of Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[43] I.D. Schizas, G. Mateos, and G.B. Giannakis, "Distributed LMS for Consensus-Based In-Network Adaptive Processing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365–2382, June 2009.

[44] Paul Tseng, "Applications of a splitting algorithm to decomposition in convex programming and variational inequalities," *SIAM Journal on Control and Optimization*, vol. 29, no. 1, pp. 119–138, 1991.

[45] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976.

[46] J.A. Bazerque and G.B. Giannakis, "Distributed spectrum sensing for cognitive radios by exploiting sparsity," in *42nd Asilomar Conference on Signals, Systems and Computers*. Oct. 2008, pp. 1588–1592, IEEE.

[47] H. Zhu, Alfonso Cano, and Georgios B. Giannakis, "Distributed Demodulation Using Consensus Averaging in Wireless Sensor Networks," *Asilomar Conference on Signals, Systems and Computers*, pp. 1170–1174, Oct. 2008.

[48] Pedro A. Forero, Alfonso Cano, and Georgios B. Giannakis, "Consensus-based distributed expectation-maximization algorithm for density estimation and classification using wireless sensor networks," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. Mar. 2008, pp. 1989–1992, IEEE.

[49] S.Y. Tu and A.H. Sayed, "Mobile adaptive networks with self-organization abilities," in *7th International Symposium on Wireless Communication Systems (ISWCS)*. 2010, number 4, pp. 379–383, IEEE.

[50] R.H. Kwong and E.W. Johnston, "A variable step size LMS algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 7, pp. 1633–1642, 1992.

[51] Tyseer Aboulnasr and K Mayyas, "A robust variable step-size LMS-type algorithm: analysis and simulations," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 631–639, 1997.

[52] M.H. Costa and J.C.M. Bermudez, "A robust variable step size algorithm for LMS adaptive filters," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. 2006, vol. 3, pp. 93–96, IEEE.

[53] V.J. Mathews and Z. Xie, "A stochastic gradient adaptive filter with gradient adaptive step size," *IEEE Transactions on Signal Processing*, vol. 41, no. 6, pp. 2075–2087, June 1993.

[54] A.I. Sulyman and A. Zerguine, "Convergence and steady-state analysis of a variable step-size NLMS algorithm," *Signal Processing*, vol. 83, no. 6, pp. 1255–1273, June 2003.

[55] C.G. Lopes and J.C.M. Bermudez, "Evaluation and design of variable step size adaptive algorithms," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. 2001, vol. 6, pp. 3845–3848, IEEE.

[56] T.Y. Al-Naffouri and A.H. Sayed, "Transient analysis of adaptive filters with error nonlinearities," *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 653–663, Mar. 2003.

[57] Ali H. Sayed, *Fundamentals of Adaptive Filtering*, John Wiley & Sons, Inc., Hoboken, NJ, 2003.

[58] R.H. Koning, H. Neudecker, and T. Wansbeek, "Block Kronecker products and the vecb operator 1," *Linear algebra and its applications*, vol. 149, pp. 165–184, 1991.

[59] Y. Wei, S.B. Gelfand, and J.V. Krogmeier, "Noise-constrained least mean squares algorithm," *IEEE Transactions on Signal Processing*, vol. 49, no. 9, pp. 1961–1970, 2001.

[60] Marie Dufflo, *Random iterative Models*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[61] Y. Sato, "A method of self-recovering equalization for multilevel amplitude-modulation systems," *IEEE Transactions on Communications*, vol. 23, no. 6, pp. 679–682, June 1975.

[62] G. Xu, H. Liu, L. Tong, and T. Kailath, "A least-squares approach to blind channel identification," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 2982–2993, 1995.

[63] K. Abed-Meraim, Wanzhi Qiu, and Yingbo Hua, "Blind system identification," *Proceedings of the IEEE*, vol. 85, no. 8, pp. 1310–1322, 1997.

[64] L. Tong and S. Perreau, "Multichannel blind identification: from subspace to maximum likelihood methods," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 1951–1968, 1998.

[65] A. Scaglione, G.B. Giannakis, and S. Barbarossa, "Redundant filterbank precoders and equalizers. II. Blind channel estimation, synchronization, and direct equalization," *IEEE Transactions on Signal Processing*, vol. 47, no. 7, pp. 2007–2022, July 1999.

[66] J.H. Manton and W.D. Neumann, "Totally blind channel identification by exploiting guard intervals," *Systems & Control Letters*, vol. 48, no. 2, pp. 113–119, Feb. 2003.

[67] D.H. Pham and J.H. Manton, "A subspace algorithm for guard interval based channel identification and source recovery requiring just two received blocks," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. 2003, vol. 4, pp. IV–317, IEEE.

[68] Borching Su and P. P. Vaidyanathan, "A Generalized Algorithm for Blind Channel Identification with Linear Redundant Precoders," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–14, 2007.

[69] Jinho Choi and Cheng-chew Lim, "A Cholesky Factorization Based Approach for Blind FIR Channel Identification," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1730–1735, Apr. 2008.

Vitae

- Muhammad Omer Bin Saeed
- Nationality: Pakistani
- Current Address: P.O. Box 8073, K.F.U.P.M., Dhahran 31261, Saudi Arabia
- Permanent Address: House 9, Street 67, F-8/3, Islamabad 44000, Pakistan
- Telephone: (+966) 569 130 240
- Email: *mobs_1981@yahoo.com*
- Born in Rawalpindi, Pakistan on July 13, 1981
- Received Bachelor of Engineering (B.E.) in Electrical Engineering from College of Electrical & Mechanical Engineering, National University of Sciences & Technology, Rawalpindi, Pakistan in 2003.
- Received Master of Science (M.S.) in Electrical Engineering from College of Electrical & Mechanical Engineering, National University of Sciences & Technology, Rawalpindi, Pakistan in 2005.
- Joined King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia as a Lecturer-B in September 2005.
- Completed Ph.D. in Electrical Engineering in June 2011.