ONLINE LEARNING OVER DISTRIBUTED NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Muhammed Ömer Sayın
July, 2015

Online Learning Over Distributed Networks
By Muhammed Ömer Sayın
July, 2015

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Süleyman Serdar Kozat (Advisor)

Prof. Dr. Orhan Arıkan

Prof. Dr. Aydın Alatan

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

ONLINE LEARNING OVER DISTRIBUTED NETWORKS

Muhammed Ömer Sayın
M.S. in Electrical and Electronics Engineering
Advisor: Assoc. Prof. Süleyman Serdar Kozat
July, 2015

We study online learning strategies over distributed networks. Here, we have a distributed collection of agents with learning and cooperation capabilities. These agents observe a noisy version of a desired state of the nature through a linear model. The agents seek to learn this state by also interacting with each other yet the communication load plays significant role. To this end, we propose compressive diffusion strategies that extract the compressed information from the diffused data. Agents can compress the information into a scalar or a single bit, i.e., a substantial reduction in the communication load. Importantly, we show that agents can achieve a comparable performance to the conventional diffusion strategies that require the direct diffusion of information without compression and with infinite precision. We also examine which information to disclose and how to utilize them optimally in the mean-square-error (MSE) sense. Note that all the well-known distributed learning strategies achieve suboptimal learning performance in the MSE sense. Hence, we provide algorithms that achieve distributed minimum MSE (MMSE) performance over an arbitrary network topology based on the aggregation of information at each agent. This approach differs from the diffusion of information across network, i.e., exchange of local estimates. Notably, exchange of local estimates is sufficient only over the certain network topologies. For these networks, we also propose strategies that achieve the distributed MMSE performance through the diffusion of information. Hence, we can substantially reduce the communication load while achieving the best possible MSE performance. Finally, for practical implementations we provide approaches to reduce the complexity of the algorithms through the time-windowing of the observations.

Keywords: Online learning, compressive diffusion, distributed networks, big data.

ÖZET

DAĞITILMIŞ AĞ ÜZERİNDE ONLINE ÖĞRENME

Muhammed Ömer Sayın
Elektrik Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Doç. Dr. Süleyman Serdar Kozat
Temmuz, 2015

Dağıtılmış ağlar üzerinde online öğrenme stratejileri üzerinde çalışmaktayız. Burada öğrenme ve işbirliği kabiliyetine sahip ajanların dağıtılmış bir koleksiyonuna sahibiz. Bu ajanlar istenilen state parametresinin gürültülü versiyonunu gözlemlemekteler. State öğrenmeyi amaçlayan ajanlar aynı zamanda birbirleri ile etkileşime girmektedir fakat iletişim yükü önemli bir rol oynamaktadr. Bu amaçla yayılan veriden sıkıştırılmış bilgiyi çıkarabilen sıkıştırılmış yayınım stratejilerini öneriyoruz. Ajanlar bilgiyi bir skalere veya bir bite kadar sıkıştırabilirler. Bir diğer ifadeyle iletişim yükünü büyük ölçüde azaltabilirler. Önemli olarak ajanlar sonsuz hassasiyet gerektiren ve herhangi bir sıkıştırma işlemi kullanmayan geleneksel yayınım stratejileri ile karşılatırabilinir performans sergileyebilirler. Ayrıca hatanın karesinin ortalaması açısından en iyileştirilmisi için hangi bilginin yayılması ve nasıl kullanılması gerektiğini inceliyoruz. Bilinen tüm dağıtılmış öğrenme stratejileri hatanın karesinin ortalaması açısından suboptimaldir. Bu yüzden herhangi bir ağ topolojisi üzerinde bilginin her bir ajanda toplanmasını kullanan, en küçük dağıtılmış hatanın karesinin ortalamasını başarabilen algoritmalar sağlyoruz. Bu yaklaşım yerel kestirimlerin değiş tokuş yapıldığı, ağ üzerinde bilginin yayıldığı, yaklaşımlardan farklılık göstermektedir. Fakat yerel kestirim parametrelerinin yayınımı sadece özel ağ topolojileri üzerinde yeterli olmaktadır. Bu ağlar için bilginin yayınımını kullanan ve en küçük dağıtılmış hatanın karesinin ortalamasını başarabilen algoritmalar da öneriyoruz. Böylece olası en iyi performansı başarırken iletişim yükünü büyük ölçüde azaltabiliriz. Son olarak pratik uygulamalar için gözlemlerin zaman içerisinde pencerelenmesini kullanan yaklaşımlar sağlıyoruz.

Anahtar sözcükler: Online öğrenme, sıkıştırılmış yayılma, dağıtılmış ağlar, büyük veri.

Acknowledgement

I acknowledge that this thesis is supported by TUBİTAK BİDEB 2228-A and 2210-C Scholarship Programmes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Distributed network of nodes provides enhanced performance for several different applications, such as source tracking, environment monitoring and source localization [1–4]. In such a distributed network, each node encounters possibly a different statistical profile, which provides broadened perspective on the monitored phenomena. In general, we could reach the best estimate with access to all observation data across the whole network since the observation of each node carries valuable information [5]. In the distributed adaptive estimation framework, we distribute the processing over the network and allow the information exchange among the nodes so that the parameter estimate of each node converges to the best estimate [4, 6].

In the distributed architectures, one can use different approaches to regulate the information exchange among nodes such as the diffusion implementations [6–11]. The generic diffusion implementation defines a communication protocol in which only the nodes from a certain neighborhood could exchange information with each other [1, 6–11]. In this framework, each node uses a local adaptive algorithm and improves its parameter estimation by fusing its information with the diffused parameter estimations of the neighboring nodes. Via this information sharing, the diffusion approach provides robustness against link failures and changing network topologies [6]. However, diffusion of the parameter

vectors within the neighborhoods results in high amount of communication load. For example, in a typical diffusion network of N nodes the overall communication burden is given by $N \times M$ where M is the size of the diffused vector, which implies that the size of the diffused information has a multiplicative impact on the overall communication burden. Additionally, in a wireless network, the neighborhood size also plays a crucial role on the overall communication load since the larger the neighborhood is, the more power is required in the transmission of the information [1–4].

We study the compressive diffusion strategies that achieve a better trade-off in terms of the amount of cooperation and the required communication load [12]. Unlike the full diffusion configuration, the compressed diffusion approach diffuses a single-bit of information or a reduced dimensional data vector achieving an impressive reduction in the communication load, i.e., from a full vector to a single bit or to a single scalar. The diffused data is generated through certain random projections of the local parameter estimation vector. Then, the neighboring nodes adaptively construct the original parameter estimations based on the diffused information and fuse their individual estimates for the final estimate. In this sense, this approach reduces the communication load in the spirit of the compressive sensing [12, 13]. The compression is lossy since we do not assume any sparseness or compressibility on the parameter estimation vector [13, 14]. However, the compressive diffusion approach achieves comparable convergence performance to the full diffusion configurations. Since the communication load increases far more in the large networks or the networks where the paths among the nodes are relatively longer, the compressive diffusion strategies play a crucial role in achieving comparable convergence performance with significantly reduced communication load.

There exist several other approaches that seek to reduce the communication load in distributed networks. In [15, 16] and [17], the authors propose the partial diffusion strategies where the nodes diffuse only selected coefficients of the parameter estimation vector. In [18], the dimension of the diffused information is reduced through the Krylov subspace projection techniques in the set-theoretic estimation framework. In [19], within a predefined neighborhood, the parameter

estimate is quantized before the diffusion in order to avoid unlimited bandwidth requirement. In [20], the nodes transmit the sign of the innovation sequence in the decentralized estimation framework. In [21], in a consensus network, the relative difference between the states of the nodes is exchanged by using a single bit of information. As distinct from the mentioned works, the compressive diffusion strategies substantially compress the whole diffused information and extract the information from the compressed data adaptively [12].

In this study, we provide a complete performance analysis for the compressive diffusion strategies, which demonstrates comparable convergence performance of the compressed diffusion to the full information exchange configuration. We note that studying the performance of distributed networks with compressive diffusion strategies is not straight-forward since adaptive extraction of information from the diffused data brings in an additional adaptation level. Moreover, such a theoretical analysis is rather challenging for the single-bit diffusion strategy due to the highly nonlinear compression. However, we analyze the transient, steady-state and tracking performance of the configurations in which the diffused data is compressed into a scalar or a single-bit. We also propose a new adaptive combination method improving the performance for any conventional combination rule. In the compressive diffusion framework, we fuse the local estimates with the adaptively extracted information from substantially compressed diffusion data. The extracted information carries relatively less information than the original data. Hence, we introduce "a confidence parameter" concept, which adds one more freedom-of-dimension in the combination matrix. The confidence parameter determines how much we are confident with the local parameter estimation. Through the adaptation of the confidence parameter, we observe enormous enhancement in the convergence performance of the compressive diffusion strategies even for relatively long filter lengths.

Additionally, online learning, estimation, and prediction over distributed networks have recently attracted substantial attention in diverse disciplines due to their natural advantages using cooperation (see e.g., [6–8, 22–31] and references therein). In these approaches, we have agents that observe as well as process data in order to learn the true state of the underlying problem, as an example,

say that we have a collection of buses connected via transmission lines on an electric grid where the physical state is the voltage magnitude and angle [32]. These agents can cooperate within the network in order to alleviate the learning process in a fully distributed manner [24]. Notably, the agents can respond to streaming data in an online manner by disclosing information among each other. This framework is conveniently used to model highly complex structures from biological systems to social and economical networks [28–31]. Although there is an extensive literature on this field, we still have significant and yet unexplored problems for disclosure and utilization of information among agents.

There exist several different approaches to control the information sharing between the agents. As an example, widely used consensus and diffusion implementations benefit from the propagation of information across the network. In the consensus implementation, after collecting and processing the local measurements, the agents seek to reach a "consensus" over the collected data [33, 34]. Recently, average consensus and gossip algorithms [35, 36] have been applied extensively in the multi-agent systems [37,38] or in the distributed estimation frameworks [23,34]. Additionally, the diffusion approach seeks to respond to streaming data in an online manner [6, 24]. At each instant, the agents process their observation data locally, disclose their state estimate to the neighboring agents, and enhance the performance through the received estimated parameters [6]. Correspondingly, the diffusion based approaches have been extensively applied in distributed detection [39], estimation [6–8,25], learning [11] and optimization [40,41] frameworks. In [42], the authors show that the diffusion strategies outperform the consensus based strategies in terms of mean-square error (MSE) performance.

As a common property, consensus and diffusion approaches require the agents to disclose the local estimate to the neighboring agents, hence information can propagate across the network. However, which information to disclose among the agents for the optimal learning performance, such as in the MSE sense, is an unsolved problem yet. Hence, the one of the main objective of this thesis is to study optimal information sharing strategies for the minimum MSE performance over distributed networks. Additionally, in both consensus and diffusion based iterations, the agents construct the final estimate by linearly combining the received

information such as the estimates. To this end, the agents generally utilize certain static combination rules, e.g., the uniform rule [43], the Laplacian rule [44] and the Metropolis rule [45]. However, we emphasize that the agents should choose the cooperation rule properly otherwise the cooperation may influence the performance adversely in general [25, 46–48]. As an example, if the statistical profile of the measurement data varies over the network, i.e., each agent observes diverse signal-to-noise ratios, the cooperation rules ignoring the variation in noise, e.g., the uniform, Laplacian, and Metropolis rules, yield severe decline in the estimation performance [24]. Particularly, in such cases the agents can perform better without cooperation [24]. Hence, we can enhance the performance significantly by adjusting the cooperation rule according to the prior information about the statistical profiles across the network and the network topology [25, 48, 49].

In the diffusion based approaches, there are several studies to choose the combination weights with the prior information in terms of certain performance measures. In [25], the authors optimize the weights numerically in terms of the steady-state average mean-square deviation over all the network through the prior information about the statistical profiles of the measurements of the agents. Similarly, through certain approximations, in [48], the authors formulate a convex optimization problem to determine the combination weights in time. In [49] and [50], the authors adjust the combination weights for the transient and steady state phases separately and formulate a hypothesis testing strategy to switch from one phase to the other. However, we point out that these online strategies do not achieve the optimal learning performance in MSE sense continuously at all phases. Correspondingly, there are also a few strategies to increase the convergence rate in the average consensus algorithms [51–53], however, different from these, we aim to formulate online algorithms for streaming data.

In this study, we also seek to formulate the optimal learning strategy over distributed networks in the MSE sense. In particular, we design which information to disclose and how to utilize them for the minimum MSE (MMSE) performance. Over a sparsely connected network, the agents observe a noisy version of the underlying state and can exchange information with only certain agents at each time instant. For that model, we introduce a comprehensive cost measure considering

the transmission of information over the hops due to partial connections. Correspondingly, we formulate the MMSE estimator using prior information about the statistical profiles and the network topology. For the case of jointly Gaussian state and noise signals, we derive a consensus-like iterative algorithm, i.e., the optimal distributed online learning (ODOL) algorithm, based on the aggregation of information at each agent. We point out that the ODOL algorithm achieves the linear MMSE (LMMSE) performance for other cases.

Notably, the ODOL algorithm is different from the conventional algorithms that benefit from the propagation of information through the disclosure of the local estimates. However, we can achieve the MMSE performance through the disclosure of local estimates *only* over certain network topologies. We analytically show that over the tree networks involving cell structures we can achieve the MMSE performance through the disclosure of the local estimates. To this end, we formulate the optimal and efficient distributed online learning (OEDOL) algorithm. The OEDOL algorithm achieves the MMSE performance over the tree networks and can reduce the communication load tremendously by using propagation of information across the network instead of aggregation of information at each agent. Note that for an arbitrary network topology, we can construct such network connections by eliminating certain communication links. Hence, by exploiting an arbitrary network, the proposed algorithms achieve the MMSE performance asymptotically. Additionally, we propose sub-optimal versions of the algorithms using the time windowing of the observation set in order to reduce the complexity for practical applications. These sub-optimal versions have consensus-like iterations with time invariant weights that can be calculated beforehand.

The remainder of the thesis is organized as follows. In Chapter 2, we introduce the compressive diffusion strategies over distributed networks for reduced communication load. In Chapter 3, we provide the communication efficient channel estimation strategy over distributed networks as an application of the compressive diffusion strategies. We study the optimal static state estimation strategies over distributed networks in Chapter 4. In Chapter 5, we provide an implementation of the optimal distributed learning strategies for Big Data applications.

Notation: Bold lower (or upper) case letters denote column vectors (or matrices). For a vector $\boldsymbol{a}$ (or matrix $\boldsymbol{A}$), $\boldsymbol{a}^T$ (or $\boldsymbol{A}^T$) is its ordinary transpose. $\|\cdot\|$ and $\|\cdot\|_{\boldsymbol{A}}$ denote the L_2 norm and the weighted L_2 norm with the matrix $\boldsymbol{A}$, respectively (provided that $\boldsymbol{A}$ is positive-definite). We work with real data for notational simplicity. Here, $\mathrm{Tr}(\boldsymbol{A})$ denotes the trace of the matrix $\boldsymbol{A}$. The operator $\mathrm{col}\{\cdot\}$ produces a column vector or a matrix in which the arguments of $\mathrm{col}\{\cdot\}$ are stacked one under the other. For a matrix argument, $\mathrm{diag}\{\boldsymbol{A}\}$ operator constructs a diagonal matrix with the diagonal entries of $\boldsymbol{A}$ and for a vector argument, it creates a diagonal matrix whose diagonal entries are elements of the vector. The terms of the vector $\mathbf{1}$ (and $\mathbf{0}$) are all 1s (and 0s) and the size of the vector could be understood from the context. We use calligraphic letters for random variables and underlined calligraphic letters for random vectors, e.g., $\mathcal{X}$ and $\underline{\mathcal{X}}$. Bold calligraphic letters, e.g., $\boldsymbol{\mathcal{Z}}$, denote a set of random variables. For a random variable $\mathcal{X}$ (or vector $\underline{\mathcal{X}}$), $E[\mathcal{X}]$ (or $E[\underline{\mathcal{X}}]$) represents its expectation. We work with real data for notational simplicity. For the vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$, $\boldsymbol{a} \odot \boldsymbol{b}$ is the Hadamard (element-wise) product. The operator $\otimes$ takes the Kronecker tensor product of two matrices.

Chapter 2

Compressive Diffusion Strategies

In this chapter, for Gaussian regressors, we analyze the transient, steady-state and tracking performance of scalar and single-bit diffusion techniques. We demonstrate that our theoretical analysis accurately models the simulated results. We propose a new adaptive combination method for compressive diffusion strategies, which achieves a better trade-off in terms of the transient and steady state performance. We provide numerical examples showing the enhanced convergence performance with the new adaptive combination method in our simulations.

2.1 Distributed Network

Consider a network of N nodes where each node i measures[1] $d_{i,t}$ and $\boldsymbol{u}_{i,t} \in \mathbb{R}^M$ related via the true parameter vector $\boldsymbol{w_o} \in \mathbb{R}^M$ through a linear model

$$d_{i,t} = \boldsymbol{w_o}^T \boldsymbol{u}_{i,t} + v_{i,t},$$

where $v_{i,t}$ denotes the temporally and spatially white noise. We assume that the regression vector $\boldsymbol{u}_{i,t}$ is spatially and temporally uncorrelated with the other regressors and the observation noise. If we know the whole temporal and spatial

[1]Although we assume a time invariant unknown system vector $\boldsymbol{w_o}$, we also provide the tracking performance analysis for certain non-stationary models later in the paper.

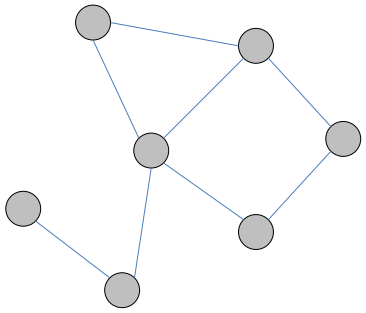


Figure 2.1: Distributed network of nodes and the neighborhood $\mathcal{N}_i$.

data overall network, then we can obtain the parameter of interest $\boldsymbol{w_o}$ by minimizing the following global cost with respect to the parameter estimate $\boldsymbol{w}$ [6]:

$$J_{\text{glob}}(\boldsymbol{w}) = \sum_{i=1}^{N} E\left[(d_{i,t} - \boldsymbol{w}^T \boldsymbol{u}_{i,t})^2\right]. \tag{2.1}$$

The stochastic gradient update for (2.1) leads to the global least-mean square (LMS) algorithm as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \mu \sum_{i=1}^{N} \boldsymbol{u}_{i,t}\left(d_{i,t} - \boldsymbol{u}_{i,t}^T \boldsymbol{w}_t\right), \tag{2.2}$$

where $\mu > 0$ is the step size [7]. Note that (2.2) brings a significant communication burden by gathering the network-wise information in a central processing unit. Additionally, centralized approach is not robust against link failures and changing network statistics [4, 6]. On the other hand, in the diffusion implementation framework, we utilize a protocol in which each node i can only exchange information with the nodes from its neighborhood $\mathcal{N}_i$ (with the convention $i \in \mathcal{N}_i$) [6, 7]. This protocol distributes the processing to the nodes and provides tracking ability for time-varying statistical profiles [6].

Assuming the inner-node links are symmetric, we model the distributed network as an undirected graph where the nodes and the communication links correspond to its vertices and edges, respectively (See Fig. 2.1). In the distributed network, each node employs a local adaptation algorithm and benefits from the information diffused by the neighboring nodes in the construction of the final estimate [6–9]. For example, in [6], nodes *diffuse their parameter estimate* to the

neighboring nodes and each node i performs the LMS algorithm given as

$$\boldsymbol{w}_{i,t+1} = (\boldsymbol{I} - \mu_i \boldsymbol{u}_{i,t}\boldsymbol{u}_{i,t}^T)\boldsymbol{\varphi}_{i,t} + \mu_i d_{i,t}\boldsymbol{u}_{i,t}, \tag{2.3}$$

where $\mu_i > 0$ is the local step-size. The intermediate parameter vector $\boldsymbol{\varphi}_{i,t}$ is generated through

$$\boldsymbol{\varphi}_{i,t} = \sum_{j \in \mathcal{N}_i} \gamma_{i,j} \boldsymbol{w}_{j,t}$$

with $\gamma_{i,j}$'s are the combination weights such that $\sum_{j=1}^{N} \gamma_{i,j} = 1$ for all $i \in \{1, \cdots, N\}$. For a given network topology, the combination weights are determined according to certain combination rules such as uniform [43], the Metropolis [45, 54], relative-degree rules [8] or adaptive combiners [55].

We note that in (2.3) we could assign $\boldsymbol{\varphi}_{i,t}$ as the final estimate in which we adapt the local estimate through the local observation data and then we fuse with the diffused estimates to generate the final estimate. In [7], authors examine these approaches as combine-than-adapt (CTA) and adapt-than-combine (ATC) diffusion strategies, respectively. In this paper, we study the ATC diffusion strategy, however, the theoretical results hold for both the ATC and the CTA cases for certain parameter changes provided later in the paper.

We emphasize that the diffusion of the parameter estimation vector also brings in high amount of communication load. In the next section, we introduce the compressive diffusion strategies enabling the adaptive construction of the required information from the reduced dimensional diffused information.

2.2 Compressive Diffusion

We seek to estimate the parameter of interest $\boldsymbol{w_o}$ through the *reduced dimension information exchange* within the neighborhoods. To this end, in the compressed diffusion approach, unlike the full diffusion scheme, we exchange a significantly reduced amount of information. The diffused information is generated through a certain projection operator, e.g., a time-variant vector $\boldsymbol{c}_t$, by linearly transforming the parameter vector, e.g., $\boldsymbol{w}_{i,t}$. In particular, node i diffuses $\boldsymbol{c}_t^T \boldsymbol{w}_{i,t}$ instead

of the whole parameter vector $\boldsymbol{w}_{i,t}$ in the scalar diffusion scheme. We might also use a projection matrix, e.g., $\mathcal{C}_t \in \mathbb{R}^{M\times p}$, such that $\dim\{\mathcal{C}_t^T \boldsymbol{w}_{i,t}\} \ll \dim\{\boldsymbol{w}_{i,t}\}$ or $p \ll M$. Then the neighboring nodes of i can generate an estimate $\boldsymbol{a}_{i,t}$ of $\boldsymbol{w}_{i,t}$ through the exchanged information by using an adaptive estimation algorithm as explained later in this chapter and in [12]. We emphasize that the estimates $\boldsymbol{a}_{i,t}$'s are the constructed information using the diffused information, not the actual diffused information. Hence, the diffused information might have far smaller dimensions than the parameter estimation vector, which reduces the communication load significantly.

We note that the projection operator plays a crucial role in the construction of $\boldsymbol{a}_{i,t}$. Hence we constrain the projection operator to span the whole parameter space in order to avoid biased estimate of the original parameters [12]. Based on this constraint, we can construct the projection operator through the pseudo-random number generators (PRNG), which generates a sequence of numbers determined by a seed to approximate the properties of random numbers [56], or through a round-robin fashion in the sequential selection scheme as in [16].

Remark 3.1: We point out that the randomized projection vector could be generated at each node synchronously provided that each node uses the same *seed* for the pseudo-random generator mechanism [56]. Such seed exchanges and the synchronisation can be done periodically by using pilot signals without a serious increase in the communication load [57]. In Section X, we examine the sensitivity of the proposed strategies against the asynchronous events, e.g., complete loss of diffused information, in several scenarios through numerical examples.

Most of the conventional adaptive filtering algorithms can be derived using the following generic update:

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} \left\{ D(\boldsymbol{w}, \boldsymbol{w}_t) + \mu L(d_t, \boldsymbol{u}_t, \boldsymbol{w}) \right\}, \tag{2.4}$$

where $D(\boldsymbol{w}, \boldsymbol{w}_t)$ is the divergence, distance or *a priori* knowledge, e.g., the Euclidean distance $\|\boldsymbol{w} - \boldsymbol{w}_t\|^2$, and $L(d_t, \boldsymbol{u}_t, \boldsymbol{w})$ is the loss function, e.g., the mean square error $E[(d_t - \boldsymbol{u}_t^T \boldsymbol{w})]$ [58, 59]. Correspondingly, the diffusion based distributed estimation algorithms can also be generated through the update (2.4).

However, note that the compressive diffusion scheme possesses different side information about the parameter of interest $\boldsymbol{w_o}$ from the full diffusion configuration, i.e., the constructed estimates instead of the original parameters. Although the constructed estimates $\boldsymbol{a}_{j,t}$'s track the original parameter estimation vectors, they are also parameter estimates of $\boldsymbol{w_o}$ as the original parameters. Particularly, in the proposed schemes, each node i has access to the *a priori* knowledge about the true parameter vector $\boldsymbol{w_o}$ as $\boldsymbol{w}_{i,t}$ and $\boldsymbol{a}_{j,t}$'s for $j \in \mathcal{N}_i \setminus i$. Hence, in the compressive diffusion implementation, we update according to

$$\begin{aligned}\boldsymbol{w}_{i,t+1} = \arg\min_{\boldsymbol{w}_i} \Bigg\{ & \gamma_{ii}\|\boldsymbol{w}_i - \boldsymbol{w}_{i,t}\|^2 + \sum_{j\in\mathcal{N}_i\setminus i} \gamma_{ij}\|\boldsymbol{w}_i - \boldsymbol{a}_{j,t}\|^2 \\ & + \mu_i \left(d_{i,t} - \boldsymbol{u}_{i,t}^T\boldsymbol{w}_i\right)^2 \Bigg\} \end{aligned} \tag{2.5}$$

such that in the update we also consider the Euclidean distance with the local parameter estimation $\boldsymbol{w}_{i,t}$ and the constructed estimates $\boldsymbol{a}_{j,t}$ of the neighboring nodes. In order to simplify the optimization in (2.5) and to obtain an LMS update exactly, we can replace the loss term $(d_{i,t} - \boldsymbol{u}_{i,t}^T\boldsymbol{w}_i)^2$ with the first order Taylor series expansion around $\boldsymbol{a}_{j,t}$, i.e.,

$$\begin{aligned}(d_{i,t} - \boldsymbol{u}_{i,t}^T\boldsymbol{w}_i)^2 = & \bar{e}_{i,t}(\boldsymbol{a}_{j,t})^2 - 2\bar{e}_{i,t}(\boldsymbol{a}_{j,t})\boldsymbol{u}_{i,t}^T(\boldsymbol{w}_i - \boldsymbol{a}_{j,t}) \\ & + O(\|\boldsymbol{w}_i\|^2), \end{aligned} \tag{2.6}$$

where we denote $\bar{e}_{i,t}(\boldsymbol{a}_{j,t}) \stackrel{\triangle}{=} d_{i,t} - \boldsymbol{u}_{i,t}^T\boldsymbol{a}_{j,t}$. Similarly, the first order Taylor series expansion around $\boldsymbol{w}_{i,t}$ leads

$$(d_{i,t} - \boldsymbol{u}_{i,t}^T\boldsymbol{w}_i)^2 = e_{i,t}^2 - 2e_{i,t}\boldsymbol{u}_{i,t}^T(\boldsymbol{w}_i - \boldsymbol{w}_{i,t}) + O(\|\boldsymbol{w}_i\|^2), \tag{2.7}$$

where $e_{i,t} \stackrel{\triangle}{=} d_{i,t} - \boldsymbol{u}_{i,t}^T\boldsymbol{w}_{i,t}$. Since $\sum_{j\in\mathcal{N}_i}\gamma_{ij} = 1$, the approximations (2.6) and (2.7) in (2.5) yields

$$\begin{aligned}\boldsymbol{w}_{i,t+1} = \arg\min_{\boldsymbol{w}_i} \Bigg\{ & \gamma_{ii}\|\boldsymbol{w}_i - \boldsymbol{w}_{i,t}\|^2 + \sum_{j\in\mathcal{N}_i\setminus i} \gamma_{ij}\|\boldsymbol{w}_i - \boldsymbol{a}_{j,t}\|^2 \\ & + \mu_i\gamma_{ii}\left[e_{i,t}^2 - 2e_{i,t}\boldsymbol{u}_{i,t}^T(\boldsymbol{w}_i - \boldsymbol{w}_{i,t})\right] \\ & + \mu_i \sum_{j\in\mathcal{N}_i\setminus i} \gamma_{ij}\left[\bar{e}_{i,t}(\boldsymbol{a}_{j,t})^2 - 2\bar{e}_{i,t}(\boldsymbol{a}_{j,t})\boldsymbol{u}_{i,t}^T(\boldsymbol{w}_i - \boldsymbol{a}_{j,t})\right] \Bigg\}. \end{aligned} \tag{2.8}$$

The minimized term in (2.8) is a convex function of $\boldsymbol{w}_i$ and the Hessian matrix $2\boldsymbol{I}_M \succ \boldsymbol{0}$ is positive definite. Hence, taking derivative and equating zero, we get the following update

$$\boldsymbol{w}_{i,t+1} = \boldsymbol{\varphi}_{i,t+1} + \mu_i \boldsymbol{u}_{i,t}(d_{i,t} - \boldsymbol{u}_{i,t}^T \boldsymbol{\varphi}_{i,t+1}), \tag{2.9}$$

where

$$\boldsymbol{\varphi}_{i,t+1} = \gamma_{ii}\boldsymbol{w}_{i,t} + \sum_{j \in \mathcal{N}_i \backslash i} \gamma_{ij} \boldsymbol{a}_{j,t}, \tag{2.10}$$

which is similar to the distributed LMS algorithm (2.3). Note that if we interchange $\boldsymbol{\varphi}_{i,t}$ and $\boldsymbol{w}_{i,t}$, in other words, when we assign the outcome of the combination as the final estimate rather than the outcome of the adaptation, we have the following algorithm:

$$\boldsymbol{\varphi}_{i,t+1} = \boldsymbol{w}_{i,t} + \mu_i \boldsymbol{u}_{i,t}(d_{i,t} - \boldsymbol{u}_{i,t}^T \boldsymbol{w}_{i,t}), \tag{2.11}$$

$$\boldsymbol{w}_{i,t+1} = \gamma_{ii}\boldsymbol{\varphi}_{i,t+1} + \sum_{j \in \mathcal{N}_i \backslash i} \gamma_{ij} \boldsymbol{a}_{j,t+1}. \tag{2.12}$$

We point out that (2.9) and (2.10) are the CTA diffusion strategy while (2.11) and (2.12) are the ATC diffusion strategy. Fig. 2.2 and 2.3 summarize the compressive diffusion strategy for the CTA and ATC strategies where $j_k \in \mathcal{N}_i$. We next introduce different approaches to generate the diffused information (which are used to construct $\boldsymbol{a}_{j,t+1}$'s).

In the compressive diffusion approach, instead of the full vector and irrespective of the final estimate, we always diffuse the linear transformation of the outcome of the adaptation, e.g., we diffuse $z_{i,t} = \boldsymbol{c}_t^T \boldsymbol{w}_{i,t}$ in the CTA strategy and $z_{i,t} = \boldsymbol{c}_t^T \boldsymbol{\varphi}_{i,t}$ in the ATC strategy. At each node, with the diffused information $z_{i,t}$, we update the constructed estimate $\boldsymbol{a}_{i,t}$ according to

$$\boldsymbol{a}_{i,t+1} = \arg\min_{\boldsymbol{a}_i} \left\{ \|\boldsymbol{a}_i - \boldsymbol{a}_{i,t}\|^2 + \eta_i \|z_{i,t} - \boldsymbol{c}_t^T \boldsymbol{a}_i\|^2 \right\},$$

where we choose the diffused data as the desired signal and try to minimize the mean-square of the difference between the estimate $\hat{z}_{i,t} = \boldsymbol{c}_t^T \boldsymbol{a}_i$ and $z_{i,t}$. Here, $\boldsymbol{a}_{i,t}$'s are the estimates of the $\boldsymbol{w}_{i,t}$'s or $\boldsymbol{\varphi}_{i,t}$'s in the CTA and the ATC strategies, respectively. The first order Taylor series approximation of the loss term $\|z_{i,t} -$

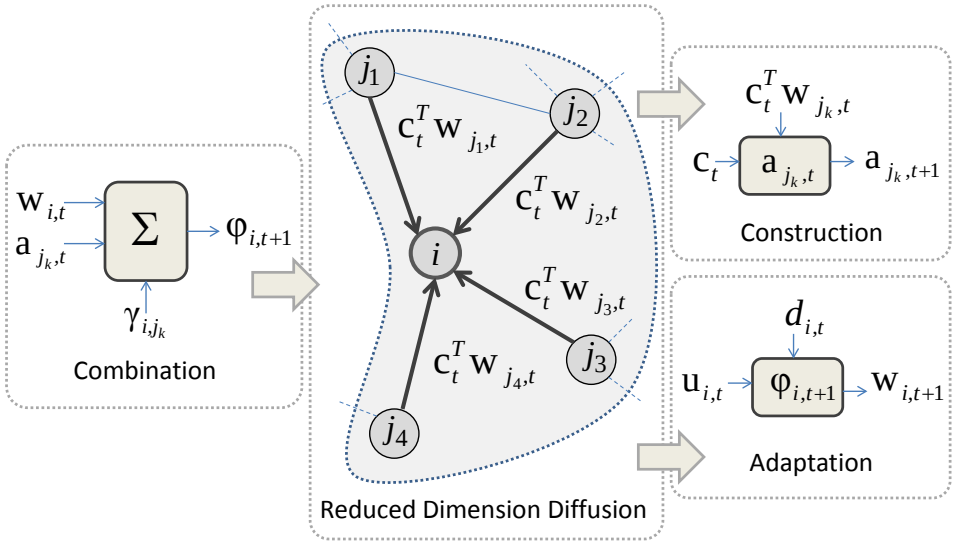


Figure 2.2: CTA strategy in the scalar diffusion framework.

$\hat{z}_{i,t}\|^2$ around $\boldsymbol{a}_{i,t}$ yields the following update

$$\boldsymbol{a}_{i,t+1} = \boldsymbol{a}_{i,t} + \eta_i \boldsymbol{c}_t \left(z_{i,t} - \boldsymbol{c}_t^T \boldsymbol{a}_{i,t}\right) \tag{2.13}$$

where $\eta_i > 0$ is the construction step size. We note that in [12] the reduced dimension diffusion approach constructs $\boldsymbol{a}_{i,t+1}$'s through the minimum disturbance principle and resulted update involves $\left[\boldsymbol{c}_t^T \boldsymbol{c}_t\right]^{-1}$ as the normalization term. The constructed estimates $\boldsymbol{a}_{i,t+1}$'s are combined with the outcome of the local adaptation algorithm through (2.10) or (2.12).

We next introduce methods where the information exchange is only a single bit [12]. When we construct $\boldsymbol{a}_{i,t}$ at node i, assuming $\boldsymbol{a}_{i,t}$'s are initialized with the same value, node $j \in \mathcal{N}_i$ has knowledge of the constructed estimate $\boldsymbol{a}_{i,t}$. Hence, we can perform the construction update at each neighboring node via the diffusion of the estimation error, i.e., $\epsilon_{i,t} \stackrel{\triangle}{=} z_{i,t} - \hat{z}_{i,t}$. Note that this does not influence the communication load, however, through the access to the exchange estimate $\boldsymbol{a}_{i,t+1}$ we can further reduce the communication load. Using the well-known sign algorithm [5], we can construct $\boldsymbol{a}_{i,t+1}$ as

$$\boldsymbol{a}_{i,t+1} = \boldsymbol{a}_{i,t} + \eta_i \boldsymbol{c}_t \text{sign}(\epsilon_{i,t}). \tag{2.14}$$

Hence, we can repeat (2.14) at each neighboring node via the diffusion of $z_{i,t} =$

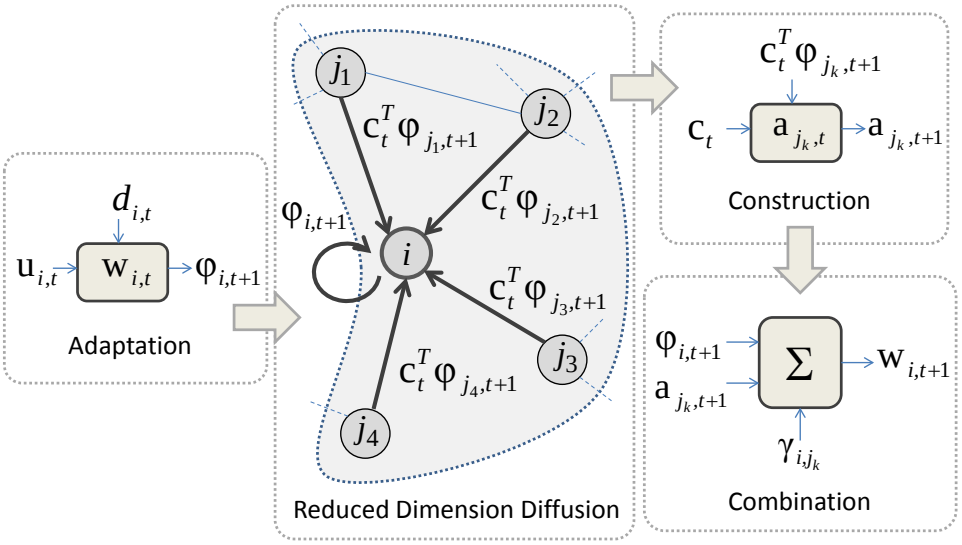


Figure 2.3: ATC strategy in the scalar diffusion framework.

sign($\epsilon_{i,t}$) only and then we combine with the local estimate by using (2.10) or (2.12).

In Tables 2.1 and 2.2, we tabulate the description of the proposed algorithms. We note that as seen in the Tables 2.1 and 2.2, the construction of $\boldsymbol{a}_{j,t}$ requires additional updates at each neighboring nodes. However, in the following, we propose an approach significantly reducing this computational load provided that all nodes use the same projection operator. We note that (2.9) and (2.11) require the linear combination of the constructed estimates. To this end, we define

$$\boldsymbol{\omega}_{i,t} \stackrel{\triangle}{=} \sum_{j \in \mathcal{N}_i \backslash i} \gamma_{ij} \boldsymbol{a}_{j,t},$$

so that for the same step size, i.e., $\eta = \eta_1 = \cdots = \eta_N$, the following relations

$$\begin{aligned} \boldsymbol{a}_{1,t+1} &= \boldsymbol{a}_{1,t} + \eta_1 \boldsymbol{c}_t \left(z_{1,t} - \boldsymbol{c}_t^T \boldsymbol{a}_{1,t} \right), \\ &\vdots \\ \boldsymbol{a}_{N,t+1} &= \boldsymbol{a}_{N,t} + \eta_N \boldsymbol{c}_t \left(z_{N,t} - \boldsymbol{c}_t^T \boldsymbol{a}_{N,t} \right) \end{aligned}$$

can be rewritten in a single update as

$$\boldsymbol{\omega}_{i,t+1} = \boldsymbol{\omega}_{i,t} + \eta \, \boldsymbol{c}_t \left(\sum_{j \in \mathcal{N}_i \backslash i} \gamma_{i,j} z_{j,t} - \boldsymbol{c}_t^T \boldsymbol{\omega}_{i,t} \right). \tag{2.15}$$

Table 2.1: The description of the scalar diffusion scheme with the ATC strategy.

Algorithm 2.1: Scalar Diffusion Strategies - ATC

Initialization:

For $i = 1$ to N do

$\boldsymbol{u}_{i,0} = \boldsymbol{c}_0 = \boldsymbol{w}_{i,0} = \boldsymbol{a}_{i,0} = [0, \cdots, 0]^T$

End for

Do for $t \geq 0$

For $i = 1$ to N do

Adaptation:

$e_{i,t} = d_{i,t} - \boldsymbol{u}_{i,t}^T \boldsymbol{w}_{i,t}$

$\boldsymbol{\varphi}_{i,t+1} = \boldsymbol{w}_{i,t} + \mu_i \boldsymbol{u}_{i,t} e_{i,t}$

Diffuse $z_{i,t} = \boldsymbol{c}_t^T \boldsymbol{\varphi}_{i,t}$ to neighboring nodes

Construction:

For all $j \in \mathcal{N}_i \setminus i$ do

$\epsilon_{j,t} = z_{j,t} - \boldsymbol{c}_t^T \boldsymbol{a}_{j,t}$

$\boldsymbol{a}_{j,t+1} = \boldsymbol{a}_{j,t} + \eta_j \boldsymbol{c}_t \epsilon_{j,t}$

End for

Combination:

$\boldsymbol{w}_{i,t+1} = \gamma_{i,i} \boldsymbol{\varphi}_{i,t+1} + \sum_{j \in \mathcal{N}_i \setminus i} \gamma_{i,j} \boldsymbol{a}_{j,t+1}$

End for

In that sense, as an example, instead of (2.12), we can construct the final parameter estimate $\boldsymbol{w}_{i,t+1}$ through

$$\boldsymbol{w}_{i,t+1} = \gamma_{i,i} \boldsymbol{\varphi}_{i,t+1} + \boldsymbol{\omega}_{i,t+1}, \tag{2.16}$$

thanks to the linear error function in the LMS update. Hence, we can significantly reduce the computational load, i.e., to only an additional LMS update, in the scalar diffusion strategies through (2.15) and (2.16). On the other hand, if the sign algorithm is used at each node in the construction of $\boldsymbol{a}_{j,t}$, each node should construct $\boldsymbol{a}_{j,t}$'s separately since the sign algorithm has a nonlinear error update, i.e., $\text{sign}(\epsilon_{j,t})$. However, the sign algorithm is known for its low complexity implementation and can be implemented through shift-registers provided that the

Table 2.2: The description of the single-bit diffusion scheme with the ATC strategy.

Algorithm 2.2: Single-bit Diffusion Strategies - ATC

Initialization:

For $i = 1$ to N do

$\boldsymbol{u}_{i,0} = \boldsymbol{c}_0 = \boldsymbol{w}_{i,0} = \boldsymbol{a}_{i,0} = [0, \cdots, 0]^T$

End for

Do for $t \geq 0$

For $i = 1$ to N do

Adaptation:

$e_{i,t} = d_{i,t} - \boldsymbol{u}_{i,t}^T \boldsymbol{w}_{i,t}$

$\epsilon_{i,t} = \boldsymbol{c}_t^T (\boldsymbol{\varphi}_{i,t} - \boldsymbol{a}_{i,t})$

$\boldsymbol{a}_{i,t+1} = \boldsymbol{a}_{i,t} + \eta_i \boldsymbol{c}_t \text{sign}(\epsilon_{i,t})$

$\boldsymbol{\varphi}_{i,t+1} = \boldsymbol{w}_{i,t} + \mu_i \boldsymbol{u}_{i,t} e_{i,t}$

Diffuse $z_{i,t} = \text{sign}(\epsilon_{i,t})$ to neighboring nodes

Construction:

For all $j \in \mathcal{N}_i \setminus i$ do

$\boldsymbol{a}_{j,t+1} = \boldsymbol{a}_{j,t} + \eta_j \boldsymbol{c}_t z_{j,t}$

End for

Combination:

$\boldsymbol{w}_{i,t+1} = \gamma_{i,i} \boldsymbol{\varphi}_{i,t+1} + \sum_{j \in \mathcal{N}_i \setminus i} \gamma_{i,j} \boldsymbol{a}_{j,t+1}$

End for

step-size is chosen as a power of 2 [5]. In this sense, the single-bit diffusion strategy significantly reduces the communication load, i.e., from continuum to a single bit, with a relatively small computational complexity increase. We point out that the single-bit diffusion also overcomes the bandwidth related issues especially in the wireless networks due to the significant reduction in the communication load and the inherently quantized diffusion data.

In the sequel, we introduce a global model gathering all network operations into a single update.

2.3 Global Model

We can write the scalar (2.13) and single bit (2.14) diffusion approaches for the ATC diffusion strategy in a compact form as

$$\boldsymbol{\varphi}_{i,t+1} = \boldsymbol{w}_{i,t} + \mu_i \boldsymbol{u}_{i,t} e_{i,t}, \tag{2.17}$$

$$\boldsymbol{a}_{j,t+1} = \boldsymbol{a}_{j,t} + \eta_j \boldsymbol{c}_t h(\epsilon_{j,t}), \tag{2.18}$$

$$\boldsymbol{w}_{i,t+1} = \gamma_{i,i}\boldsymbol{\varphi}_{i,t+1} + \sum_{j \in \mathcal{N}_i \backslash i} \gamma_{i,j} \boldsymbol{a}_{j,t+1},$$

where $e_{i,t} \triangleq d_{i,t} - \boldsymbol{u}_{i,t}^T \boldsymbol{w}_{i,t}$ and $\epsilon_{j,t} \triangleq \boldsymbol{c}_t^T \left(\boldsymbol{\varphi}_{j,t} - \boldsymbol{a}_{j,t}\right)$. For scalar and single bit diffusion approaches, $h(\epsilon_{j,t}) = \epsilon_{j,t}$ and $h(\epsilon_{j,t}) = \text{sign}(\epsilon_{j,t})$, respectively.

For the state-space representation that collects all network operations into a single update, we define $\boldsymbol{\varphi}_t = \text{col}\{\boldsymbol{\varphi}_{1,t}, \ldots, \boldsymbol{\varphi}_{N,t}\}$, $\boldsymbol{a}_t = \text{col}\{\boldsymbol{a}_{1,t}, \ldots, \boldsymbol{a}_{N,t}\}$, $\boldsymbol{w}_t = \text{col}\{\boldsymbol{w}_{1,t}, \ldots, \boldsymbol{w}_{N,t}\}$, $\underline{\boldsymbol{w}}_{\boldsymbol{o}} = \text{col}\{\boldsymbol{w}_{\boldsymbol{o}}, \ldots, \boldsymbol{w}_{\boldsymbol{o}}\}$ with $MN \times 1$ dimensions and $\boldsymbol{e}_t = \text{col}\{e_{1,t}, \ldots, e_{N,t}\}$, $\boldsymbol{\epsilon}_t = \text{col}\{\epsilon_{1,t}, \ldots, \epsilon_{N,t}\}$, $\boldsymbol{d}_t = \text{col}\{d_{1,t}, \ldots, d_{N,t}\}$, $\boldsymbol{v}_t = \text{col}\{v_{1,t}, \ldots, v_{N,t}\}$ with $N \times 1$ dimensions. For a given combination matrix $\boldsymbol{\Gamma} = [\gamma_{i,j}]$, we denote $\boldsymbol{G} \triangleq \boldsymbol{\Gamma} \otimes \boldsymbol{I}_M$. Additionally, the regression and projection vectors yields the following $MN \times N$ global matrices

$$\boldsymbol{U}_t \triangleq \begin{bmatrix} \boldsymbol{u}_{1,t} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{u}_{N,t} \end{bmatrix}, \; \boldsymbol{C}_t \triangleq \begin{bmatrix} \boldsymbol{c}_{1,t} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{c}_{N,t} \end{bmatrix}.$$

Indeed, we can model the network with compressive diffusion strategy as a larger network in which each node i has an imaginary counterpart which diffuses $\boldsymbol{a}_{i,t}$ to the neighbors of i, which is similar to the full diffusion configuration. The real nodes only get information from the imaginary nodes and do not diffuse any information. In that case, the network can be modelled as a directed graph with asymmetric inner node links and the combination matrix is given by

$$\tilde{\boldsymbol{\Gamma}} = \begin{bmatrix} \boldsymbol{\Gamma_D} & \boldsymbol{\Gamma_C} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix},$$

where $\boldsymbol{\Gamma_D} = \text{diag}\{\boldsymbol{\Gamma}\}$ and $\boldsymbol{\Gamma_C} = \boldsymbol{\Gamma} - \boldsymbol{\Gamma_D}$. Then, we can write $\boldsymbol{w}_t$ in terms of $\boldsymbol{\varphi}_t$ and $\boldsymbol{a}_t$ as

$$\boldsymbol{w}_t = \boldsymbol{G_D}\boldsymbol{\varphi}_t + \boldsymbol{G_C}\boldsymbol{a}_t, \tag{2.19}$$

where $\boldsymbol{G_D} \stackrel{\triangle}{=} \boldsymbol{\Gamma_D} \otimes \boldsymbol{I}_M$ and $\boldsymbol{G_C} \stackrel{\triangle}{=} \boldsymbol{\Gamma_C} \otimes \boldsymbol{I}_M$. The state-space representation is given by

$$\begin{aligned} \boldsymbol{\varphi}_{t+1} &= \boldsymbol{w}_t + \boldsymbol{M}\boldsymbol{U}_t\boldsymbol{e}_t, \\ \boldsymbol{a}_{t+1} &= \boldsymbol{a}_t + \boldsymbol{N}\boldsymbol{C}_t\boldsymbol{h}(\boldsymbol{\epsilon}_t), \\ \boldsymbol{w}_{t+1} &= \boldsymbol{G_D}\boldsymbol{\varphi}_{t+1} + \boldsymbol{G_C}\boldsymbol{a}_{t+1}, \end{aligned} \tag{2.20}$$

where $\boldsymbol{h}(\boldsymbol{\epsilon}_t) = \operatorname{col}\{h(\epsilon_{1,t}), h(\epsilon_{2,t}), \cdots, h(\epsilon_{N,t})\}$, $\boldsymbol{M} \stackrel{\triangle}{=} \operatorname{diag}\{[\mu_1, \ldots, \mu_N]\} \otimes \boldsymbol{I}_M$, and $\boldsymbol{N} \stackrel{\triangle}{=} \operatorname{diag}\{[\eta_1, \ldots, \eta_N]\} \otimes \boldsymbol{I}_M$. We obtain the global deviation vectors as

$$\tilde{\boldsymbol{\varphi}}_t \stackrel{\triangle}{=} \underline{\boldsymbol{w}}_{\boldsymbol{o}} - \boldsymbol{\varphi}_t \text{ and } \tilde{\boldsymbol{a}}_t \stackrel{\triangle}{=} \underline{\boldsymbol{w}}_{\boldsymbol{o}} - \boldsymbol{a}_t. \tag{2.21}$$

Since $\boldsymbol{\Gamma}\underline{\mathbf{1}} = \underline{\mathbf{1}}$,

$$\boldsymbol{G}\underline{\boldsymbol{w}}_{\boldsymbol{o}} = \underline{\boldsymbol{w}}_{\boldsymbol{o}} \tag{2.22}$$

then the global deviation update yields

$$\tilde{\boldsymbol{\varphi}}_{t+1} = \boldsymbol{G_D}\tilde{\boldsymbol{\varphi}}_t + \boldsymbol{G_C}\tilde{\boldsymbol{a}}_t - \boldsymbol{M}\boldsymbol{U}_t\boldsymbol{e}_t, \tag{2.23}$$

$$\tilde{\boldsymbol{a}}_{t+1} = \tilde{\boldsymbol{a}}_t - \boldsymbol{N}\boldsymbol{C}_t\boldsymbol{h}(\boldsymbol{\epsilon}_t). \tag{2.24}$$

We represent the global deviation updates (2.23) and (2.24) in a single equation as

$$\overbrace{\begin{bmatrix} \tilde{\boldsymbol{\varphi}}_{t+1} \\ \tilde{\boldsymbol{a}}_{t+1} \end{bmatrix}}^{\tilde{\boldsymbol{\psi}}_{t+1}} = \overbrace{\begin{bmatrix} \boldsymbol{G_D} & \boldsymbol{G_C} \\ \boldsymbol{0} & \boldsymbol{I}_{MN} \end{bmatrix}}^{\boldsymbol{X}} \overbrace{\begin{bmatrix} \tilde{\boldsymbol{\varphi}}_t \\ \tilde{\boldsymbol{a}}_t \end{bmatrix}}^{\tilde{\boldsymbol{\psi}}_t} - \overbrace{\begin{bmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N} \end{bmatrix}}^{\boldsymbol{D}} \overbrace{\begin{bmatrix} \boldsymbol{U}_t & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_t \end{bmatrix}}^{\boldsymbol{Y}_t} \overbrace{\begin{bmatrix} \boldsymbol{e}_t \\ \boldsymbol{h}(\boldsymbol{\epsilon}_t) \end{bmatrix}}^{\underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)} \tag{2.25}$$

or equivalently

$$\tilde{\boldsymbol{\psi}}_{t+1} = \boldsymbol{X}\tilde{\boldsymbol{\psi}}_t - \boldsymbol{D}\boldsymbol{Y}_t\underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t), \tag{2.26}$$

where $\tilde{\boldsymbol{\psi}}_t \stackrel{\triangle}{=} \operatorname{col}\{\tilde{\boldsymbol{\varphi}}_t, \tilde{\boldsymbol{a}}_t\}$. We next use the following assumptions in the analyses of the weighted-energy recursion of (2.26):

Assumption 1:

The regressor signal $\boldsymbol{u}_{i,t}$ is zero-mean independently and identically distributed (i.i.d.) Gaussian random vector process and spatially and temporally independent from the other regressor signals, the randomized projection operator and the observation noise. Each node uses spatially and

temporally independent projection vector, i.e., $\boldsymbol{c}_{i,t}$. The projection operator is zero-mean i.i.d. Gaussian random vector process and the observation noise $v_{i,t}$ is also a zero-mean i.i.d. Gaussian random variable. Note that such assumptions are commonly used in the analysis of traditional adaptive schemes [5, 60].

Assumption 2:

The *a priori* estimation error vector in the construction update (2.20), i.e., $\boldsymbol{\epsilon}_{a,t} \stackrel{\triangle}{=} \boldsymbol{C}_t^T(\tilde{\boldsymbol{a}}_t - \tilde{\boldsymbol{\varphi}}_t)$, has Gaussian distribution and it is jointly Gaussian with the weighted *a priori* estimation error vector, i.e., $\boldsymbol{C}_t^T\boldsymbol{\Sigma}(\tilde{\boldsymbol{a}}_t - \tilde{\boldsymbol{\varphi}}_t)$, for any constant matrix $\boldsymbol{\Sigma}$. This assumption is reasonable for long filters, i.e. M is large, sufficiently small step size η_i's and by the Assumption 1 [61]. We adopt the Assumption 2 in the analyses of the single-bit diffusion schemes due to the nonlinearity in the corresponding construction update.

We point out that the Assumptions 1 and 2 are impractical in general, however, widely used in the adaptive filtering literature to analyze the performance of the schemes analytically due to the mathematical tractability and the analytical results match closely with the ensemble averaged simulation results. In the next sections, we analyze the mean-square convergence performance of the proposed approaches separately.

2.4 Scalar Diffusion with Gaussian Regressors

For the one-dimension diffusion approach, (2.26) yields

$$\tilde{\boldsymbol{\psi}}_{t+1} = \boldsymbol{X}\tilde{\boldsymbol{\psi}}_t - \boldsymbol{D}\boldsymbol{Y}_t\underline{\boldsymbol{e}}_t, \tag{2.27}$$

where $\underline{\boldsymbol{e}}_t \stackrel{\triangle}{=} \text{col}\{\boldsymbol{e}_t, \boldsymbol{\epsilon}_t\}$. By (2.19), (2.21) and (2.22), we note that $\boldsymbol{e}_t$ is given by

$$\boldsymbol{e}_t = \boldsymbol{U}_t^T(\boldsymbol{G}_{\boldsymbol{D}}\tilde{\boldsymbol{\varphi}}_t + \boldsymbol{G}_{\boldsymbol{C}}\tilde{\boldsymbol{a}}_t) + \boldsymbol{v}_t. \tag{2.28}$$

Similarly, we have

$$\boldsymbol{\epsilon}_t = \boldsymbol{C}_t^T(-\tilde{\boldsymbol{\varphi}}_t + \tilde{\boldsymbol{a}}_t). \tag{2.29}$$

Hence, through (2.28) and (2.29), we obtain the global estimation error $\underline{\boldsymbol{e}}_t$ as

$$\begin{aligned}\underline{\boldsymbol{e}}_t &= \begin{bmatrix} \boldsymbol{U}_t & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_t \end{bmatrix}^T \underbrace{\begin{bmatrix} \boldsymbol{G_D} & \boldsymbol{G_C} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix}}_{\boldsymbol{Z}} \begin{bmatrix} \tilde{\boldsymbol{\varphi}}_t \\ \tilde{\boldsymbol{a}}_t \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{v}_t \\ \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{n}_t} \\ &= \boldsymbol{Y}_t^T \boldsymbol{Z} \tilde{\boldsymbol{\psi}}_t + \boldsymbol{n}_t. \end{aligned} \tag{2.30}$$

Through (2.30), we rewrite (2.27) as

$$\begin{aligned}\tilde{\boldsymbol{\psi}}_{t+1} &= \boldsymbol{X}\tilde{\boldsymbol{\psi}}_t - \boldsymbol{D}\boldsymbol{Y}_t(\boldsymbol{Y}_t^T \boldsymbol{Z}\tilde{\boldsymbol{\psi}}_t + \boldsymbol{n}_t) \\ &= (\boldsymbol{X} - \boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{Z})\tilde{\boldsymbol{\psi}}_t - \boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{n}_t. \end{aligned} \tag{2.31}$$

We utilize the weighted-energy relation relating the energy of the error and deviation quantities in the performance analyzes through a weighting matrix $\boldsymbol{\Sigma}$. Then, we obtain

$$\begin{aligned}\tilde{\boldsymbol{\psi}}_{t+1}^T\boldsymbol{\Sigma}\tilde{\boldsymbol{\psi}}_{t+1} = \tilde{\boldsymbol{\psi}}_t^T(\boldsymbol{X} - \boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{Z})^T\boldsymbol{\Sigma}(\boldsymbol{X} - \boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{Z})\tilde{\boldsymbol{\psi}}_t \\ - 2\boldsymbol{n}_t^T\boldsymbol{Y}_t^T\boldsymbol{D}\boldsymbol{\Sigma}(\boldsymbol{X} - \boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{Z})\tilde{\boldsymbol{\psi}}_t \\ + \boldsymbol{n}_t^T\boldsymbol{Y}_t^T\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{n}_t.\end{aligned}$$

By the Assumption 1, the observation noise $\boldsymbol{v}_t$ is independent from the network statistics and the weighted energy relation for (2.31) is given by

$$E\|\tilde{\boldsymbol{\psi}}_{t+1}\|_{\boldsymbol{\Sigma}}^2 = E\|\tilde{\boldsymbol{\psi}}_t\|_{\boldsymbol{\Sigma}'}^2 + E[\boldsymbol{n}_t^T\boldsymbol{Y}_t^T\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{n}_t], \tag{2.32}$$

where

$$\begin{aligned}\boldsymbol{\Sigma}' \triangleq &\boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{X} - \boldsymbol{Z}^T\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{X} - \boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{Z} \\ &+ \boldsymbol{Z}^T\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{Z}.\end{aligned}$$

Apart form the weighting matrix $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}'$ is random due to the data dependence. By the Assumption 1, $\boldsymbol{Y}_t$ is independent of $\tilde{\boldsymbol{\psi}}_t$ and we can replace $\boldsymbol{\Sigma}'$ by its mean value, i.e., $\boldsymbol{\Sigma}' = E[\boldsymbol{\Sigma}']$ [5, 6]. Hence, the weighting matrix is given by

$$\begin{aligned}\boldsymbol{\Sigma}' = &\boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{X} - \boldsymbol{Z}^T E\left[\boldsymbol{Y}_t\boldsymbol{Y}_t^T\right]\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{X} - \boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{D}E\left[\boldsymbol{Y}_t\boldsymbol{Y}_t^T\right]\boldsymbol{Z} \\ &+ \boldsymbol{Z}^T\boldsymbol{D}E\left[\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{\Sigma}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\right]\boldsymbol{D}\boldsymbol{Z}.\end{aligned} \tag{2.33}$$

Note that in the last term of the right hand side (RHS) of (2.33), we take $\boldsymbol{D}$'s out of the expectation thanks to the block diagonal structure of $\boldsymbol{D}$ and $\boldsymbol{Y}_t\boldsymbol{Y}_t^T$.

In order to calculate certain data moments in (2.32) and (2.33), by the Assumption 1, we obtain

$$\begin{aligned}\boldsymbol{\Lambda_u} &\triangleq E[\boldsymbol{U}_t\boldsymbol{U}_t^T] = \text{diag}\{[\sigma_{u,1}^2, \sigma_{u,2}^2, \ldots, \sigma_{u,N}^2]\} \otimes \boldsymbol{I}_M \\ \boldsymbol{\Lambda_c} &\triangleq E[\boldsymbol{C}_t\boldsymbol{C}_t^T] = \text{diag}\{[\sigma_{c,1}^2, \sigma_{c,2}^2, \ldots, \sigma_{c,N}^2]\} \otimes \boldsymbol{I}_M.\end{aligned}$$

Then, we obtain

$$\boldsymbol{\Lambda} \triangleq E[\boldsymbol{Y}_t\boldsymbol{Y}_t^T] = \begin{bmatrix}\boldsymbol{\Lambda_u} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Lambda_c}\end{bmatrix}.$$

In the performance analysis, convenient vectorisation notation is used to exploit the diagonal structure of matrices [5,62]. In (2.32) and (2.33), matrices have block diagonal structures, thus we use the block vectorisation operator bvec$\{\cdot\}$ [6] such that given an $NM \times NM$ block matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \ldots & \boldsymbol{\Sigma}_{1N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{N1} & \ldots & \boldsymbol{\Sigma}_{NN}\end{bmatrix},$$

where each block $\boldsymbol{\Sigma}_{ij}$ is a $M \times M$ block, $\boldsymbol{\sigma}_{ij} = \text{vec}\{\boldsymbol{\Sigma}_{ij}\}$ with standard vec$\{\cdot\}$ operator and $\boldsymbol{\sigma}_j = \text{col}\{\boldsymbol{\sigma}_{1j}, \boldsymbol{\sigma}_{2j}, \ldots, \boldsymbol{\sigma}_{Nj}\}$, then

$$\text{bvec}\{\boldsymbol{\Sigma}\} = \boldsymbol{\sigma} = \text{col}\{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_N\}. \tag{2.34}$$

We also use the *block Kronecker product* of two block matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, denoted by $\boldsymbol{A} \odot \boldsymbol{B}$. The ij-block is given by

$$[\boldsymbol{A} \odot \boldsymbol{B}]_{ij} = \begin{bmatrix}\boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{11} & \ldots & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{1N} \\ \vdots & \ddots \vdots & \\ \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{N1} & \ldots & \boldsymbol{A}_{ij} \otimes \boldsymbol{B}_{NN}\end{bmatrix}. \tag{2.35}$$

The block vectorisation operator bvec$\{\cdot\}$ (2.34) and the block Kronecker product (2.35) are related by

$$\text{bvec}\{\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{B}\} = (\boldsymbol{B}^T \odot \boldsymbol{A})\boldsymbol{\sigma} \tag{2.36}$$

and

$$\mathrm{Tr}\{\boldsymbol{A}^T\boldsymbol{B}\} = (\mathrm{bvec}\{\boldsymbol{A}\})^T\mathrm{bvec}\{\boldsymbol{B}\}. \tag{2.37}$$

The term in the RHS of (2.32) yields

$$E\left[\boldsymbol{n}_t^T\boldsymbol{Y}_t^T\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{n}_t\right] = \mathrm{Tr}\left(\boldsymbol{\Lambda}\boldsymbol{D}^2E\left[\boldsymbol{n}_t\boldsymbol{n}_t^T\right]\boldsymbol{\Sigma}\right)$$

and let

$$E\left[\boldsymbol{n}_t\boldsymbol{n}_t^T\right] = \boldsymbol{R}_{\boldsymbol{n}} = \begin{bmatrix}\boldsymbol{R}_{\boldsymbol{v}} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix},$$

where $\boldsymbol{R}_{\boldsymbol{v}} \triangleq \mathrm{diag}\{\sigma_{v,1}^2,\ldots,\sigma_{v,N}^2\}\otimes\boldsymbol{I}_M$. Then by (2.37),

$$E\left[\boldsymbol{n}_t^T\boldsymbol{Y}_t^T\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{D}\boldsymbol{Y}_t\boldsymbol{n}_t\right] = \boldsymbol{b}^T\boldsymbol{\sigma},$$

where

$$\boldsymbol{b} \triangleq \mathrm{bvec}\{\boldsymbol{R}_{\boldsymbol{n}}\boldsymbol{D}^2\boldsymbol{\Lambda}\}. \tag{2.38}$$

The last term on the RHS of (2.33) yields $\boldsymbol{A} = E\left[\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{\Sigma}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\right]$, where the $M\times M$ block is given by

$$[\boldsymbol{A}]_{ij} = \begin{cases}\boldsymbol{\Lambda}_i(\boldsymbol{\Sigma}_{ii}+\boldsymbol{\Sigma}_{ii}^T)\boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_i\mathrm{Tr}\left(\boldsymbol{\Sigma}_{ii}\boldsymbol{\Lambda}_i\right) & i=j\\ \boldsymbol{\Lambda}_i\boldsymbol{\Sigma}_{ij}\boldsymbol{\Lambda}_j & i\neq j\end{cases}$$

by the Assumption 1 [5]. The matrix $\boldsymbol{\Lambda}$ could be denoted as $\boldsymbol{\Lambda} = \mathrm{diag}\{\boldsymbol{\Lambda}_1,\cdots,\boldsymbol{\Lambda}_{2N}\}$ where $\boldsymbol{\Lambda}_i$ for $i=\{1,2,\ldots,2N\}$ is $M\times M$ block matrix, e.g., $\boldsymbol{\Lambda}_1 = \sigma_{u,1}^2\boldsymbol{I}_M$ or $\boldsymbol{\Lambda}_{N+1} = \sigma_{c,1}^2\boldsymbol{I}_M$. The $M\times M$ ijth block of $\boldsymbol{\Sigma}$ is denoted by $\boldsymbol{\Sigma}_{ij}$.

Remark 5.1: We note that if each node used the same projection operator, $\boldsymbol{c}_{i,t}$'s would be spatially dependent. In that case, $[\boldsymbol{A}]_{ij}$ is defined as

$$[\boldsymbol{A}]_{ij} = \begin{cases}\boldsymbol{\Lambda}_i(\boldsymbol{\Sigma}_{ii}+\boldsymbol{\Sigma}_{ii}^T)\boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_i\mathrm{Tr}\left(\boldsymbol{\Sigma}_{ii}\boldsymbol{\Lambda}_i\right) & i=j,\\ \boldsymbol{\Lambda}_i(\boldsymbol{\Sigma}_{ij}+\boldsymbol{\Sigma}_{ij}^T)\boldsymbol{\Lambda}_j + \boldsymbol{\Lambda}_i\mathrm{Tr}\left(\boldsymbol{\Sigma}_{ij}\boldsymbol{\Lambda}_j\right) & i>N\wedge j>N,\\ \boldsymbol{\Lambda}_i\boldsymbol{\Sigma}_{ij}\boldsymbol{\Lambda}_j & \text{otherwise.}\end{cases}$$

Through (2.35), (2.37), we obtain $\mathrm{bvec}\{\boldsymbol{A}\} = \mathcal{A}\boldsymbol{\sigma}$ with $\mathcal{A} = \mathrm{diag}\{\mathcal{A}_1,\ldots,\mathcal{A}_{2N}\}$, $\mathcal{A}_j = \mathrm{diag}\{\mathcal{A}_{1j},\ldots,\mathcal{A}_{2Nj}\}$ and

$$\mathcal{A}_{ij} = \begin{cases}2\boldsymbol{\Lambda}_i\otimes\boldsymbol{\Lambda}_i + \boldsymbol{\lambda}_i\boldsymbol{\lambda}_i^T & i=j\\ \boldsymbol{\Lambda}_i\otimes\boldsymbol{\Lambda}_j & i\neq j\end{cases}$$

where $\boldsymbol{\lambda}_i = \text{vec}\{\boldsymbol{\Lambda}_i\}$.

Hence, the block vectorization of the weighting matrix $\boldsymbol{\Sigma}'$ (2.33) yields

$$\begin{aligned}\text{bvec}\{\boldsymbol{\Sigma}'\} = \big(&\boldsymbol{X}^T \odot \boldsymbol{X}^T - (\boldsymbol{X}^T \odot \boldsymbol{Z}^T)(\boldsymbol{I}_{2MN} \odot \boldsymbol{\Lambda}\boldsymbol{D}) \\ &- (\boldsymbol{Z}^T \odot \boldsymbol{X}^T)(\boldsymbol{\Lambda}\boldsymbol{D} \odot \boldsymbol{I}_{2MN}) \\ &+ (\boldsymbol{Z}^T \odot \boldsymbol{Z}^T)(\boldsymbol{D} \odot \boldsymbol{D})\mathcal{A}\big)\,\boldsymbol{\sigma}.\end{aligned}$$

For notational simplicity, we change the weighted-norm notation such that $\|\tilde{\boldsymbol{\varphi}}_t\|^2_{\boldsymbol{\sigma}}$ refers to $\|\tilde{\boldsymbol{\varphi}}_t\|^2_{\boldsymbol{\Sigma}}$ where $\boldsymbol{\sigma} = \text{bvec}\{\boldsymbol{\Sigma}\}$. As a result, we obtain the weighted-energy recursion as

$$E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\sigma}} = E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{F}\boldsymbol{\sigma}} + \boldsymbol{b}^T\boldsymbol{\sigma} \tag{2.39}$$

$$\begin{aligned}\boldsymbol{F} \triangleq\ &\boldsymbol{X}^T \odot \boldsymbol{X}^T + (\boldsymbol{Z}^T \odot \boldsymbol{Z}^T)(\boldsymbol{D} \odot \boldsymbol{D})\mathcal{A} \\ &- (\boldsymbol{X}^T \odot \boldsymbol{Z}^T)(\boldsymbol{I}_{2MN} \odot \boldsymbol{\Lambda}\boldsymbol{D}) \\ &- (\boldsymbol{Z}^T \odot \boldsymbol{X}^T)(\boldsymbol{\Lambda}\boldsymbol{D} \odot \boldsymbol{I}_{2MN}).\end{aligned} \tag{2.40}$$

Through (2.39) and (2.40), we can analyze the learning, convergence and stability behavior of the network. Iterating the weighted-energy recursion, we obtain

$$\begin{aligned}E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\sigma}} &= E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{F}\boldsymbol{\sigma}} + \boldsymbol{b}^T\boldsymbol{\sigma} \\ E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{F}\boldsymbol{\sigma}} &= E\|\tilde{\boldsymbol{\psi}}_{t-1}\|^2_{\boldsymbol{F}^2\boldsymbol{\sigma}} + \boldsymbol{b}^T\boldsymbol{F}\boldsymbol{\sigma} \\ &\vdots \\ E\|\tilde{\boldsymbol{\psi}}_1\|^2_{\boldsymbol{F}^t\boldsymbol{\sigma}} &= E\|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{F}^{t+1}\boldsymbol{\sigma}} + \boldsymbol{b}^T\boldsymbol{F}^t\boldsymbol{\sigma}.\end{aligned}$$

Assuming the parameter estimates $\boldsymbol{\varphi}_{i,t}$ and $\boldsymbol{a}_{i,t}$ are initialized with zeros, $E\|\tilde{\boldsymbol{\psi}}_0\|^2 = \|\underline{\underline{\boldsymbol{w}}}_{\boldsymbol{o}}\|^2$ where $\underline{\underline{\boldsymbol{w}}}_{\boldsymbol{o}} \triangleq \text{col}\{\underline{\boldsymbol{w}}_{\boldsymbol{o}}, \underline{\boldsymbol{w}}_{\boldsymbol{o}}\}$. The iterations yield

$$E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\sigma}} = \|\underline{\underline{\boldsymbol{w}}}_{\boldsymbol{o}}\|^2_{\boldsymbol{F}^{t+1}\boldsymbol{\sigma}} + \boldsymbol{b}^T\left(\sum_{k=0}^{t}\boldsymbol{F}^k\right)\boldsymbol{\sigma}. \tag{2.41}$$

By (2.41), we reach the following final recursion:

$$E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\sigma}} = E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{\sigma}} + \boldsymbol{b}^T\boldsymbol{F}^t\boldsymbol{\sigma} - \|\underline{\underline{\boldsymbol{w}}}_{\boldsymbol{o}}\|^2_{\boldsymbol{F}^t(\boldsymbol{I}-\boldsymbol{F})\boldsymbol{\sigma}}. \tag{2.42}$$

Remark 5.2: We note that (2.42) is of essence since through the weighting matrix $\boldsymbol{\Sigma}$ we can extract information about the learning and convergence behavior

Table 2.3: Initial conditions and weighting matrices for different configurations.

Framework	$E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{\Sigma}}$	$E\|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\Sigma}}$	$\boldsymbol{\Sigma}$
CTA	$\frac{1}{N}E\|\tilde{\boldsymbol{\varphi}}_t\|^2$	$\frac{1}{N}\|\underline{\boldsymbol{w}}_{\boldsymbol{o}}\|^2$	$\frac{1}{N}\begin{bmatrix} \boldsymbol{I}_{MN} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$
ATC	$\frac{1}{N}E\|\tilde{\boldsymbol{w}}_t\|^2$	$\frac{1}{N}\|\underline{\boldsymbol{w}}_{\boldsymbol{o}}\|^2$	$\frac{1}{N}\begin{bmatrix} \boldsymbol{G_D}^T\boldsymbol{G_D} & \boldsymbol{G_D}^T\boldsymbol{G_C} \\ \boldsymbol{G_C}^T\boldsymbol{G_D} & \boldsymbol{G_C}^T\boldsymbol{G_C} \end{bmatrix}$

Framework	$E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{\Sigma}}$	$E\|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\Sigma}}$	$\boldsymbol{\Sigma}$
CTA	$\frac{1}{N}E\|\tilde{\boldsymbol{\varphi}}_t\|^2_{\boldsymbol{\Lambda_u}}$	$\frac{1}{N}\|\underline{\boldsymbol{w}}_{\boldsymbol{o}}\|^2_{\boldsymbol{\Lambda_u}}$	$\frac{1}{N}\begin{bmatrix} \boldsymbol{\Lambda_u} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$
ATC	$\frac{1}{N}E\|\tilde{\boldsymbol{w}}_t\|^2_{\boldsymbol{\Lambda_u}}$	$\frac{1}{N}\|\underline{\boldsymbol{w}}_{\boldsymbol{o}}\|^2_{\boldsymbol{\Lambda_u}}$	$\frac{1}{N}\begin{bmatrix} \boldsymbol{G_D}^T\boldsymbol{\Lambda_u}\boldsymbol{G_D} & \boldsymbol{G_D}^T\boldsymbol{\Lambda_u}\boldsymbol{G_C} \\ \boldsymbol{G_C}^T\boldsymbol{\Lambda_u}\boldsymbol{G_D} & \boldsymbol{G_C}^T\boldsymbol{\Lambda_u}\boldsymbol{G_C} \end{bmatrix}$

of the network. In Table 2.3, we tabulate the initial conditions (we assume the initial parameter vectors are set to $\boldsymbol{0}$) and the weighting matrices corresponding to various conventional performance measures.

Remark 5.3: In this paper, (2.42) provides a recursion for the weighted deviation parameter where we assign $\boldsymbol{\varphi}_{i,t}$ as the final estimate instead of $\boldsymbol{w}_{i,t}$, which implies the CTA strategy, however, the recursion also provides the performance of the ATC strategy with appropriate combination matrix $\boldsymbol{\Sigma}$ and the initial condition (See Table 2.3).

Next, we analyze the mean-square convergence performance of the single-bit diffusion approach for Gaussian regressors.

2.5 Single-Bit Diffusion with Gaussian Regressors

The weighted-energy relation of (2.26) yields

$$\begin{aligned} E\left[\tilde{\boldsymbol{\psi}}_{t+1}^T \boldsymbol{\Sigma} \tilde{\boldsymbol{\psi}}_{t+1}\right] &= E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X}^T \boldsymbol{\Sigma} \boldsymbol{X} \tilde{\boldsymbol{\psi}}_t\right] \\ &\quad - E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X}^T \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)\right] \\ &\quad - E\left[\underline{\boldsymbol{h}}^T(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t) \boldsymbol{Y}_t^T \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{X} \tilde{\boldsymbol{\psi}}_t\right] \\ &\quad + E\left[\underline{\boldsymbol{h}}^T(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t) \boldsymbol{Y}_t^T \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)\right]. \end{aligned} \tag{2.43}$$

We evaluate RHS of (2.43) term by term in order to find the variance relation. We first partition the weighting matrix as follows:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_2 \\ \boldsymbol{\Sigma}_3 & \boldsymbol{\Sigma}_4 \end{bmatrix}. \tag{2.44}$$

Through the partitioning (2.44), we obtain

$$\begin{aligned} E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X}^T \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)\right] &= E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \boldsymbol{\Sigma}_1 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{U}_t^T \boldsymbol{Z_u} \tilde{\boldsymbol{\psi}}_t\right] \\ &\quad + E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{C}_t \mathrm{sign}\left(\boldsymbol{C}_t^T \boldsymbol{Z_d} \tilde{\boldsymbol{\psi}}_t\right)\right] \\ &\quad + E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_d}^T \boldsymbol{\Sigma}_3 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{U}_t^T \boldsymbol{Z_u} \tilde{\boldsymbol{\psi}}_t\right] \\ &\quad + E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_d}^T \boldsymbol{\Sigma}_4 \boldsymbol{N} \boldsymbol{C}_t \mathrm{sign}\left(\boldsymbol{C}_t^T \boldsymbol{Z_d} \tilde{\boldsymbol{\psi}}_t\right)\right], \end{aligned} \tag{2.45}$$

where we partition $\boldsymbol{X}$ and $\boldsymbol{Z}$ such that $\boldsymbol{X} = \mathrm{col}\{\boldsymbol{X_u}, \boldsymbol{X_d}\}$ and $\boldsymbol{Z} = \mathrm{col}\{\boldsymbol{Z_u}, \boldsymbol{Z_d}\}$. We note that the second and fourth terms in the RHS of (2.45) include the nonlinear sign$(\cdot)$ function. It is not straight-forward to evaluate the expectations with this nonlinearity, thus we introduce the following lemma.

Lemma 1: *Under the Assumption 2, the Price's theorem [5] leads to*

$$\begin{aligned} &E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{C}_t \mathrm{sign}\left(\boldsymbol{C}_t^T \boldsymbol{Z_d} \tilde{\boldsymbol{\psi}}_t\right)\right] \\ &\qquad = E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{\Omega}_t \boldsymbol{C}_t \boldsymbol{C}_t^T \boldsymbol{Z_d} \tilde{\boldsymbol{\psi}}_t\right], \end{aligned} \tag{2.46}$$

$$\begin{aligned} &E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_d}^T \boldsymbol{\Sigma}_4 \boldsymbol{N} \boldsymbol{C}_t \mathrm{sign}\left(\boldsymbol{C}_t^T \boldsymbol{Z_d} \tilde{\boldsymbol{\psi}}_t\right)\right] \\ &\qquad = E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_d}^T \boldsymbol{\Sigma}_4 \boldsymbol{N} \boldsymbol{\Omega}_t \boldsymbol{C}_t \boldsymbol{C}_t^T \boldsymbol{Z_d} \tilde{\boldsymbol{\psi}}_t\right], \end{aligned} \tag{2.47}$$

where $\boldsymbol{\Omega}_t$ *is defined as*

$$\boldsymbol{\Omega}_t \triangleq \begin{bmatrix} \frac{E|\epsilon_{1,t}|}{E[\epsilon_{1,t}^2]}\boldsymbol{I}_M & \cdots & \boldsymbol{0}_M \\ \vdots & \ddots & \vdots \\ \boldsymbol{0}_M & \cdots & \frac{E|\epsilon_{N,t}|}{E[\epsilon_{N,t}^2]}\boldsymbol{I}_M \end{bmatrix}.$$

Proof: We first show the equality of (2.46) for the two-node case. Then the extension for a larger network is straight forward. We can rewrite the term on the left hand side (LHS) of (2.46) as

$$\begin{aligned} &E[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{C}_t \text{sign}(\boldsymbol{\epsilon}_t)] \\ &\qquad = E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \underbrace{\begin{bmatrix} \boldsymbol{\varsigma_1} & \boldsymbol{\varsigma_2} \\ \boldsymbol{\varsigma_3} & \boldsymbol{\varsigma_4} \end{bmatrix}}_{\boldsymbol{\Sigma}_2} \boldsymbol{N} \boldsymbol{C}_t \text{sign}(\boldsymbol{\epsilon}_t) \right]. \end{aligned} \tag{2.48}$$

After some algebra, (2.48) yields

$$\begin{aligned} &E[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{C}_t \text{sign}(\boldsymbol{\epsilon}_t)] \\ &\qquad = E[(\gamma_{11}\tilde{\boldsymbol{\varphi}}_{1,t}^T + \gamma_{12}\tilde{\boldsymbol{a}}_{2,t}^T)\boldsymbol{\varsigma_1}\eta_1 \boldsymbol{c}_{1,t}\text{sign}(\epsilon_{1,t})] \\ &\qquad\quad + E[(\gamma_{11}\tilde{\boldsymbol{\varphi}}_{1,t}^T + \gamma_{12}\tilde{\boldsymbol{a}}_{2,t}^T)\boldsymbol{\varsigma_2}\eta_2 \boldsymbol{c}_{2,t}\text{sign}(\epsilon_{2,t})] \\ &\qquad\quad + E[(\gamma_{22}\tilde{\boldsymbol{\varphi}}_{2,t}^T + \gamma_{21}\tilde{\boldsymbol{a}}_{1,t}^T)\boldsymbol{\varsigma_3}\eta_1 \boldsymbol{c}_{1,t}\text{sign}(\epsilon_{1,t})] \\ &\qquad\quad + E[(\gamma_{22}\tilde{\boldsymbol{\varphi}}_{2,t}^T + \gamma_{21}\tilde{\boldsymbol{a}}_{1,t}^T)\boldsymbol{\varsigma_4}\eta_2 \boldsymbol{c}_{2,t}\text{sign}(\epsilon_{2,t})]. \end{aligned} \tag{2.49}$$

In order to evaluate the expectations on the RHS of (2.49), by the Assumption 2 and the Price's result [63–65], we obtain

$$\begin{aligned} &E[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X_u}^T \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{C}_t \text{sign}(\boldsymbol{\epsilon}_t)] \\ &\qquad = E[(\gamma_{11}\tilde{\boldsymbol{\varphi}}_{1,t}^T + \gamma_{12}\tilde{\boldsymbol{a}}_{2,t}^T)\boldsymbol{\varsigma_1}\eta_1 \boldsymbol{c}_{1,t}\epsilon_{1,t}]\frac{E|\epsilon_{1,t}|}{E[\epsilon_{1,t}^2]} \\ &\qquad\quad + E[(\gamma_{11}\tilde{\boldsymbol{\varphi}}_{1,t}^T + \gamma_{12}\tilde{\boldsymbol{a}}_{2,t}^T)\boldsymbol{\varsigma_2}\eta_2 \boldsymbol{c}_{2,t}\epsilon_{2,t}]\frac{E|\epsilon_{2,t}|}{E[\epsilon_{2,t}^2]} \\ &\qquad\quad + E[(\gamma_{22}\tilde{\boldsymbol{\varphi}}_{2,t}^T + \gamma_{21}\tilde{\boldsymbol{a}}_{1,t}^T)\boldsymbol{\varsigma_3}\eta_1 \boldsymbol{c}_{1,t}\epsilon_{1,t}]\frac{E|\epsilon_{1,t}|}{E[\epsilon_{1,t}^2]} \\ &\qquad\quad + E[(\gamma_{22}\tilde{\boldsymbol{\varphi}}_{2,t}^T + \gamma_{21}\tilde{\boldsymbol{a}}_{1,t}^T)\boldsymbol{\varsigma_4}\eta_2 \boldsymbol{c}_{2,t}\epsilon_{2,t}]\frac{E|\epsilon_{2,t}|}{E[\epsilon_{2,t}^2]}. \end{aligned} \tag{2.50}$$

Rearranging (2.50) into a matrix product form leads (2.46). Following the same way, we can also get (2.47) and the proof is concluded. □

By (2.45), (2.46), (2.47), the second term on the RHS of (2.43) is given by

$$E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X}^T \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}\right] = E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X}^T \boldsymbol{\Sigma} \boldsymbol{D} \underline{\boldsymbol{\Omega}}_t \boldsymbol{Y}_t \boldsymbol{Y}_t^T \boldsymbol{Z} \tilde{\boldsymbol{\psi}}_t\right], \tag{2.51}$$

where we drop the arguments of $\underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)$ for notational simplicity and $\underline{\boldsymbol{\Omega}}_t$ denotes

$$\underline{\boldsymbol{\Omega}}_t \triangleq \begin{bmatrix} \boldsymbol{I}_{MN} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Omega}_t \end{bmatrix}.$$

Similarly, the third term on the RHS of (2.43) is evaluated as

$$E\left[\underline{\boldsymbol{h}}^T \boldsymbol{Y}_t^T \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{X} \tilde{\boldsymbol{\psi}}_t\right] = E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{Z}^T \boldsymbol{Y}_t \boldsymbol{Y}_t^T \underline{\boldsymbol{\Omega}}_t \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{X} \tilde{\boldsymbol{\psi}}_t\right]. \tag{2.52}$$

Through partitioning, the last term on the RHS of (2.43) yields

$$\begin{aligned} E\left[\underline{\boldsymbol{h}}^T(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t) \boldsymbol{Y}_t^T \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)\right] & \\ = E\left[\boldsymbol{e}_t^T \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_1 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{e}_t\right] & \\ + E\left[\boldsymbol{e}_t^T \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{C}_t \mathrm{sign}(\boldsymbol{\epsilon}_t)\right] & \\ + E\left[\mathrm{sign}(\boldsymbol{\epsilon}_t)^T \boldsymbol{C}_t^T \boldsymbol{N} \boldsymbol{\Sigma}_3 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{e}_t\right] & \\ + E\left[\mathrm{sign}(\boldsymbol{\epsilon}_t)^T \boldsymbol{C}_t^T \boldsymbol{N} \boldsymbol{\Sigma}_4 \boldsymbol{N} \boldsymbol{C}_t \mathrm{sign}(\boldsymbol{\epsilon}_t)\right]. & \end{aligned}$$

Corollary 1: *Since $\boldsymbol{U}_t$ and $\boldsymbol{C}_t$ are independent from each other, similar to the Lemma 1, we obtain*

$$\begin{aligned} E\left[\underline{\boldsymbol{h}}^T(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t) \boldsymbol{Y}_t^T \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)\right] & \\ = E\left[\boldsymbol{e}_t^T \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_1 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{e}_t\right] & \\ + E\left[\boldsymbol{e}_t^T \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_2 \boldsymbol{N} \boldsymbol{\Omega}_t \boldsymbol{C}_t \boldsymbol{\epsilon}_t\right] & \\ + E\left[\boldsymbol{\epsilon}_t^T \boldsymbol{C}_t^T \boldsymbol{\Omega}_t \boldsymbol{N} \boldsymbol{\Sigma}_3 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{e}_t\right] & \\ + E\left[\mathrm{sign}(\boldsymbol{\epsilon}_t)^T \boldsymbol{C}_t^T \boldsymbol{N} \boldsymbol{\Sigma}_4 \boldsymbol{N} \boldsymbol{C}_t \mathrm{sign}(\boldsymbol{\epsilon}_t)\right]. & \end{aligned} \tag{2.53}$$

By the Assumption 1, the first term on the RHS of (2.53) yields

$$\begin{aligned} E\left[\boldsymbol{e}_t^T \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_1 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{e}_t\right] = E\left[\boldsymbol{v}_t^T \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_1 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{v}_t\right] & \\ + E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{Z}_{\boldsymbol{u}}{}^T \boldsymbol{U}_t \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_1 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{U}_t^T \boldsymbol{Z}_u \tilde{\boldsymbol{\psi}}_t\right]. & \end{aligned} \tag{2.54}$$

For the last term on the RHS of (2.53), we introduce the following lemma.

Lemma 2: *Through the Price's theorem, we obtain*

$$\begin{aligned} &E\left[\text{sign}(\boldsymbol{\epsilon}_t)^T \boldsymbol{C}_t^T \boldsymbol{N}\boldsymbol{\Sigma}_4 \boldsymbol{N}\boldsymbol{C}_t \text{sign}(\boldsymbol{\epsilon}_t)\right] \\ &\qquad = E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{Z_d}^T \boldsymbol{C}_t \boldsymbol{C}_t^T \boldsymbol{N}\boldsymbol{\Omega}_t \boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{C}} \boldsymbol{\Omega}_t \boldsymbol{N}\boldsymbol{C}_t \boldsymbol{C}_t^T \boldsymbol{Z_d} \tilde{\boldsymbol{\psi}}_t\right] \\ &\qquad\quad + E\left[\mathbf{1}^T \boldsymbol{C}_t^T \boldsymbol{N}\boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}} \boldsymbol{N}\boldsymbol{C}_t \mathbf{1}\right], \end{aligned} \tag{2.55}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}}$ *is the block diagonal matrix of* $\boldsymbol{\Sigma}_4$ *such that*

$$\boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{\Theta}_{11} & \cdots & \mathbf{0}_M \\ \vdots & \ddots & \vdots \\ \mathbf{0}_M & \cdots & \boldsymbol{\Theta}_{NN} \end{bmatrix}$$

with $\boldsymbol{\Theta}_{ii}$ *is the* ii*'th* $M \times M$ *block of* $\boldsymbol{\Sigma}_4$ *and* $\boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{C}} = \boldsymbol{\Sigma}_4 - \boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}}$.

Proof: We derive the RHS of (2.55) for the two-node case for notational simplicity, however, the derivation holds for any order of network. For the two-node case, the LHS of (2.55) yields

$$\begin{aligned} &E\left[\text{sign}(\boldsymbol{\epsilon}_t)^T \boldsymbol{C}_t^T \boldsymbol{N}\boldsymbol{\Sigma}_4 \boldsymbol{N}\boldsymbol{C}_t \text{sign}(\boldsymbol{\epsilon}_t)\right] \\ &\qquad = E\left[\text{sign}(\epsilon_{1,t}) \boldsymbol{c}_{1,t}^T \eta_1 \boldsymbol{\varsigma_1} \eta_1 \boldsymbol{c}_{1,t} \text{sign}(\epsilon_{1,t})\right] \\ &\qquad\quad + E\left[\text{sign}(\epsilon_{1,t}) \boldsymbol{c}_{1,t}^T \eta_1 \boldsymbol{\varsigma_2} \eta_2 \boldsymbol{c}_{2,t} \text{sign}(\epsilon_{2,t})\right] \\ &\qquad\quad + E\left[\text{sign}(\epsilon_{2,t}) \boldsymbol{c}_{2,t}^T \eta_2 \boldsymbol{\varsigma_3} \eta_1 \boldsymbol{c}_{1,t} \text{sign}(\epsilon_{1,t})\right] \\ &\qquad\quad + E\left[\text{sign}(\epsilon_{2,t}) \boldsymbol{c}_{2,t}^T \eta_2 \boldsymbol{\varsigma_4} \eta_2 \boldsymbol{c}_{2,t} \text{sign}(\epsilon_{2,t})\right]. \end{aligned}$$

We re-emphasize that the regressor $\boldsymbol{c}_{i,t}$ is spatially and temporarily independent. Hence, we obtain

$$\begin{aligned} &E\left[\text{sign}(\boldsymbol{\epsilon}_t)^T \boldsymbol{C}_t^T \boldsymbol{N}\boldsymbol{\Sigma}_4 \boldsymbol{N}\boldsymbol{C}_t \text{sign}(\boldsymbol{\epsilon}_t)\right] \\ &\qquad = E\left[\boldsymbol{c}_{1,t}^T \eta_1 \boldsymbol{\varsigma_1} \eta_1 \boldsymbol{c}_{1,t}\right] + E\left[\boldsymbol{c}_{2,t}^T \eta_2 \boldsymbol{\varsigma_4} \eta_2 \boldsymbol{c}_{2,t}\right] \\ &\qquad\quad + E\left[\boldsymbol{c}_{1,t} \text{sign}(\epsilon_{1,t})\right]^T \eta_1 \boldsymbol{\varsigma_2} \eta_2 E\left[\boldsymbol{c}_{2,t} \text{sign}(\epsilon_{2,t})\right] \\ &\qquad\quad + E\left[\boldsymbol{c}_{2,t} \text{sign}(\epsilon_{2,t})\right]^T \eta_2 \boldsymbol{\varsigma_3} \eta_1 E\left[\boldsymbol{c}_{1,t} \text{sign}(\epsilon_{1,t})\right]. \end{aligned} \tag{2.56}$$

Using the Price's result, we can evaluate the last two terms on the RHS of (2.56) for $i \in \{1, 2\}$ as

$$E\left[\boldsymbol{c}_{i,t} \text{sign}(\epsilon_{i,t})\right] = \frac{E|\epsilon_{i,t}|}{E[\epsilon_{i,t}^2]} E\left[\boldsymbol{c}_{i,t} \epsilon_{i,t}\right].$$

We point out that the terms involving the diagonal entries of the weighting matrix $\boldsymbol{\Sigma}_4$ in (2.56) do not include the deviation terms. As a result, rearranging (2.56) into a compact form results in (2.55). This concludes the proof. □

As a result, by (2.51), (2.52), (2.53), (2.54) and (2.55), the relation (2.43) leads to

$$\begin{aligned} E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\Sigma}} =& E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{\Sigma}'} + E\left[\boldsymbol{v}_t^T\boldsymbol{U}_t^T\boldsymbol{M}\boldsymbol{\Sigma}_1\boldsymbol{M}\boldsymbol{U}_t\boldsymbol{v}_t\right] \\ &+ E\left[\mathbf{1}^T\boldsymbol{C}_t^T\boldsymbol{N}\boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}}\boldsymbol{N}\boldsymbol{C}_t\mathbf{1}\right] \end{aligned} \tag{2.57}$$

and

$$\begin{aligned} \boldsymbol{\Sigma}' =& \boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{X} - \boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{D}\underline{\boldsymbol{\Omega}}_t\boldsymbol{Y}_t\boldsymbol{Y}_t^T\boldsymbol{Z} - \boldsymbol{Z}^T\boldsymbol{Y}_t\boldsymbol{Y}_t^T\underline{\boldsymbol{\Omega}}_t\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{X} \\ &+ \boldsymbol{Z}^T\boldsymbol{D}\underline{\boldsymbol{\Omega}}_t\boldsymbol{Y}_t\boldsymbol{Y}_t^T\tilde{\boldsymbol{\Sigma}}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\underline{\boldsymbol{\Omega}}_t\boldsymbol{D}\boldsymbol{Z}, \end{aligned}$$

where $\tilde{\boldsymbol{\Sigma}}$ denotes

$$\tilde{\boldsymbol{\Sigma}} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_2 \\ \boldsymbol{\Sigma}_3 & \boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{C}} \end{bmatrix}.$$

We again note that by the Assumption 1, we get $\boldsymbol{\Sigma}' = E[\boldsymbol{\Sigma}']$ which results

$$\begin{aligned} \boldsymbol{\Sigma}' =& \boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{X} - \boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{D}\underline{\boldsymbol{\Omega}}_t\boldsymbol{\Lambda}\boldsymbol{Z} - \boldsymbol{Z}^T\boldsymbol{\Lambda}\underline{\boldsymbol{\Omega}}_t\boldsymbol{D}\boldsymbol{\Sigma}\boldsymbol{X} \\ &+ \boldsymbol{Z}^T\boldsymbol{D}\underline{\boldsymbol{\Omega}}_t E\left[\boldsymbol{Y}_t\boldsymbol{Y}_t^T\tilde{\boldsymbol{\Sigma}}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\right]\underline{\boldsymbol{\Omega}}_t\boldsymbol{D}\boldsymbol{Z} \end{aligned} \tag{2.58}$$

and define $\boldsymbol{B} \triangleq E\left[\boldsymbol{Y}_t\boldsymbol{Y}_t^T\tilde{\boldsymbol{\Sigma}}\boldsymbol{Y}_t\boldsymbol{Y}_t^T\right]$.

In the following, we resort to the vector notation, i.e., the block vectorisation operator bvec$\{\cdot\}$ and the block Kronecker product. Hence, the block vectorization of the weighting matrix $\boldsymbol{\Sigma}'$ (2.58) yields

$$\begin{aligned} \text{bvec}\{\boldsymbol{\Sigma}'\} =& \big(\boldsymbol{X}^T \odot \boldsymbol{X}^T - (\boldsymbol{X}^T \odot \boldsymbol{Z}^T)(\boldsymbol{I}_{2MN} \odot \boldsymbol{\Lambda}\boldsymbol{D}\underline{\boldsymbol{\Omega}}_t) \\ &-(\boldsymbol{Z}^T \odot \boldsymbol{X}^T)(\boldsymbol{\Lambda}\boldsymbol{D}\underline{\boldsymbol{\Omega}}_t \odot \boldsymbol{I}_{2MN})\big)\,\boldsymbol{\sigma} \\ &+ (\boldsymbol{Z}^T \odot \boldsymbol{Z}^T)(\boldsymbol{D} \odot \boldsymbol{D})(\underline{\boldsymbol{\Omega}}_t \odot \underline{\boldsymbol{\Omega}}_t)\text{bvec}\{\boldsymbol{B}\}. \end{aligned} \tag{2.59}$$

Block vectorisation of the matrix $\boldsymbol{B}$ is given by $\text{bvec}\{\boldsymbol{B}\} = \mathcal{A}\,\text{bvec}\{\tilde{\boldsymbol{\Sigma}}\}$. In order to denote bvec$\{\tilde{\boldsymbol{\Sigma}}\}$ in terms of $\boldsymbol{\sigma}$, we introduce $\boldsymbol{K}_1 \triangleq \text{col}\{\mathbf{0}_{MN}, \boldsymbol{I}_{MN}\}$,

$\boldsymbol{K}_2 \triangleq \mathrm{col}\{\boldsymbol{I}_{MN}, \boldsymbol{0}_{MN}\}$, and $\boldsymbol{T_k} \triangleq \mathrm{diag}\{\boldsymbol{0}_{(k-1)M}, \boldsymbol{I}_M, \boldsymbol{0}_{(N-k)M}\}$. Then, we get

$$\boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}} = \sum_{k=1}^{N} \boldsymbol{T_k} \boldsymbol{K}_2^T \boldsymbol{\Sigma} \boldsymbol{K}_2 \boldsymbol{T_k}, \tag{2.60}$$

$$\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} - \boldsymbol{K}_2 \boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}} \boldsymbol{K}_2^T. \tag{2.61}$$

By (2.60) and (2.61), we obtain

$$\begin{aligned} \mathrm{bvec}\{\tilde{\boldsymbol{\Sigma}}\} &= \underbrace{\left(\boldsymbol{I} - (\boldsymbol{K}_2 \odot \boldsymbol{K}_2) \sum_{k=1}^{N} (\boldsymbol{T_k} \odot \boldsymbol{T_k})(\boldsymbol{K_2}^T \odot \boldsymbol{K}_2^T) \right)}_{\boldsymbol{K}} \boldsymbol{\sigma} \\ &= \boldsymbol{K}\boldsymbol{\sigma}. \end{aligned} \tag{2.62}$$

The $\tilde{\boldsymbol{\psi}}$-free terms in (2.57) are evaluated as

$$E\left[\boldsymbol{v}_t^T \boldsymbol{U}_t^T \boldsymbol{M} \boldsymbol{\Sigma}_1 \boldsymbol{M} \boldsymbol{U}_t \boldsymbol{v}_t\right] = \boldsymbol{b}_1^T (\boldsymbol{K}_1^T \odot \boldsymbol{K}_1^T)\boldsymbol{\sigma}, \tag{2.63}$$

$$E\left[\boldsymbol{1}^T \boldsymbol{C}_t^T \boldsymbol{N} \boldsymbol{\Sigma}_{\boldsymbol{4}}^{\boldsymbol{D}} \boldsymbol{N} \boldsymbol{C}_t \boldsymbol{1}\right] = \boldsymbol{b}_2^T (\boldsymbol{K}_2^T \odot \boldsymbol{K}_2^T)\boldsymbol{\sigma}, \tag{2.64}$$

where $\boldsymbol{b}_1 \triangleq \mathrm{bvec}\{\boldsymbol{R_v} \boldsymbol{M}^2 \boldsymbol{\Lambda_u}\}$ and $\boldsymbol{b}_2 \triangleq \mathrm{bvec}\{\boldsymbol{1}\boldsymbol{1}^T \boldsymbol{N}^2 \boldsymbol{\Lambda_c}\}$.

As a result, by (2.59), (2.62), (2.63) and (2.64), the weighted-energy relation is given by

$$E\|\tilde{\boldsymbol{\psi}}_{t+1}\|_{\boldsymbol{\sigma}}^2 = E\|\tilde{\boldsymbol{\psi}}_t\|_{\boldsymbol{F}_t\boldsymbol{\sigma}}^2 + \boldsymbol{b}^T\boldsymbol{\sigma} \tag{2.65}$$

$$\begin{aligned} \boldsymbol{F}_t =& \boldsymbol{X}^T \odot \boldsymbol{X}^T - (\boldsymbol{X}^T \odot \boldsymbol{Z}^T)(\boldsymbol{I}_{2MN} \odot \boldsymbol{\Lambda} \boldsymbol{D} \underline{\boldsymbol{\Omega}}_t) \\ & - (\boldsymbol{Z}^T \odot \boldsymbol{X}^T)(\boldsymbol{\Lambda} \boldsymbol{D} \underline{\boldsymbol{\Omega}}_t \odot \boldsymbol{I}_{2MN}) \\ & + (\boldsymbol{Z}^T \odot \boldsymbol{Z}^T)(\boldsymbol{D} \odot \boldsymbol{D})(\underline{\boldsymbol{\Omega}}_t \odot \underline{\boldsymbol{\Omega}}_t)\mathcal{A}\boldsymbol{K} \end{aligned} \tag{2.66}$$

$$\boldsymbol{b} = (\boldsymbol{K}_1^T \odot \boldsymbol{K}_1^T)^T \boldsymbol{b}_1 + (\boldsymbol{K}_2^T \odot \boldsymbol{K}_2^T)^T \boldsymbol{b}_2. \tag{2.67}$$

Iterating the weighted-energy recursion (2.65), (2.66) and (2.67), we obtain

$$\begin{aligned} E\|\tilde{\boldsymbol{\psi}}_{t+1}\|_{\boldsymbol{\sigma}}^2 &= E\|\tilde{\boldsymbol{\psi}}_t\|_{\boldsymbol{F}_t\boldsymbol{\sigma}}^2 + \boldsymbol{b}^T\boldsymbol{\sigma} \\ E\|\tilde{\boldsymbol{\psi}}_t\|_{\boldsymbol{F}_t\boldsymbol{\sigma}}^2 &= E\|\tilde{\boldsymbol{\psi}}_{t-1}\|_{\boldsymbol{F}_{t-1}\boldsymbol{F}_t\boldsymbol{\sigma}}^2 + \boldsymbol{b}^T\boldsymbol{F}_t\boldsymbol{\sigma} \\ &\vdots \\ E\|\tilde{\boldsymbol{\psi}}_1\|_{\boldsymbol{F}_1\ldots\boldsymbol{F}_t\boldsymbol{\sigma}}^2 &= E\|\tilde{\boldsymbol{\psi}}_0\|_{\boldsymbol{F}_0\ldots\boldsymbol{F}_t\boldsymbol{\sigma}}^2 + \boldsymbol{b}^T\boldsymbol{F}_1\ldots\boldsymbol{F}_t\boldsymbol{\sigma}. \end{aligned}$$

In this part of the analyzes, we do not assume that the parameter vectors are initialized with zeros since such an assumption results in infinite terms in the $\boldsymbol{\Omega}_t$ matrix. Hence, we initialize $\boldsymbol{a}_t$ with $\zeta\,\mathbf{1}_{MN\times 1}$ where ζ has a small value (See Table 2.4).

The iterations yield

$$E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\sigma}} = \|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\Pi}_t\boldsymbol{\sigma}} + \boldsymbol{b}^T\boldsymbol{\Delta}_t\boldsymbol{\sigma}, \tag{2.68}$$

$$E\|\tilde{\boldsymbol{\psi}}_{t}\|^2_{\boldsymbol{\sigma}} = \|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\Pi}_{t-1}\boldsymbol{\sigma}} + \boldsymbol{b}^T\boldsymbol{\Delta}_{t-1}\boldsymbol{\sigma}, \tag{2.69}$$

where $\boldsymbol{\Pi}_t \triangleq \prod_{i=0}^{t}\boldsymbol{F}_i$ and $\boldsymbol{\Delta}_t \triangleq \boldsymbol{I} + \boldsymbol{F}_t + \boldsymbol{F}_{t-1}\boldsymbol{F}_t + \cdots + \boldsymbol{F}_1\ldots\boldsymbol{F}_t$. We note that $\boldsymbol{\Pi}_t = \boldsymbol{\Pi}_{t-1}\boldsymbol{F}_t$ and $\boldsymbol{\Delta}_t = \boldsymbol{\Delta}_{t-1}\boldsymbol{F}_t + \boldsymbol{I}$. By (2.68) and (2.69), we have the following recursion

$$\begin{aligned} E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\sigma}} =& E\|\tilde{\boldsymbol{\psi}}_{t}\|^2_{\boldsymbol{\sigma}} - \|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\Pi}_{t-1}(\boldsymbol{I}-\boldsymbol{F}_t)\boldsymbol{\sigma}} \\ &+ \boldsymbol{b}^T\left(\boldsymbol{I} - \boldsymbol{\Delta}_{t-1}(\boldsymbol{I}-\boldsymbol{F}_t)\right)\boldsymbol{\sigma}. \end{aligned} \tag{2.70}$$

We point out that $\boldsymbol{\Pi}_{-1} = \boldsymbol{I}_{(2MN)^2}$ and $\boldsymbol{\Delta}_{-1} = \mathbf{0}_{(2MN)^2}$.

Remark 6.1: The iterations of (2.70) require the recalculation of $\boldsymbol{F}_t$ for each time instants since $\boldsymbol{F}_t$ changes with time because of $\underline{\boldsymbol{\Omega}}_t$ (2.66). Evaluating the expectations, $\boldsymbol{\Omega}_t$ yields

$$\boldsymbol{\Omega}_t = \sqrt{\frac{2}{\pi}}\begin{bmatrix} \frac{1}{\sigma_{\epsilon_1}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_{\epsilon_N}} \end{bmatrix} \otimes \boldsymbol{I}_M, \tag{2.71}$$

where $\sigma^2_{\epsilon_i} = E[\epsilon_i^2]$. For analytical reasons, we approximate (2.71) as

$$\boldsymbol{\Omega}_t \approx \sqrt{\frac{2}{\pi}}\frac{1}{(1/\sqrt{N})\sigma_{\boldsymbol{\epsilon}_t}}\boldsymbol{I}_{MN} \tag{2.72}$$

with $\sigma^2_{\boldsymbol{\epsilon}_t} = E\left[\boldsymbol{\epsilon}_t^T\boldsymbol{\epsilon}_t\right] = E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{\xi}}$ and

$$\boldsymbol{\xi} \triangleq \text{bvec}\left\{\begin{bmatrix} \boldsymbol{\Lambda}_{\boldsymbol{c}} & -\boldsymbol{\Lambda}_{\boldsymbol{c}} \\ -\boldsymbol{\Lambda}_{\boldsymbol{c}} & \boldsymbol{\Lambda}_{\boldsymbol{c}} \end{bmatrix}\right\}.$$

Table 2.4: Initial conditions and weighting matrices for the performance measure of the construction update for the single-bit diffusion approach (for the scalar diffusion approach, set $\zeta = 0$) and the global MSD of the ATC diffusion strategy for the single-bit diffusion approach (for the scalar diffusion approach, see Table 2.3).

$E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{\Sigma}}$	$E\|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\Sigma}}$	$\boldsymbol{\Sigma}$
$\frac{1}{N}E\|\tilde{\boldsymbol{a}}_t\|^2$	$\frac{1}{N}\|\underline{\boldsymbol{w}}_{\boldsymbol{o}} - \zeta\mathbf{1}\|^2$	$\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1}{N}\boldsymbol{I}_{MN} \end{bmatrix}$
$\sigma^2_{\boldsymbol{\epsilon}_t} = E[\boldsymbol{\epsilon}_t^T\boldsymbol{\epsilon}_t]$	$\zeta\mathbf{1}^T\boldsymbol{\Lambda}_{\boldsymbol{c}}\mathbf{1}$	$\begin{bmatrix} \boldsymbol{\Lambda}_{\boldsymbol{c}} & -\boldsymbol{\Lambda}_{\boldsymbol{c}} \\ -\boldsymbol{\Lambda}_{\boldsymbol{c}} & \boldsymbol{\Lambda}_{\boldsymbol{c}} \end{bmatrix}$
$\frac{1}{N}E\|\tilde{\boldsymbol{w}}_t\|^2$	$\frac{1}{N}\|\underline{\boldsymbol{w}}_{\boldsymbol{o}} - \zeta\boldsymbol{G}_{\boldsymbol{C}}\mathbf{1}\|^2$	$\frac{1}{N}\begin{bmatrix} \boldsymbol{G}_{\boldsymbol{D}}{}^T\boldsymbol{G}_{\boldsymbol{D}} & \boldsymbol{G}_{\boldsymbol{D}}{}^T\boldsymbol{G}_{\boldsymbol{C}} \\ \boldsymbol{G}_{\boldsymbol{C}}{}^T\boldsymbol{G}_{\boldsymbol{D}} & \boldsymbol{G}_{\boldsymbol{C}}{}^T\boldsymbol{G}_{\boldsymbol{C}} \end{bmatrix}$

Hence, we can calculate $\boldsymbol{F}_t$ by iterating the following

$$\begin{aligned} E\|\tilde{\boldsymbol{\psi}}_{t+1}\|^2_{\boldsymbol{\xi}} =& E\|\tilde{\boldsymbol{\psi}}_t\|^2_{\boldsymbol{\xi}} - \|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\Pi}_{t-1}(\boldsymbol{I}-\boldsymbol{F}_t)\boldsymbol{\xi}} \\ &+ \boldsymbol{b}^T\left(\boldsymbol{I} - \boldsymbol{\Delta}_{t-1}(\boldsymbol{I}-\boldsymbol{F}_t)\right)\boldsymbol{\xi}, \end{aligned} \tag{2.73}$$

where $E\|\tilde{\boldsymbol{\psi}}_0\|^2_{\boldsymbol{\xi}} = \zeta\mathbf{1}^T\boldsymbol{\Lambda}_{\boldsymbol{c}}\mathbf{1}$. In Table 2.4, we tabulate the initial condition and the weighting matrix necessary for the recursion iterations (2.73) of $\sigma^2_{\boldsymbol{\epsilon}_t} = E[\boldsymbol{\epsilon}_t^T\boldsymbol{\epsilon}_t]$.

2.6 Steady-state Analysis

At the steady-state, (2.39) yields

$$E\|\tilde{\boldsymbol{\psi}}_{\infty}\|^2_{(\boldsymbol{I}-\boldsymbol{F})\boldsymbol{\sigma}} = \boldsymbol{b}^T\boldsymbol{\sigma}.$$

In order to calculate the steady-state performance measure $E\|\tilde{\boldsymbol{\psi}}_\infty\|^2_{\boldsymbol{\sigma}'}$, we choose the weighting matrix as $\boldsymbol{\sigma}' = (\boldsymbol{I} - \boldsymbol{F})\boldsymbol{\sigma}$, then the steady-state performance measure is given by

$$E\|\tilde{\boldsymbol{\psi}}_\infty\|^2_{\boldsymbol{\sigma}'} = \boldsymbol{b}^T(\boldsymbol{I} - \boldsymbol{F})^{-1}\boldsymbol{\sigma}'. \tag{2.74}$$

Similar to (2.74), the steady state mean square error $E[\boldsymbol{\epsilon}_t^T\boldsymbol{\epsilon}_t]$ for the single bit diffusion strategy is given by

$$E\|\tilde{\boldsymbol{\psi}}_\infty\|^2_{\boldsymbol{\xi}} = \boldsymbol{b}^T\left(\boldsymbol{I} - \boldsymbol{F}_\infty\right)^{-1}\boldsymbol{\xi}. \tag{2.75}$$

We point out that $\boldsymbol{F}_\infty$ depends on $E\|\tilde{\boldsymbol{\psi}}_\infty\|^2_{\boldsymbol{\xi}}$. Once we calculate $\boldsymbol{F}_\infty$ numerically by (2.75) or through approximations, we can obtain the steady state performance by (2.74).

2.7 Tracking Performance

The diffusion implementation improves the ability of the network to track variations in the underlying statistical profiles [6]. In this section, we analyze the tracking performance of the compressive diffusion strategies in a non-stationary environment. We assume a first-order random walk model, which is commonly used in the literature [5], for $\boldsymbol{w}_{\boldsymbol{o},t}$ such that

$$\boldsymbol{w}_{\boldsymbol{o},t+1} = \boldsymbol{w}_{\boldsymbol{o},t} + \boldsymbol{q}_t,$$

where $\boldsymbol{q}_t \in \mathbb{R}^M$ denotes a zero-mean vector process independent of the regression data and observation noise with covariance matrix $E[\boldsymbol{q}_t\boldsymbol{q}_t^T] = \boldsymbol{Q}$. We introduce the global time-variant parameter vectors as $\underline{\boldsymbol{w}}_{\boldsymbol{o},t} \overset{\triangle}{=} \text{col}\{\boldsymbol{w}_{\boldsymbol{o},t}, \cdots, \boldsymbol{w}_{\boldsymbol{o},t}\}$ and we have the global deviation vectors as $\tilde{\boldsymbol{\varphi}}_t \overset{\triangle}{=} \underline{\boldsymbol{w}}_{\boldsymbol{o},t} - \boldsymbol{\varphi}_t$ and $\tilde{\boldsymbol{a}}_t \overset{\triangle}{=} \underline{\boldsymbol{w}}_{\boldsymbol{o},t} - \boldsymbol{a}_t$. Then, by (2.26), we obtain

$$\tilde{\boldsymbol{\psi}}_{t+1} = \boldsymbol{X}\tilde{\boldsymbol{\psi}}_t - \boldsymbol{D}\boldsymbol{Y}_t\underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t) + \underline{\underline{\boldsymbol{q}_t}}, \tag{2.76}$$

where $\underline{\underline{\boldsymbol{q}_t}} \overset{\triangle}{=} \text{col}\{\boldsymbol{q}_t, \cdots, \boldsymbol{q}_t\}$ with $2MN \times 1$ dimensions. Since we assume that $\boldsymbol{q}_t$ is independent from the regression data $\boldsymbol{u}_{i,t}$, $\boldsymbol{c}_{i,t}$ and the observation noise $v_{i,t}$ for

all $i \in \{1, \cdots, N\}$, (2.76) yields the following weighted-energy relation

$$\begin{aligned} E\left[\tilde{\boldsymbol{\psi}}_{t+1}^T \boldsymbol{\Sigma} \tilde{\boldsymbol{\psi}}_{t+1}\right] = & E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X}^T \boldsymbol{\Sigma} \boldsymbol{X} \tilde{\boldsymbol{\psi}}_t\right] \\ & -E\left[\tilde{\boldsymbol{\psi}}_t^T \boldsymbol{X}^T \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)\right] \\ & -E\left[\underline{\boldsymbol{h}}^T(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t) \boldsymbol{Y}_t^T \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{X} \tilde{\boldsymbol{\psi}}_t\right] \\ & +E\left[\underline{\boldsymbol{h}}^T(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t) \boldsymbol{Y}_t^T \boldsymbol{D} \boldsymbol{\Sigma} \boldsymbol{D} \boldsymbol{Y}_t \underline{\boldsymbol{h}}(\boldsymbol{e}_t, \boldsymbol{\epsilon}_t)\right] \\ & +E\left[\underline{\underline{\boldsymbol{q}}}_t^T \boldsymbol{\Sigma} \underline{\underline{\boldsymbol{q}}}_t\right]. \end{aligned} \tag{2.77}$$

We note that (2.77) is similar to (2.43) except for the last term $E\left[\underline{\underline{\boldsymbol{q}}}_t^T \boldsymbol{\Sigma} \underline{\underline{\boldsymbol{q}}}_t\right]$. We denote $2N \times 2N$ matrix whose terms are 1 as $\underline{\underline{\mathbf{1}}}_{2N} \stackrel{\triangle}{=} [\mathbf{1}, \cdots, \mathbf{1}]$. Then, the last term in (2.77) is given by $\boldsymbol{\rho}^T \boldsymbol{\sigma}$ where $\boldsymbol{\rho} = \text{bvec}\{\underline{\underline{\mathbf{1}}}_{2N} \otimes \boldsymbol{Q}\}$. Through (2.77), we get

$$E\|\tilde{\boldsymbol{\psi}}_{t+1}\|_{\boldsymbol{\sigma}}^2 = E\|\tilde{\boldsymbol{\psi}}_t\|_{\boldsymbol{F}_t \boldsymbol{\sigma}}^2 + \boldsymbol{b}^T \boldsymbol{\sigma} + \boldsymbol{\rho}^T \boldsymbol{\sigma}. \tag{2.78}$$

We define $\boldsymbol{F}_t$ in (2.40) and (2.66) for scalar and single-bit diffusion strategies, respectively. Similarly, $\boldsymbol{b}$ is introduced in (2.38) and (2.67) for the scalar (time-invariant) and single-bit diffusion strategies. We point out that (2.78) is different from (2.39) and (2.65) only for the term $\boldsymbol{\rho}^T \boldsymbol{\sigma}$. As a result, at steady state, (2.74) and (2.78) leads

$$E\|\tilde{\boldsymbol{\psi}}_{\infty}\|_{\boldsymbol{\sigma}}^2 = (\boldsymbol{b} + \boldsymbol{\rho})^T (\boldsymbol{I} - \boldsymbol{F}_{\infty})^{-1} \boldsymbol{\sigma}. \tag{2.79}$$

Through (2.79) and Table 2.3, we can obtain the tracking performance of the network for the conventional performance measures. We point out that in the full diffusion configuration, $\boldsymbol{\rho} = \text{bvec}\{\underline{\underline{\mathbf{1}}}_{N} \otimes \boldsymbol{Q}\}$.

In the next section, we introduce the confidence parameter and the adaptive combination method, which provides a better trade-off in terms of the transient and the steady-state performances.

2.8 Confidence Parameter and Adaptive Combination

The cooperation among the nodes is not beneficial in general unless the cooperation rule is chosen properly [1]. For example, the uniform [43], the Metropolis [54], the relative-degree rules [8] and the adaptive combiners [55] provide improved convergence performance relative to the no-cooperation configuration in which nodes aim to estimate the parameter of interest $\boldsymbol{w_o}$ without information exchange. However, the compressive diffusion strategies have a different diffusion protocol than the full diffusion configuration. At each node i, we combine the local estimates $\boldsymbol{\varphi}_{i,t}$ with the constructed estimates $\boldsymbol{a}_{j,t}$ that track the local estimates $\boldsymbol{\varphi}_{j,t}$ of the neighboring nodes, i.e., $j \in \mathcal{N}_i \setminus i$. Especially at the early stages of the adaptation, the constructed estimates carry far less information than the local estimates since they are not sufficiently close to the original estimates in the mean square sense. Hence, we can consider the constructed estimates as noisy version of the original parameter vectors. Then the overall network operation is akin to the full diffusion scheme with noisy observation. In [11, 66, 67], the authors demonstrate that for imperfect cooperation cases a node should place more weight on the local estimate in the combination step even if the node has worse quality of measurement than its neighbors. To this end, we add one more freedom of dimension to the update by introducing a confidence parameter δ. The confidence parameter determines the weight of the local estimates relative to the constructed estimates such that the new combination matrix $\boldsymbol{\Gamma}'$ is given by

$$\boldsymbol{\Gamma}' = \delta \boldsymbol{I}_N + (1-\delta)\boldsymbol{\Gamma} \tag{2.80}$$

where $0 \leq \delta \leq 1$. We note that $\delta = 1$, in which case we are confident with the local estimates, yields the no-cooperation scheme and $\delta = 0$ is the full diffusion configuration where we thrust the diffused information totally.

For the new combination matrix (2.80), the combination of the local estimate

and the constructed estimates (2.12) yields

$$\boldsymbol{w}_{i,t+1} =(1-\delta)\underbrace{\left[\gamma_{i,i}\boldsymbol{\varphi}_{i,t+1}+\sum_{j\in\mathcal{N}_i\setminus i}\gamma_{i,j}\boldsymbol{a}_{j,t+1}\right]}_{\hat{\boldsymbol{\varphi}}_{i,t+1}} + \delta\boldsymbol{\varphi}_{i,t+1}. \tag{2.81}$$

We note that (2.81) is a convex combination of the parameter vectors $\hat{\boldsymbol{\varphi}}_{i,t+1}$ and $\boldsymbol{\varphi}_{i,t+1}$. Hence, we can adapt the convex combination weight δ using a stochastic gradient update [68–71]. Then, (2.81) yields

$$\boldsymbol{w}_{i,t+1} = \delta_{i,t+1}\boldsymbol{\varphi}_{i,t+1} + (1-\delta_{i,t+1})\hat{\boldsymbol{\varphi}}_{i,t+1}. \tag{2.82}$$

In [69], authors update all combination weights $\gamma_{i,j}$'s indirectly through a sigmoidal function. Similarly, we re-parameterize the confidence parameter $\delta_{i,t}$ using the sigmoidal function [72] and an unconstrained variable $\alpha_{i,t}$ such that

$$\delta_{i,t} = \frac{1}{1+e^{-\alpha_{i,t}}}. \tag{2.83}$$

We train the unconstrained weight $\alpha_{i,t}$ using a stochastic gradient update minimizing $e_{i,t}^2 = \left(d_{i,t} - \boldsymbol{u}_{i,t}^T\boldsymbol{w}_{i,t}\right)^2$ as follows

$$\begin{aligned}\alpha_{i,t+1} &= \alpha_{i,t} - \frac{1}{2}\mu_{\text{cvx}}\frac{\partial e_{i,t}^2}{\partial\alpha_{i,t}} \\ &= \alpha_{i,t} + \mu_{\text{cvx}}e_{i,t}\boldsymbol{u}_{i,t}^T(\boldsymbol{\varphi}_{i,t}-\hat{\boldsymbol{\varphi}}_{i,t})\delta_{i,t}(1-\delta_{i,t}).\end{aligned} \tag{2.84}$$

As a result, we combine the local and constructed estimates via (2.82), (2.83) and (2.84).

In the next section, we provide numerical examples showing the match of the theoretical derivations and simulated results, and the improved convergence performance with the adaptive confidence parameter.

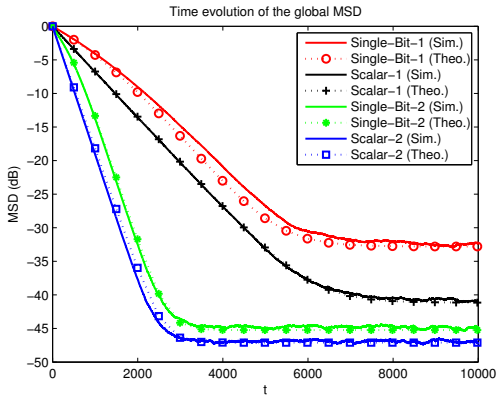
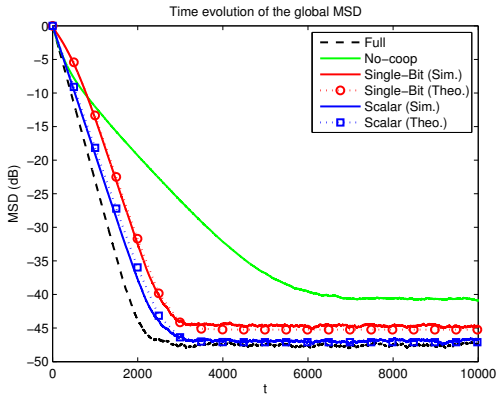


Figure 2.4: Comparison of global MSD curves $1/NE\|\tilde{\boldsymbol{\varphi}}_t\|^2$ where the single-bit-1 and the scalar-1 schemes use $\delta = 0$ while the single-bit-2 and the scalar-2 schemes have $\delta = 0.9$.

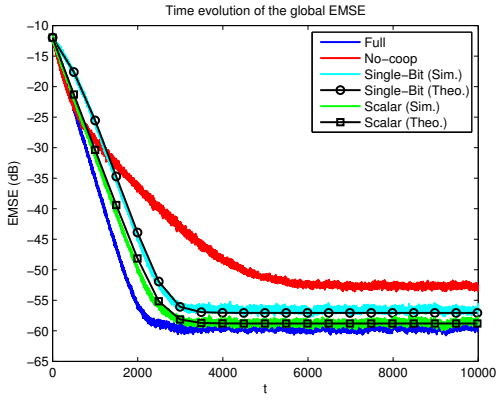
2.9 Numerical Examples

In this section, we examine two distinct network scenarios where we demonstrate that the theoretical analysis accurately model the simulated results and the confidence parameter provides significantly improved convergence performance. In the first example, we have a network of $N = 5$ nodes where at each node i, we observe a stationary data $d_{i,t} = \boldsymbol{u}_{i,t}^T \boldsymbol{w_o} + v_{i,t}$ for $i \in \{1, 2, \cdots, N\}$. The regression data $\boldsymbol{u}_{i,t}$ is a zero-mean i.i.d. Gaussian with randomly chosen standard deviation σ_{u_i}, i.e., $\sigma_{u_i} = 0.1(\sqrt{10} - 1)b_i + 0.1$ where $b_i \sim U[0, 1]$ is a uniform random variable. The variance of the observation noise is $\sigma_{v,i}^2 = 10^{-3}$. Hence, the signal-to-noise ratio over the network varies between 10 to 100. The standard deviation of the projection operator is $\sigma_{c_i} = 1$. The parameter of interest $\boldsymbol{w_o} \in \mathbb{R}^4$ is randomly chosen. Note that we examine a relatively small network with a short filter length since the computational complexity of the theoretical performance relations (2.42) and (2.70) increases exponentially with the filter length M and the network size N. We point out that the overall communication burden $(N \times M)$ in the scalar diffusion strategy is 25% of the full diffusion configuration and the overall communication load in the single-bit diffusion strategy is given by 5-bits per iterations.

In the no-cooperation configuration, the combination matrix is given by



(a) Global MSD curves



(b) Global EMSE curves

Figure 2.5: Comparison of the global MSD and EMSE curves in the CTA strategy.

$\boldsymbol{\Gamma}_0 = \boldsymbol{I}_N$. We use the Metropolis combination rule [54] for the full diffusion configuration where the adjacency matrix of the network is given by

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

In the Metropolis rule [45], the combination weights are chosen according to

$$\gamma_{i,j} = \begin{cases} \frac{1}{\max\{n_i, n_j\}} & \text{if } j \in \mathcal{N}_i \setminus i, \\ 0 & \text{if } j \notin \mathcal{N}_i, \\ 1 - \sum_{j \in \mathcal{N}_i \setminus i} \gamma_{i,j} & \text{if i = j}, \end{cases}$$

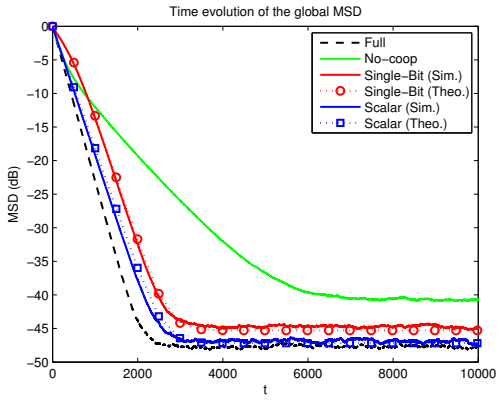


Figure 2.6: Comparison of the global MSD curves in the ATC diffusion strategy.

where n_i and n_j denote the number of neighboring nodes for i and j. For the single-bit and the one-dimension diffusion strategies we examine the convergence performance for the confidence parameter $\delta = 0$ and $\delta = 0.9$ in Fig. 2.4. We choose the step sizes the same for the distributed LMS update (2.17) of all configurations at all nodes, i.e., $\mu_i = 0.042$. At each node, the step sizes for the construction update (2.18) are $\eta_i = 0.0015$ (for single-bit approach) and $\eta_i = 0.25$ (for one-dimension diffusion approach). For the single-bit diffusion approach, we set $\zeta = 0.001$ to initialize $\boldsymbol{a}_{j,t}$. In Fig. 2.4, we show the global MSD curves, i.e., $E\|\tilde{\boldsymbol{\varphi}}_t\|^2$, of the single-bit and scalar diffusion approaches and compare the performance for different δ values. The confidence parameter $\delta = 0.9$ implies that we give ten times more weight to the local estimate $\boldsymbol{\varphi}_{i,t}$ than the constructed estimates $\boldsymbol{a}_{j,t}$, where $j \in \mathcal{N}_i \setminus i$. The Fig. 2.4 demonstrates that the confidence parameter $\delta = 0.9$ improves the convergence performance of the compressive diffusion strategies.

In the same example, Fig. 2.5 and Fig. 2.6 compare the convergence performance of the single-bit and the scalar diffusion strategies with the no-cooperation and full diffusion configurations for $\delta = 0.9$, which shows the match of the theoretical and ensemble averaged performance results (we perform 200 independent trials). The Fig. 2.5 shows the time-evolution of the MSD and EMSE curves in the CTA diffusion strategy while the Fig. 2.6 displays the time-evolution of the MSD curves in the ATC diffusion strategy in which the theoretical curves (2.42) and (2.70) are iterated according to the Table 2.3 and 2.4. We note that we obtain

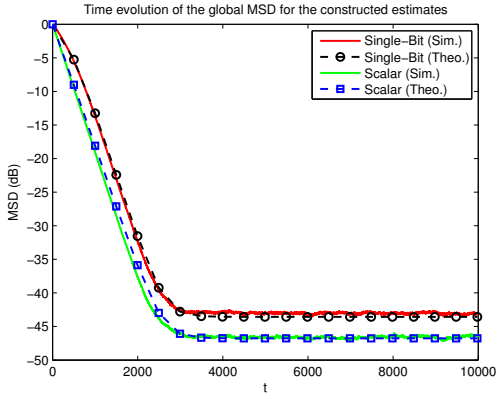
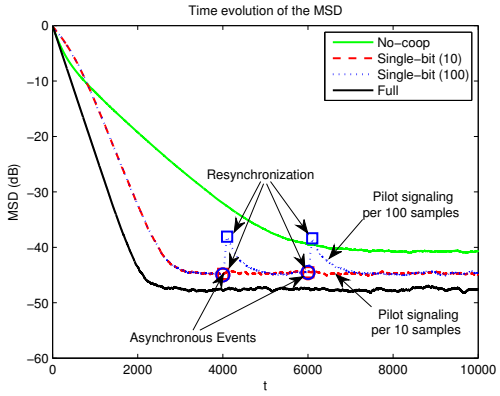


Figure 2.7: The MSD curves of the construction estimate $1/NE\|\tilde{\boldsymbol{a}}_t\|^2$ of the single-bit and scalar diffusion approaches.

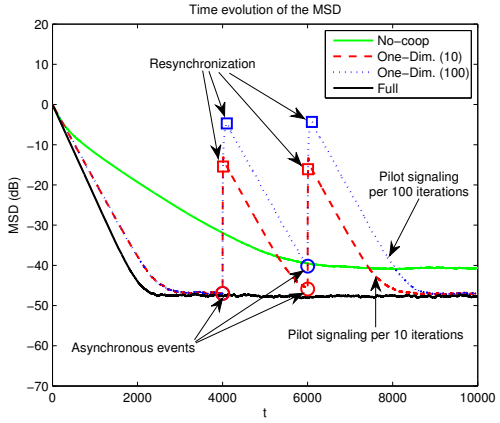
similar MSD curves in the CTA and ATC strategies since we set $\delta = 0.9$ and the outcomes of the adaptation and combination operations contain relatively close amount of information.

In Fig. 2.7, we demonstrate the convergence of the constructed estimates $\boldsymbol{a}_{j,t}$'s to the parameter of interest $\boldsymbol{w_o}$ in the mean-square sense. We point out that the recursions (2.42) and (2.70) also provide the global mean-square deviation of the constructed estimates for the certain combination weight $\boldsymbol{\Sigma}$ in Table 2.3 and the theoretical recursion matches with the simulated results.

In Fig. 2.8, we examine the impact of the synchronization issues on the estimation performance in several different scenarios. As an example, we utilize pilot signals for the re-synchronization at every 10 or 100 samples. In the asynchronous events we assume that the diffused information is completely lost and each neighboring node loses the synchronization of the projection operator until the arrival of the next pilot signal, i.e., a severe synchronization event. We point out that the single-bit diffusion strategy requires the synchronization of the construction updates in addition to the synchronization of the projection operator. Hence, the pilot signals in the single-bit diffusion scheme also re-synchronize the construction updates in each node within the neighborhood. In the Fig. 2.8, we observe 2 asynchronous events at 5001st and 6001st time instants, however, through the pilot signals at 5100th and 6100th (pilot signaling per 100 iterations)



(a) Single-bit diffusion strategy



(b) Scalar diffusion strategy

Figure 2.8: Impact of asynchronous events on the learning curves.

or at 5010th and 6010th (pilot signaling per 10 iterations) time instants each node can re-synchronize again. We note that single-bit diffusion strategy has performed less sensitive to the asynchronous events thanks to the relatively small learning rate of the construction update.

Fig. 2.9 shows the time evolution of the global MSD of the proposed schemes, i.e., both ensemble averaged and theoretical results, in a non-stationary environment. We consider a first order random walk model and choose $E[\boldsymbol{q}_t \boldsymbol{q}_t^T] = 10^{-5}\boldsymbol{I}_M$ in the same configuration of the first example. In the Fig. 2.9, we observe the match of the ensemble averaged and the theoretical results.

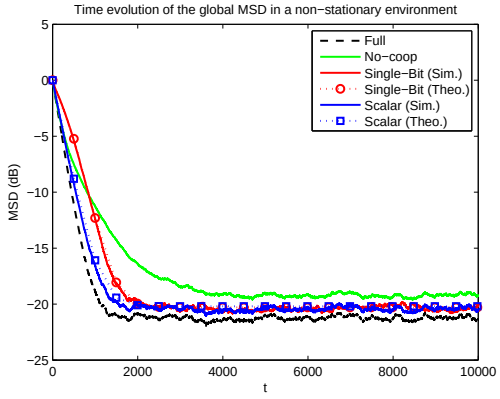


Figure 2.9: Tracking performance of the proposed schemes in a non-stationary environment.

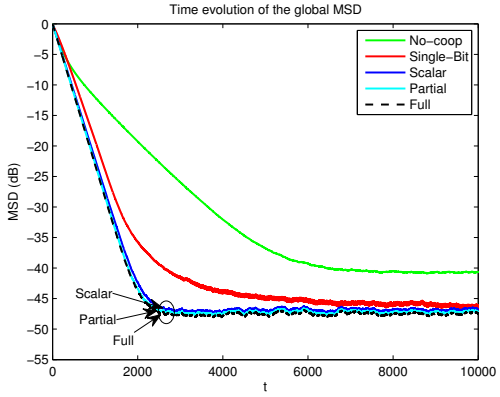


Figure 2.10: Comparison of the global MSD curves of the proposed schemes with the partial diffusion configuration.

In Fig. 2.10, we compare the time evolution of the proposed schemes with the partial diffusion strategy, where each node diffuses only one coefficient of the parameter vector. For the projection operator, we utilize a sequential selection scheme based on the round robin fashion such that

$$\boldsymbol{c}_t = \begin{cases} \text{col}\{0,0,1,1\} & \text{if } t \equiv 0 \bmod(4), \\ \text{col}\{1,-1,0,0\} & \text{if } t \equiv 1 \bmod(4), \\ \text{col}\{1,1,0,0\} & \text{if } t \equiv 2 \bmod(4), \\ \text{col}\{0,0,-1,1\} & \text{if } t \equiv 3 \bmod(4). \end{cases}$$

Note that this scheme satisfies the constraint to span the whole parameter space. Correspondingly, we use a sequential partial-diffusion scheme such that each node shares the same coefficients in order. In the proposed schemes, we choose $\delta = 0.9$

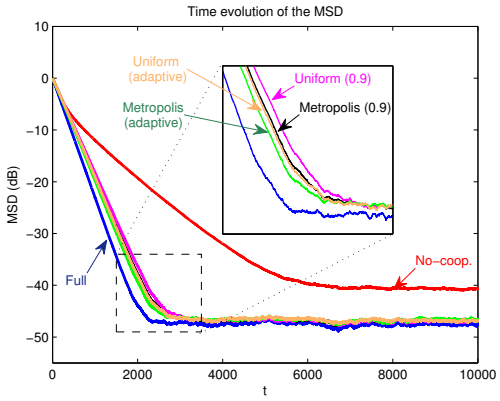


Figure 2.11: Comparison of the adaptive and fixed confidence parameter for the Metropolis and uniform combination rules.

and the step sizes of the all schemes are $\mu_i = 0.042$. For the construction updates, $\eta_i = 0.0035$ and $\eta_i = 0.75$ in the single-bit and scalar diffusion strategies, respectively. The Fig. 2.10 shows that sequential selection scheme provides enhanced estimation performance also for the compressive diffusion strategies. In the Fig. 2.10, we also observe that both scalar diffusion and partial diffusion approaches achieve comparable performance.

We can enhance the performance of the scheme through the adaptation of the confidence parameter irrespective of the cooperation rule. As an example, in Fig. 2.11, we compare the time evolution of the MSD of the scalar diffusion scheme for the adaptive and fixed confidence parameter cases with the Metropolis and uniform combination rules. We use the same configuration with the example 1, initialize $\alpha_{i,t} = 2$, and set $\mu_{cvx} = 10$. Additionally, in Fig. 2.12, we also plot the time-evolution of the scalar diffusion scheme for $\delta = 0.95$. Note that the adaptive scheme converges as fast as $\delta = 0.95$ scheme while achieving smaller steady-state error similar to $\delta = 0.9$ scheme. Hence, through the confidence parameter we can enhance the performance of the compressive diffusion scheme for certain scenarios.

In the second example, we examine the convergence performance of the adaptive confidence parameter in a relatively large network $N = 20$ with a long filter length $M = 100$. We again observe a stationary data $d_{i,t} = \boldsymbol{u}_{i,t}^T \boldsymbol{w_o} + v_{i,t}$ for

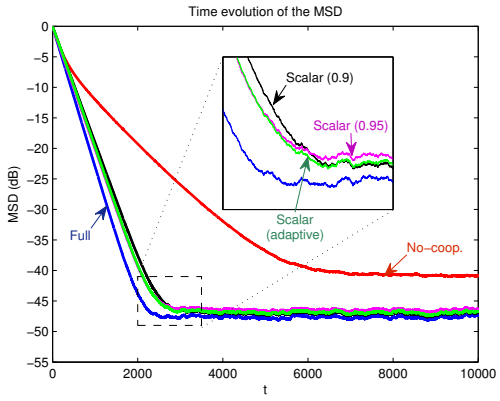


Figure 2.12: Comparison of the adaptive and fixed confidence parameter for the scalar diffusion scheme.

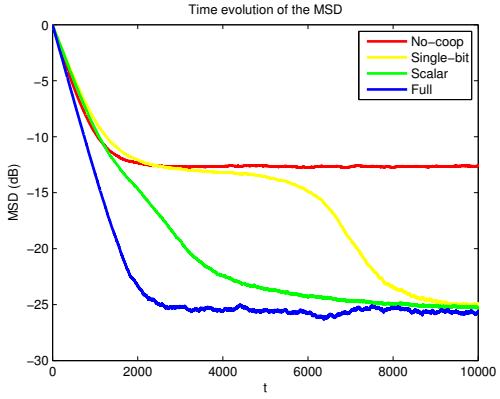


Figure 2.13: The global MSD curves in relatively large network and long filter length while the confidence parameter is adapted in time.

$i \in \{1, 2, \cdots, N\}$. The regressor data $\boldsymbol{u}_{i,t}$ is zero-mean i.i.d. Gaussian whose standard deviation is around 0.4. The observation noise $v_{i,t}$ is zero-mean i.i.d. Gaussian whose variance is $\sigma_{n_i}^2 = 10^{-1}$. We note that the signal-to-noise ratio is around 1.55 over the network, which is relatively lower than the signal-to-noise ratio for the example 1. The variance of the projection operator $\boldsymbol{c}_{i,t}$ is $\sigma_{c_i}^2 = 10^{-2}$ ($\sigma_{c_i}^2 = 10^{-3}$) for the scalar (single-bit) diffusion scheme. The parameter of interest $\boldsymbol{w_o} \in \mathbb{R}^{100}$ is randomly chosen from a Gaussian distribution and normalized such that $\|\boldsymbol{w_o}\| = 1$. We point out that in this example, the overall communication burden in the scalar diffusion strategy is **1%** of the full diffusion configuration while the overall communication load in the single-bit diffusion strategy is given by 20-bits per iterations.

We again use the Metropolis rule as the combination rule, however, in this example, we adapt the confidence parameter through (2.83) and (2.84) where we resort to the convex mixture of the adaptive filtering algorithms [68–71]. We also choose the step sizes the same for the distributed LMS update (2.17) of all configurations at all nodes, i.e., $\mu_i = 0.01$. In example 2, the step sizes for the construction update (2.18) are $\eta_i = 0.01$ (for the single-bit diffusion approach) and $\eta_i = 0.5$ (for the scalar diffusion approach). We set $\mu_{\text{cvx}} = 250$ in (2.84). The Fig. 2.13 shows the global MSD curves of the no-cooperation, single-bit, scalar and full diffusion strategies. We observe that the adaptive confidence parameter improves the convergence performance of the compressive diffusion strategies far more such that they achieve comparable performance while the reduction of the communication load is tremendous.

Chapter 3

Application: Communication Efficient Distributed Channel Estimation

The demand on distributed networks (or processing units) is steadily growing due to increased efficiency and performance improvements they provide in various different applications [73]. The broadened perspective provided by these architectures significantly enhances channel estimation performance; is used to avoid environmental obstructions; or permit resource sharing and allocation. However, these architectures demand astounding amount of communication between their nodes causing significant energy consumption and increase in carbon footprint. Due to growing awareness of telecommunication industry's impact on the environment, the need to mitigate this carbon footprint is indisputable. To this end, we introduce novel approaches substantially reducing the communication load for the distributed architectures, which is the main source of energy consumption, without any significant performance degradation [73].

In particular, we investigate "diffusion" based distributed architectures in a channel identification (or estimation) framework, where distributed nodes are

used for channel estimation and share their information to improve overall estimation accuracy. The diffusion based distributed algorithms define a strategy in which the nodes from a predefined neighborhood share information with each other [6, 7, 74]. Such approaches that diffuse information to their neighbors instead of using a central processing units are stable against time-varying statistical profiles [6], however, entail a high amount of communication load. For example, in a network of N nodes, where M denotes the number of channel coefficient, then the overall communication burden among nodes is given by $N \times M$ at each instant, which can be highly impractical for certain applications.

We propose diffusion based cooperation strategies that have significantly less communication load (e.g., a single bit or a couple of bits of information exchange) and achieve comparable performance to the full information exchange configurations under certain settings. In this framework, each node estimates an unknown communication channel observed through a linear model. After local estimates of the desired channel is produced in each node, a single bit or multiple bits of information is generated using certain random projections of the local estimates. This new information is then diffused and utilized in neighboring nodes instead of the original estimates; significantly reducing the communication load in the network. We only require synchronization of this randomized projection operation, which can be achieved using simple pilot signals [75].

Our approach differs from quantization based diffusion strategies such as [19], where quantized parameter estimates are exchanged among nodes to avoid infinite precision, in terms of the compression of the diffused information. Here, we substantially compress the exchanged information, even to a single bit, and perform local adaptive operations at each node to recover the full channel information. In this sense, our method is more akin to compressive sensing rather than to a quantization framework. To this end, we propose algorithms to significantly reduce the amount of communication between nodes for diffusion based distributed strategies and illustrate the comparable convergence performance of these algorithms in different numerical examples. We also emphasize that our algorithms are generic and can be straightforwardly extended to perform prediction, hypothesis testing or filtering in diffusion based distributed algorithms with

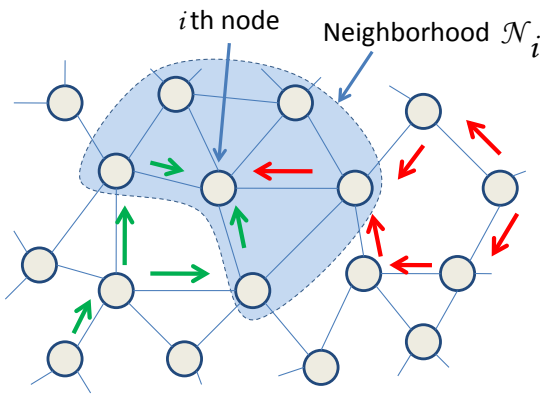


Figure 3.1: A distributed network of nodes.

significant reduction in communication load.

3.1 Distributed Channel Estimation Framework

Consider a network of nodes distributed spatially as shown in Fig. 3.1, which models a wide range of applications including spectrum sensing in cognitive radio networks to distributed processing in multi-cores [6, 7, 73, 74]. Here, we have n distributed nodes and two nodes are considered neighbors if they can exchange information, where we assume that the information exchange is bi-directional. For each node i, we denote the set of its neighbors (including itself) as $\mathcal{N}_i$. Here, each node estimates an unknown communication channel $\boldsymbol{w}_o \in \mathbb{R}^m$ observed through a linear model $d_i(t) = \boldsymbol{u}_i^T(t)\boldsymbol{w}_o + v_i(t)$, where the observations are corrupted by additive white noise and diffuses its information to neighboring nodes. The observation noise is temporally and spatially white (or independent), i.e., $E[v_i(t)v_j(l)] = \sigma_i^2\delta(i-j)\delta(t-l)$, where $\delta(\cdot)$ is the Kronecker delta and σ_i^2 is the variance of the noise. The underlying $\boldsymbol{w}_o$ can also represent spectrum parameters or a state vector of an unknown system[1] in different applications, where our derivations still hold. The linear transformation (or the regressor) $\boldsymbol{u}_i(t)$ is known by the node $\mathcal{N}_i$ but unknown to the other nodes. We also assume that $\boldsymbol{u}_i(t)$'s are

[1] Although we assume a time invariant desired vector, our derivations can be readily extended to certain non-stationary models [5].

spatially and temporally uncorrelated with each other and with the observation noise.

At each node an adaptive channel estimation algorithm is used to recover $\boldsymbol{w}_o$ such as a stochastic gradient approach [5] given as

$$\boldsymbol{\phi}_i(t+1) = (\boldsymbol{I} - \mu_i \boldsymbol{u}_i(t)\boldsymbol{u}_i^T(t))\boldsymbol{w}_i(t) + \mu_i d_i(t)\boldsymbol{u}_i(t), \tag{3.1}$$

$\mu_i > 0$, where $\boldsymbol{w}_i(t)$ represents the current estimate and $\boldsymbol{\varphi}_i(t+1)$ is the updated estimate after the new observation. We emphasize that our approach is generic such that one can use different estimation algorithms instead of (3.1) [5]. As the diffusion strategy, we next use the adapt-then-combine (ATC) diffusion strategy as an example, however, our derivations also cover other diffusion strategies [6]. In the ATC strategy, at each node i, the final channel estimate is constructed as

$$\boldsymbol{w}_i(t+1) = \sum_{k \in \mathcal{N}_i} \lambda_{i,k} \boldsymbol{\phi}_k(t+1), \tag{3.2}$$

after the estimates of the neighboring nodes arrive to i, where $\lambda_{i,k}$'s are the combination weights $\sum_{k \in \mathcal{N}_i} \lambda_{i,k} = 1$ and $\lambda_{i,k} \geq 0$. The combination weights $\lambda_{i,k}$ can also be adapted in time, however, we use weights that are constant in time with the simplex constraint (since the stabilization effect of such weights is demonstrated in [6]).

In the diffusion based distributed networks, the entire channel estimates $\boldsymbol{\phi}_k(t+1)$'s are exchanged within the neighborhood, which requires a substantial amount of communication between the nodes even if efficient quantization methods are used [19]. In the next section, we study algorithms that significantly reduce the amount of information exchange between the neighboring nodes.

3.2 A Second Estimation Level Instead of Diffusion

In the well known formulation (3.2), each node i receives the entire vector of estimated channel coefficients from all its neighbors. This requires an exchange

of $O(m)$ "real" coefficients for each node. Instead of directly transmitting the estimated vector of coefficients, we introduce a different perspective [75]. We first observe that for the node i, the estimated channel coefficients of its neighbors, $\boldsymbol{\varphi}_k(t+1)$, $k \in \mathcal{N}_i$, are naturally unknown. Hence, instead of collecting these unknown coefficients from the neighboring nodes, we next formulate another estimation level, where the node i (in addition to its original task) estimates $\boldsymbol{\varphi}_k(t+1)$'s of its neighbors. In this case instead of the original $\boldsymbol{\varphi}_k(t+1)$'s, each node i constructs estimated vectors, say $\boldsymbol{a}_k(t+1)$'s, of $\boldsymbol{\varphi}_k(t+1)$'s that are directly used in (3.2) instead of the original $\boldsymbol{\varphi}_k(t+1)$'s [75]. We demonstrate through this perspective, we can tremendously reduce the communication load, i.e., from a continuum to a couple of bits, while providing nearly equal performance to the original formulation.

For the node i, $\boldsymbol{\varphi}_k(t+1)$ is unknown and suppose both nodes i and k calculates a linear transformation of $\boldsymbol{\varphi}_k(t+1)$ through a randomized linear transformation $\boldsymbol{c}^T(t+1)\boldsymbol{\varphi}_k(t+1)$, where construction of $\boldsymbol{c}(t+1)$ is detailed later in the paper. Then, if $\boldsymbol{a}_k(t+1)$ accurately represents $\boldsymbol{\varphi}_k(t+1)$, then $\boldsymbol{c}^T(t+1)\boldsymbol{a}_k(t+1)$ should be close to $\boldsymbol{c}^T(t+1)\boldsymbol{\varphi}_k(t+1)$. In this sense, in this new framework, $\boldsymbol{\varphi}_k(t+1)$ is the desired vector of coefficients and $\boldsymbol{a}_k(t+1)$ is our estimate. Hence, we formulate another, i.e., a second level of, estimation framework at node i to recover $\boldsymbol{\varphi}_k(t+1)$ by a stochastic gradient algorithm by minimizing the estimation error as

$$\begin{aligned}\boldsymbol{a}_k(t+1) &= \boldsymbol{a}_k(t) + \rho_k \nabla_{\boldsymbol{a}_k} \epsilon_k^2(t+1) \\ &= \boldsymbol{a}_k(t) + \rho_k \epsilon_k(t+1)\boldsymbol{c}(t+1), \end{aligned} \tag{3.3}$$

where we have $\epsilon_k(t+1) \stackrel{\triangle}{=} \boldsymbol{c}^T(t+1)\boldsymbol{\varphi}_k(t+1) - \boldsymbol{c}^T(t+1)\boldsymbol{a}_k(t)$, i.e., the estimation error, and $\rho_k > 0$ is the learning rate. After we perform the update in (3.3), we then use $\boldsymbol{a}_k(t+1)$ at the node i instead of $\boldsymbol{\varphi}_k(t+1)$ in (3.2).

For this formulation, the node k needs to provide only the scalar $\epsilon_k(t+1)$ instead of the whole $\boldsymbol{\varphi}_k(t+1)$ to the node i since all the other quantities are known by both nodes i and k. Note that both nodes i and k can synchronously run the same (3.3) provided that $\epsilon_k(t+1)$ is communicated to i from k, i.e., the node k can also construct $\boldsymbol{a}_k(t+1)$ in order to construct $\epsilon_k(t+1)$, which is transmitted to the node i.

To accomplish this effectively, we next introduce an adaptive quantization scheme in order to effectively and efficiently construct $\epsilon_k(t+1)$ at the node i for all $k \in \mathcal{N}_i$. If the estimation scheme is successful, then $\epsilon_k(t+1)$ converges to a white noise process due to the linear filtering formulation in (3.3). Hence adaptive quantizers using linear predictive formulations are usually ineffective since there should be no correlation left in $\epsilon_k(t)$ if estimation is successful. In order to effectively perform quantization, we next introduce a scalar adaptive quantization framework with guaranteed synchronization between nodes. Note that one can straightforwardly extend our formulation to a vector quantization framework, however, we use a scalar quantization framework for notational simplicity.

At time t, we quantize $\epsilon_k(t+1)$ as $\tilde{\epsilon}_k(t+1) = Q_{k,t}\left(\epsilon_k(t+1)\right)$ at the node k, where $Q_{k,t}(\cdot) : \mathbb{R} \to \{q_{k,t,1}, \ldots, q_{k,t,2^b}\}$ is a time adaptive quantizer using b-bit as explained in the following, and transmit the quantized bits to the node i. We also assume that $\epsilon_k(t+1)$ is Gaussian distributed where this assumption is widely used in the literature [5]. Hence, to construct $Q_{k,t}(\cdot)$, we need the variance and the mean of the process $\epsilon_k(t+1)$. To adaptively estimate the mean and variance of the process, we can only use the previous quantized bits $\tilde{\epsilon}_k(1), \ldots, \tilde{\epsilon}_k(t)$ to guarantee synchronization between the nodes i and k. To this end, we use a recursive mean estimator

$$\tilde{m}_k(t+1) = (1-\eta_k)\tilde{m}_k(t) + \eta_k \tilde{\epsilon}_k(t),$$

and a recursive variance estimator

$$\tilde{\sigma}_k^2(t+1) = (1-\eta_k)\tilde{\sigma}_k^2(t) + \eta_k \tilde{\epsilon}_k^2(t),$$

using only the quantized error samples [76], where the forgetting factors $\eta_k > 0$ are set equal for notational simplicity. Based on $\tilde{m}_k(t)$, $\tilde{\sigma}_k(t)$, and assuming that the estimation error is Gaussian distributed, we construct

$$\{q_{k,t,1}, \ldots, q_{k,t,2^b}\} = \arg \min_{q_1,\ldots,q_{2^b}} \int_{-\infty}^{\infty} \frac{(x - Q(x))^2}{\sqrt{2\pi\tilde{\sigma}_k^2(t)}} \exp\left\{ -\frac{[x - \tilde{m}_k(t)]^2}{2\tilde{\sigma}_k^2(t)} \right\} dx,$$

where

$$Q(x) \stackrel{\triangle}{=} \arg \min_{q_i \in \{q_1,\ldots,q_{2^b}\}} \|x - q_i\|^2,$$

yielding the mean square error optimal quantization algorithm (if the estimated mean and variance converge). We next provide a mean stability analysis of the overall diffusion based algorithm.

3.3 Mean Stability Analysis

The diffusion update at the node i with the second layer of adaptation can be written in a compact form as

$$\boldsymbol{\phi}_i(t+1) = \boldsymbol{w}_i(t) + \mu_i e_i(t)\boldsymbol{u}_i(t), \tag{3.4}$$

$$\boldsymbol{a}_k(t+1) = \boldsymbol{a}_k(t) + \rho_k Q_{k,t}\left(\epsilon_k(t+1)\right)\boldsymbol{c}(t+1), \tag{3.5}$$

$$\boldsymbol{w}_i(t+1) = \lambda_{i,i}\boldsymbol{\varphi}_i(t+1) + \sum_{k\in\mathcal{N}_i\setminus i}\lambda_{i,k}\boldsymbol{a}_k(t+1), \tag{3.6}$$

where $\mu_i > 0$ and $\rho_k > 0$. Note that (3.5) is also carried out in the neighboring nodes $k \in \mathcal{N}_i \setminus i$. Since we have two estimation algorithms, we define deviations from the parameter of interests as

$$\Delta\boldsymbol{\varphi}_k(t+1) = \boldsymbol{h}_o - \boldsymbol{\varphi}_k(t+1), \tag{3.7}$$

$$\Delta\boldsymbol{a}_k(t+1) = \boldsymbol{\varphi}_k(t+1) - \boldsymbol{a}_k(t+1). \tag{3.8}$$

Substituting (3.8) and (3.7) into (3.6), we get the final estimate as

$$\boldsymbol{h}_i(t+1) = \sum_{k\in\mathcal{N}_i}\lambda_{i,k}\boldsymbol{\varphi}_k(t+1) - \sum_{k\in\mathcal{N}_i\setminus i}\lambda_{i,k}\,\Delta\boldsymbol{a}_k(t+1).$$

To continue with the mean stability analysis, we make the following assumptions:

1) $\boldsymbol{c}(t)$ and $\boldsymbol{u}_k(t)$ are temporally independent.

2) The error $\epsilon_k(t)$ and $\boldsymbol{c}(t)$ are jointly Gaussian and uncorrelated. For sufficiently small step size and long filter length, this assumption is true [5].

3) The original parameter estimates $\boldsymbol{\varphi}_i(t)$ vary slowly relative to the constructed estimates $\boldsymbol{a}_i(t)$ through the appropriate step sizes such that

$$\Delta\boldsymbol{a}_k(t) = \boldsymbol{\varphi}_k(t) - \boldsymbol{a}_k(t) \cong \boldsymbol{\varphi}_k(t+1) - \boldsymbol{a}_k(t) \text{ or}$$

$$\Delta\boldsymbol{a}_k(t+1) = \boldsymbol{\varphi}_k(t+1) - \boldsymbol{a}_k(t+1) \cong \boldsymbol{\varphi}_k(t) - \boldsymbol{a}_k(t+1).$$

4) The quantization error $\Delta\epsilon_k(t+1) = \epsilon_k(t+1) - Q_{t,k}\left(\epsilon_k(t+1)\right)$ is i.i.d. with zero mean independent from the regressors and observation noise processes [76].

To construct a complete state space recursion, we define the following global variables

$$\boldsymbol{U}(t) \stackrel{\triangle}{=} \begin{bmatrix} \boldsymbol{u}_1(t) & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{u}_N(t) \end{bmatrix},$$

$$\boldsymbol{v}(t) \stackrel{\triangle}{=} \begin{bmatrix} v_1(t) \\ \vdots \\ v_N(t) \end{bmatrix}, \; \Delta\boldsymbol{\varphi}(t) \stackrel{\triangle}{=} \begin{bmatrix} \Delta\boldsymbol{\varphi}_1(t) \\ \vdots \\ \Delta\boldsymbol{\varphi}_N(t) \end{bmatrix},$$

$$\Delta\boldsymbol{a}(t) \stackrel{\triangle}{=} \begin{bmatrix} \Delta\boldsymbol{a}_1(t) \\ \vdots \\ \Delta\boldsymbol{a}_N(t) \end{bmatrix}, \; \Delta\boldsymbol{\epsilon}(t) \stackrel{\triangle}{=} \begin{bmatrix} \Delta\epsilon_1(t) \\ \vdots \\ \Delta\epsilon_N(t) \end{bmatrix}.$$

Using these global variables, for the first adaptation layer, we get

$$\begin{aligned} \Delta\boldsymbol{\varphi}(t+1) = &\left(\boldsymbol{I} - \boldsymbol{D}\boldsymbol{U}(t)\boldsymbol{U}(t)^T\right)\boldsymbol{G}\;\Delta\boldsymbol{\varphi}(t) - \\ &\left(\boldsymbol{I} - \boldsymbol{D}\boldsymbol{U}(t)\boldsymbol{U}(t)^T\right)\tilde{\boldsymbol{G}}\;\Delta\boldsymbol{a}(t) + \boldsymbol{D}\boldsymbol{U}(t)\boldsymbol{v}(t), \end{aligned} \tag{3.9}$$

where $\boldsymbol{G} \stackrel{\triangle}{=} \boldsymbol{\Lambda} \otimes \boldsymbol{I}_m$ is the transition matrix (and $\otimes$ is the Kronecker product), $\tilde{\boldsymbol{G}} \stackrel{\triangle}{=} \boldsymbol{G} - \operatorname{diag}(\boldsymbol{G})$, $\boldsymbol{\Lambda} \stackrel{\triangle}{=} [\lambda_{i,k}]$ is the combination matrix and $\boldsymbol{D} \stackrel{\triangle}{=} \operatorname{diag}([\mu_1, \mu_2, ..., \mu_N]) \otimes \boldsymbol{I}_m$.

The global update for the reconstructed parameters yields

$$\Delta\boldsymbol{a}(t+1) = \left(\boldsymbol{I} - \boldsymbol{S}\boldsymbol{M}(t)\right)\Delta\boldsymbol{a}(t) + \boldsymbol{S}\boldsymbol{L}(t)\Delta\boldsymbol{\epsilon}(t+1), \tag{3.10}$$

where $\boldsymbol{S} \stackrel{\triangle}{=} \operatorname{diag}([\rho_1, \rho_2, ..., \rho_N]) \otimes \boldsymbol{I}_m$, $\boldsymbol{L}(t) \stackrel{\triangle}{=} \boldsymbol{I}_m \otimes \boldsymbol{c}(t+1)$ and

$$\boldsymbol{M}(t) = \boldsymbol{I}_m \otimes \left(\frac{\boldsymbol{c}(t+1)\boldsymbol{c}(t+1)^T}{\boldsymbol{c}(t+1)^T\boldsymbol{c}(t+1)}\right).$$

If we calculate the expectations of (3.9) and (3.10) under our assumptions, then we get

$$\begin{bmatrix} \Delta\overline{\boldsymbol{\varphi}}(t+1) \\ \Delta\overline{\boldsymbol{a}}(t+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}\boldsymbol{G} & \boldsymbol{B}\tilde{\boldsymbol{G}} \\ \boldsymbol{0} & \boldsymbol{I} - \boldsymbol{S}\overline{\boldsymbol{M}}(t) \end{bmatrix} \begin{bmatrix} \Delta\overline{\boldsymbol{\varphi}}(t) \\ \Delta\overline{\boldsymbol{a}}(t) \end{bmatrix}, \tag{3.11}$$

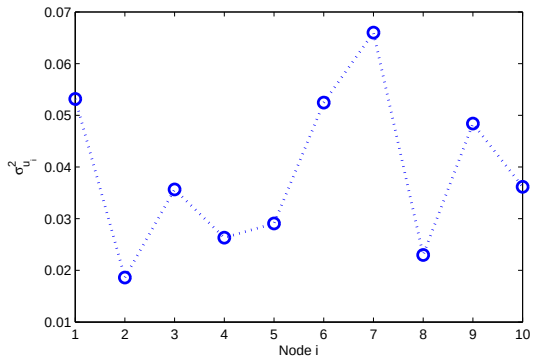


Figure 3.2: Statistical profile of the example network ($\sigma_v^2 = 0.01$).

where $\boldsymbol{B} \stackrel{\triangle}{=} \boldsymbol{I} - \boldsymbol{D}\overline{\boldsymbol{U}(t)\boldsymbol{U}(t)^T}$. From (3.11) we observe that our algorithms are stable in the mean if $|\lambda\left(\boldsymbol{I} - \boldsymbol{S}\overline{\boldsymbol{M}}\right)| < 1$ (provided that the full diffusion scheme is stable), where $\lambda(\cdot)$'s are the eigenvalues. Assuming $\boldsymbol{c}(\cdot)$ are i.i.d. zero mean with unit variance, then $|\lambda\left(\boldsymbol{I} - \boldsymbol{S}\overline{\boldsymbol{M}}\right)| < 1$ if and only if $|1 - \rho_i| < 1$ for all i.

3.4 Numerical Example

We compare the proposed diffusion algorithms with the scalar, full diffusion and the no-cooperation schemes for the example network with $n = 10$ nodes and $m = 10$. Each node i observes a stationary data $d_i(t) = \boldsymbol{u}_i^T(t)\boldsymbol{h}_o + v_i(t)$, where $\boldsymbol{u}_i(t)$ is i.i.d. zero mean Gaussian vector process with auto-covariance matrix $\boldsymbol{C} = \sigma_{u_i}^2\boldsymbol{I}$ and $\sigma_{u_i}^2$ are seen in Fig. 3.2. The observation noise $v_i(t)$ is zero-mean i.i.d. Gaussian random process with variance $\sigma_v^2 = 0.01$. We set the true channel coefficients $\boldsymbol{h}_o \in \mathbb{R}^{10}$ randomly from a normal distribution and normalize to have $\|\boldsymbol{h}_o\| = 1$.

We combine the estimation parameters through a modified Metropolis rule where

$$\lambda_{i,k} = \begin{cases} \frac{2}{m^2}\frac{1}{\max(n_i,n_k)} & \text{if } i \neq k \text{ are linked,} \\ 0 & \text{for } i \text{ and } k \text{ not linked,} \\ 1 - \sum_{k\in\mathcal{N}_i\setminus i}\lambda_{i,k} & \text{for } i = k \end{cases}$$

and n_i denotes the cardinality of the neighborhood $\mathcal{N}_i$.

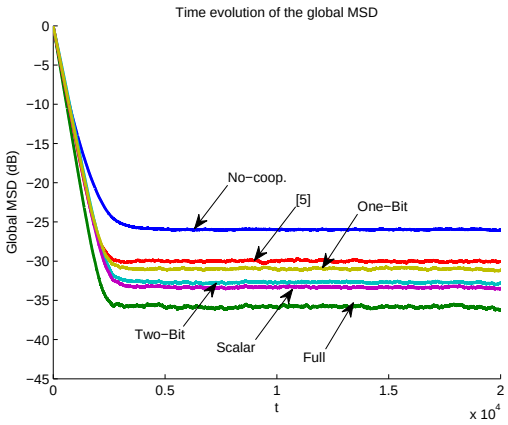


Figure 3.3: Global mean-square deviation (MSD) of diffusion and no-cooperation schemes.

We set the step sizes as 0.05 for the estimation update (3.4) and 0.01 for the construction update (3.5) (for the scalar diffusion $\rho_k = 0.1$) so that they converge with the same rate. The forgetting factor η_k is set to 0.02. The randomized projection vector $\boldsymbol{c}(t)$ is generated i.i.d. Normal random process. In Fig. 3.3, we compare the global mean-square deviation (MSD) of the diffusion schemes. We observe that introduced schemes enhance the estimation performance of the single-bit diffusion strategy and through the diffusion of two-bit we can achieve almost identical performance with the scalar diffusion strategy.

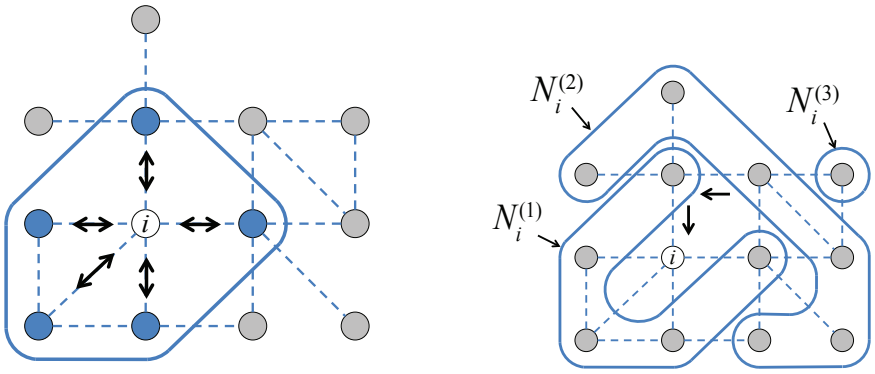
Chapter 4

Optimal Distributed Static State Estimation

In this chapter, we introduce a thorough cost measure considering the transmission of information over hops across networks. We derive the ODOL algorithm achieving the MMSE performance over arbitrary networks through the aggregation of information at each agent. We study the network structures in which we can achieve the MMSE performance through the disclosure of local estimates. We propose the OEDOL algorithm achieving the MMSE performance over certain network topologies with tremendously reduced communication load. We formulate sub-optimal versions of the algorithms with reduced complexity for practical applications. We provide numerical examples demonstrating the significant gains due to the introduced algorithms.

4.1 Distributed-MMSE Estimation Framework

Consider a sparsely connected distributed network of m agents with processing and communication capabilities. In Fig. 4.1, we illustrate this network through an undirected graph, where the vertices and the edges correspond to the agents



(a) First-order neighborhood (b) All neighborhoods

Figure 4.1: The neighborhoods of ith agent over the distributed network.

and the communication links across the network, respectively. For each agent i, we denote the set of agents whose information could be received at least after k hops, i.e., k-hop neighbors, by $N_i^{(k)}$ defined as

$$N_i^{(k)} \triangleq \left\{ j_1, \cdots, j_{\pi_i^{(k)}} \right\}$$

and $\pi_i^{(k)} = \left| N_i^{(k)} \right|$ is the cardinality of $N_i^{(k)}$ (See Fig. 4.1b). We assume that $N_i^{(0)} = \{i\}$ and $N_i^{(k)} = \varnothing$ for $k < 0$. Here, the agents observe a noisy version of a time-invariant and unknown state vector $\boldsymbol{x} \in \mathbb{R}^p$ which is a realization of the Gaussian random vector process $\underline{\mathcal{X}}$ with mean $\bar{\boldsymbol{x}}$ and auto-covariance $\boldsymbol{\Sigma}_x$. Each agent i observes the state as

$$\boldsymbol{y}_{i,t} = \boldsymbol{H}_i \boldsymbol{x} + \boldsymbol{n}_{i,t},$$

where $\boldsymbol{H}_i \in \mathbb{R}^{q \times p}$ is a known matrix, $\boldsymbol{n}_{i,t} \in \mathbb{R}^q$ is a realization of a zero-mean white Gaussian vector process $\underline{\mathcal{N}}_{i,t}$ with auto-covariance $\boldsymbol{\Sigma}_{n_i}$ and correspondingly the observation $\boldsymbol{y}_{i,t} \in \mathbb{R}^q$ is a realization of the random process $\underline{\mathcal{Y}}_{i,t} = \boldsymbol{H}_i \underline{\mathcal{X}} + \underline{\mathcal{N}}_{i,t}$. The noise $\boldsymbol{n}_{i,t}$ is also independent from the state and the other noise parameters.

We assume that the variance of the noise signals are known, which can also be readily estimated from the data [5]. At each instant, the agents observe the underlying state through $\boldsymbol{y}_{i,t}$, diffuse information to the neighboring agents and receive information from them as seen in Fig. 4.1a. Naturally, the agents can learn the true state, under certain regularity conditions [5], irrespective of cooperation

among them, provided that their own measurements are not biased and sufficient to estimate the true state in time. However, through the cooperation of agents, we can increase the learning rate significantly [24]. To this end, we aim to find the optimal learning strategy over distributed networks in the MSE sense.

We first consider the case when there is no limitation on the amount of the disclosed information. Agents disclose their own measurements and transmit the received information from the neighboring agents (with time stamps, i.e., the information is ordered). Through this implementation, each agent can obtain the measurements of the other agents separately in a connected network. We point out that the information on the non-neighboring agents could only be received after certain hops due to the sparsely connected structure. As an example, the disclosed information of $j \in N_i^{(2)}$ reaches to i by passing through two communication links as seen in Fig. 4.1b. In particular, this case assumes that each agent has access to full information from the other agents, albeit with certain hops, and corresponds to the direct aggregation of all information across the network at each agent.

At time t, all the information aggregated at ith agent is given by

$$\left\{ \{y_{i,\tau}\}^{\tau \leq t}, \{y_{j,\tau}\}_{j \in N_i^{(1)}}^{\tau \leq t-1}, \cdots, \{y_{j,\tau}\}_{j \in N_i^{(\kappa_i)}}^{\tau \leq t-\kappa_i} \right\},$$

where the information from the furthest agent is received at least after κ_i hops[1]. Here, the collection set $\{y_{j,\tau}\}_{j \in N_i^{(k)}}^{\tau \leq t-k}$ includes the observations from the k-hop neighbors $N_i^{(k)}$. Particularly, the collection is explicitly defined as

$$\begin{aligned} \{y_{j,\tau}\}_{j \in N_i^{(k)}}^{\tau \leq t-k} \triangleq \Big\{ & y_{j_1,t-k}, ..., y_{j_1,0}, \\ & ..., y_{j_{\pi_i^{(k)}},t-k}, ..., y_{j_{\pi_i^{(k)}},0} \Big\}. \end{aligned} \tag{4.1}$$

Then, we have the comprehensive cost measure in the distributed-MMSE framework as

$$\begin{aligned} \hat{\boldsymbol{x}}_{i,t} = \min_{\boldsymbol{x}} E \Bigg[\left\| \underline{\mathcal{X}}_t - \boldsymbol{x} \right\|^2 \Bigg| & \left\{ \underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau} \right\}^{\tau \leq t}, \\ & \left\{ \underline{\mathcal{Y}}_{j,\tau} = \boldsymbol{y}_{j,\tau} \right\}_{j \in N_i^{(1)}}^{\tau \leq t-1}, \cdots, \left\{ \underline{\mathcal{Y}}_{j,\tau} = \boldsymbol{y}_{j,\tau} \right\}_{j \in N_i^{(\kappa_i)}}^{\tau \leq t-\kappa_i} \Bigg] \end{aligned} \tag{4.2}$$

[1]For notational simplicity, we denote $N_i \triangleq N_i^{(1)}$ and $\pi_i \triangleq \pi_i^{(1)}$.

and the MMSE estimate for each agent i is given by the expectation of $\boldsymbol{x}$ conditioned all the accessed information:

$$\boldsymbol{x}_{i,t} = E\Bigg[\underline{\mathcal{X}}\Big|\left\{\underline{\mathcal{Y}}_{i,\tau}=\boldsymbol{y}_{i,\tau}\right\}^{\tau\leq t},\left\{\underline{\mathcal{Y}}_{j,\tau}=\boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(1)}}^{\tau\leq t-1},\cdots,\left\{\underline{\mathcal{Y}}_{j,\tau}=\boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(\kappa_i)}}^{\tau\leq t-\kappa_i}\Bigg]. \tag{4.3}$$

Correspondingly, we define the random variable $\hat{\mathcal{X}}_{i,t}$ as follows

$$\hat{\mathcal{X}}_{i,t} \triangleq E\Bigg[\underline{\mathcal{X}}\Big|\left\{\underline{\mathcal{Y}}_{i,\tau}\right\}^{\tau\leq t},\cdots,\left\{\underline{\mathcal{Y}}_{j,\tau}\right\}_{j\in N_i^{(\kappa_i)}}^{\tau\leq t-\kappa_i}\Bigg].$$

4.1.1 ODOL Algorithm

Importantly, since the state and the observation noise are independent Gaussian random parameters, we propose the ODOL algorithm constructing the distributed-MMSE estimator (4.3) in an iterative way. To this end, we collect all received information at time t as

$$\boldsymbol{z}_{i,t} = \mathrm{col}\Big\{\boldsymbol{y}_{i,t},\boldsymbol{y}_{i_1,t-1},...,\boldsymbol{y}_{i_{\pi_i},t-1},...,\boldsymbol{y}_{j_1,t-\kappa_i},...,\boldsymbol{y}_{j_{\pi_i^{(\kappa_i)}},t-\kappa_i}\Big\}.$$

Then, the iterations of the ODOL algorithm are given by

$$\begin{aligned}
\boldsymbol{K}_{i,t} &= \hat{\boldsymbol{\Sigma}}_{i,t-1}\bar{\boldsymbol{H}}_i^T\left(\bar{\boldsymbol{H}}_i\hat{\boldsymbol{\Sigma}}_{i,t-1}\bar{\boldsymbol{H}}_i^T+\bar{\boldsymbol{\Sigma}}_{n_i}\right)^{-1},\\
\hat{\boldsymbol{x}}_{i,t} &= \left(\boldsymbol{I}-\boldsymbol{K}_{i,t}\bar{\boldsymbol{H}}_i\right)\hat{\boldsymbol{x}}_{i,t-1}+\boldsymbol{K}_{i,t}\boldsymbol{z}_{i,t},\\
\hat{\boldsymbol{\Sigma}}_{i,t} &= \left(\boldsymbol{I}-\boldsymbol{K}_{i,t}\bar{\boldsymbol{H}}_i\right)\hat{\boldsymbol{\Sigma}}_{i,t-1},
\end{aligned}$$

where[2] $\bar{\boldsymbol{H}}_i \triangleq (\boldsymbol{P}_i\otimes\boldsymbol{I}_q)\,\boldsymbol{H}$, $\boldsymbol{H}\triangleq \mathrm{col}\{\boldsymbol{H}_1,\cdots,\boldsymbol{H}_m\}$, $\bar{\boldsymbol{\Sigma}}_{n_i}\triangleq(\boldsymbol{P}_i\otimes\boldsymbol{I}_p)\,\boldsymbol{\Sigma}_n\,(\boldsymbol{P}_i\otimes\boldsymbol{I}_p)^T$, $\boldsymbol{\Sigma}_n\triangleq\mathrm{diag}\{\boldsymbol{\Sigma}_{n_1},\cdots,\boldsymbol{\Sigma}_{n_m}\}$, and $\boldsymbol{P}_i$ is the corresponding permutation matrix. We provide the detailed description of the ODOL algorithm in Table 4.1.

Remark 4.1.1: The ODOL algorithm achieves the best possible MSE performance over a sparsely connected network by collecting all information directly at

[2]If the inverse fails to exist, a pseudo inverse can replace the inverse [77]

Table 4.1: The description of the ODOL algorithm.

Algorithm 4.1: The ODOL Algorithm

Initialization:

For $i = 1$ to m do

$\hat{\boldsymbol{x}}_{i,-1} = \bar{\boldsymbol{x}}$, $\hat{\boldsymbol{\Sigma}}_{i,-1} = \boldsymbol{\Sigma}_x$ and $\boldsymbol{y}_{j,\tau} = \boldsymbol{0}$ for $\tau < 0$

End for

Update:

Do for $t \geq 0$

For $i = 1$ to m do

Construct $\boldsymbol{z}_{i,t}$ *through* $\boldsymbol{y}_{i,t}$ *and received* $\boldsymbol{z}_{j,t-1}$ *for* $j \in N_i$

$$\boldsymbol{z}_{i,t} = \text{col}\left\{ \boldsymbol{y}_{i,t}, \boldsymbol{y}_{i_1,t-1}, ..., \boldsymbol{y}_{j_{\pi_i(\kappa_i)},t-\kappa_i} \right\}$$

$$\boldsymbol{K}_{i,t} = \hat{\boldsymbol{\Sigma}}_{i,t-1}\bar{\boldsymbol{H}}_i^T \left(\bar{\boldsymbol{H}}_i \hat{\boldsymbol{\Sigma}}_{i,t-1} \bar{\boldsymbol{H}}_i^T + \bar{\boldsymbol{\Sigma}}_{n_i} \right)^{-1}$$

$$\hat{\boldsymbol{x}}_{i,t} = \left(\boldsymbol{I} - \boldsymbol{K}_{i,t}\bar{\boldsymbol{H}}_i \right) \hat{\boldsymbol{x}}_{i,t-1} + \boldsymbol{K}_{i,t}\boldsymbol{z}_{i,t}$$

$$\hat{\boldsymbol{\Sigma}}_{i,t} = \left(\boldsymbol{I} - \boldsymbol{K}_{i,t}\bar{\boldsymbol{H}}_i \right) \hat{\boldsymbol{\Sigma}}_{i,t-1}$$

Disclose $\boldsymbol{z}_{i,t}$ *to the neighboring agents.*

End for

each agent and processing them optimally, however, this yields excessive communication load in general. Since the communication load is crucial for the applicability of the distributed learning algorithms [12, 47], we also seek to reduce the communication load yet achieve the MMSE performance. In the consensus and diffusion implementations, the agents disclose the local estimates, which requires relatively reduced communication load [6, 24, 33, 34]. However, in the sequel we show that the disclosure of the local estimates is sufficient to achieve the MMSE performance *only* over certain network topologies.

4.2 Distributed-MMSE Estimation with Propagation of Information

Since the underlying state and the observation noises are independent Gaussian random parameters, the conditional expectation of the state given the observations, i.e., the MMSE estimate, is an affine combination of the observations. Hence, in this section we aim to exploit the affine transformation in order to reduce the disclosed amount of information so that each agent exchanges the necessary information in an efficient way. To this end, we utilize the following lemma implying that the MMSE estimate could be obtained through the conditional expectations of the state given distinct observation sets.

Lemma 4.2.1: Let $\boldsymbol{\mathcal{M}}$ be the set of all observations at time t in (4.3) such that $\hat{\boldsymbol{x}}_{i,t} = E\left[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}} = M\right]$. We partition $\boldsymbol{\mathcal{M}}$ as $\bigcup_{i=1}^{r} \boldsymbol{\mathcal{M}}_i = \boldsymbol{\mathcal{M}}$ through disjoint sets $\boldsymbol{\mathcal{M}}_i$ and M_i is the corresponding set of the realizations. Then, we obtain

$$E\left[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}} = M\right] = E\Big[\underline{\mathcal{X}}\Big|E\big[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_1\big] = E\big[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_1 = M_1\big], \cdots, E\big[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_r\big] = E\big[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_r = M_r\big]\Big]. \tag{4.4}$$

Proof of Lemma 4.2.1: We note that for any function $f(\cdot)$ we have [77]

$$E\big[(\mathcal{X} - E[\mathcal{X}|\mathcal{Y}_1 = y_1])^2\big] \leq E\big[(\mathcal{X} - f(y_1))^2\big], \tag{4.5}$$

which leads to

$$E\bigg[\Big(\mathcal{X} - E\Big[\mathcal{X}\Big|E[\mathcal{X}|\mathcal{Y}_1] = E[\mathcal{X}|\mathcal{Y}_1 = y_1]\Big]\Big)^2\bigg] \leq E\Big[\big(\mathcal{X} - f\big(E[\mathcal{X}|\mathcal{Y}_1 = y_1]\big)\big)^2\Big].$$

Note that $f(x) = x$ yields

$$E\bigg[\Big(\mathcal{X} - E\Big[\mathcal{X}\Big|E[\mathcal{X}|\mathcal{Y}_1] = E[\mathcal{X}|\mathcal{Y}_1 = y_1]\Big]\Big)^2\bigg] \leq E\big[(\mathcal{X} - E[\mathcal{X}|\mathcal{Y}_1 = y_1])^2\big]. \tag{4.6}$$

Then, by (4.5) and (4.6), we obtain

$$E\Big[\mathcal{X}\Big|E[\mathcal{X}|\mathcal{Y}_1] = E[\mathcal{X}|\mathcal{Y}_1 = y_1]\Big] = E\big[\mathcal{X}|\mathcal{Y}_1 = y_1\big]. \tag{4.7}$$

Next, we consider $E\big[\mathcal{X}|\mathcal{Y}_1 = y_1, \mathcal{Y}_2 = y_2\big]$. Since $\mathcal{X}, \mathcal{Y}_1$ and $\mathcal{Y}_2$ are all jointly Gaussian, the conditioned random variables $\mathcal{X}|\mathcal{Y}_1$ and $\mathcal{Y}_2|\mathcal{Y}_1$ are also jointly Gaussian. Due to zero-mean independent noise term we have

$$E\big[\mathcal{X}|\mathcal{Y}_1 = y_1, \mathcal{Y}_2 = y_2\big] = g\big(E\big[\mathcal{X}|\mathcal{Y}_1 = y_1\big], y_2\big),$$

where $g(\cdot, \cdot)$ is a linear function.

For any function $f(\cdot, \cdot)$ we have

$$E\Big[\big(\mathcal{X} - E\big[\mathcal{X}|\mathcal{Y}_1 = y_1, \mathcal{Y}_2 = y_2\big]\big)^2\Big] \leq E\big[(\mathcal{X} - f(y_1, y_2))^2\big] \tag{4.8}$$

and

$$\begin{aligned} E\Bigg[\Big(\mathcal{X} - E\Big[\mathcal{X}\big|E\big[\mathcal{X}|\mathcal{Y}_1\big] = E\big[\mathcal{X}|\mathcal{Y}_1 = y_1\big], \mathcal{Y}_2 = y_2\Big]\Big)^2\Bigg] \\ \leq E\Bigg[\Big(\mathcal{X} - f\Big(E\big[\mathcal{X}|\mathcal{Y}_1 = y_1\big], y_2\Big)\Big)^2\Bigg]. \end{aligned} \tag{4.9}$$

Setting $f(\cdot, \cdot) = g(\cdot, \cdot)$, (4.8) and (4.9) yield

$$\begin{aligned} E\Bigg[\Big(\mathcal{X} - E\Big[\mathcal{X}\big|E\big[\mathcal{X}|\mathcal{Y}_1\big] = E\big[\mathcal{X}|\mathcal{Y}_1 = y_1\big], \mathcal{Y}_2 = y_2\Big]\Big)^2\Bigg] \\ \leq E\Big[\big(\mathcal{X} - E\big[\mathcal{X}|\mathcal{Y}_1 = y_1, \mathcal{Y}_2 = y_2\big]\big)^2\Big], \end{aligned}$$

which implies that

$$\begin{aligned} &E\big[\mathcal{X}|\mathcal{Y}_1 = y_1, \mathcal{Y}_2 = y_2\big] \\ &\quad = E\Big[\mathcal{X}\big|E\big[\mathcal{X}|\mathcal{Y}_1\big] = E\big[\mathcal{X}|\mathcal{Y}_1 = y_1\big], \mathcal{Y}_2 = y_2\Big]. \end{aligned}$$

Correspondingly, changing the order, we obtain

$$\begin{aligned} &E\big[\mathcal{X}|\mathcal{Y}_1 = y_1, \mathcal{Y}_2 = y_2\big] \\ &\qquad = E\Big[\mathcal{X}\big|E\big[\mathcal{X}|\mathcal{Y}_1\big] = E\big[\mathcal{X}|\mathcal{Y}_1 = y_1\big], \\ &\qquad\qquad E\big[\mathcal{X}|\mathcal{Y}_2\big] = E\big[\mathcal{X}|\mathcal{Y}_2 = y_2\big]\Big]. \end{aligned} \tag{4.10}$$

Then, by (4.7) and (4.10) for a given disjoint partitioning of $\boldsymbol{\mathcal{M}}$, i.e., $\boldsymbol{\mathcal{M}}_1, \cdots, \boldsymbol{\mathcal{M}}_r \subset \boldsymbol{\mathcal{M}}$, we obtain (4.4) and the proof is concluded. □

We emphasize that Lemma 4.2.1 does not imply the sufficiency of the disclosure of the local estimates to obtain the distributed-MMSE estimate. As a counter example, consider a cycle network of 4 agents where agent-1 is directly connected with agents 2 and 3. At time $t = 1$, we obtain

$$\hat{\boldsymbol{x}}_{2,1} = E\Big[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_{2,1} = \boldsymbol{y}_{2,1}, \underline{\mathcal{Y}}_{2,0} = \boldsymbol{y}_{2,0}, \underline{\mathcal{Y}}_{1,0} = \boldsymbol{y}_{1,0}, \underline{\mathcal{Y}}_{4,0} = \boldsymbol{y}_{4,0}\Big],$$
$$\hat{\boldsymbol{x}}_{3,1} = E\Big[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_{3,1} = \boldsymbol{y}_{3,1}, \underline{\mathcal{Y}}_{3,0} = \boldsymbol{y}_{3,0}, \underline{\mathcal{Y}}_{1,0} = \boldsymbol{y}_{1,0}, \underline{\mathcal{Y}}_{4,0} = \boldsymbol{y}_{4,0}\Big].$$

Since $\hat{\boldsymbol{x}}_{2,1}$ and $\hat{\boldsymbol{x}}_{3,1}$ are conditioned on $\underline{\mathcal{Y}}_{4,0} = \boldsymbol{y}_{4,0}$ commonly, the MMSE estimator $\hat{\boldsymbol{x}}_{1,2}$ could not be obtained by just processing the current estimates, i.e.,

$$\hat{\boldsymbol{x}}_{1,2} \neq E\Big[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_{1,2} = \boldsymbol{y}_{1,2}, \hat{\underline{\mathcal{X}}}_{1,1} = \hat{\boldsymbol{x}}_{1,1}, \hat{\underline{\mathcal{X}}}_{2,1} = \hat{\boldsymbol{x}}_{2,1}, \hat{\underline{\mathcal{X}}}_{3,1} = \hat{\boldsymbol{x}}_{3,1}\Big].$$

In particular, for i.i.d. observations $\underline{\mathcal{Y}}_1$, $\underline{\mathcal{Y}}_2$ and $\underline{\mathcal{Y}}_3$ we have

$$E\Big[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_1, \underline{\mathcal{Y}}_2, \underline{\mathcal{Y}}_3\Big] \neq E\Big[\underline{\mathcal{X}}|E\big[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_1, \underline{\mathcal{Y}}_2\big], E\big[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_2, \underline{\mathcal{Y}}_3\big]\Big].$$

Hence, optimal processing of estimates should be considered elaborately. In the following, we analytically show that the MMSE estimate could be obtained through the disclosure of local estimates over tree networks.

4.2.1 Tree Networks

A network has a "tree structure" if its corresponding graph is a *tree*, i.e., connected and undirected without any cycles [78]. As an example, the conventional star or line networks have tree structures. We remark that for an arbitrary network topology we can also construct the spanning tree of the network and eliminate the cycles. In the literature, there exists numerous distributed algorithms for minimum spanning tree construction [79–83].

Importantly, the following theorem shows that over tree networks we can achieve the performance of the oracle algorithm through the disclosure of the local estimates only.

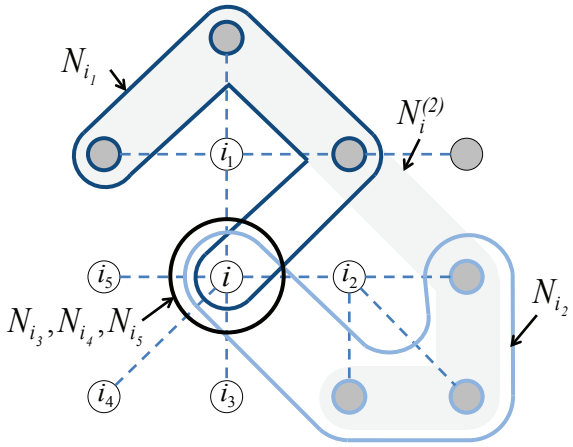


Figure 4.2: An example tree network. Notice the eliminated links from Fig. 4.1 to avoid multi path information diffusion.

Theorem 4.2.1: Consider the distributed estimation framework over a tree network. Then, the distributed-MMSE estimator (4.3) could also be obtained by

$$\hat{\boldsymbol{x}}_{i,t} = E\left[\underline{\mathcal{X}} \middle| \left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq t}, \left\{\hat{\mathcal{X}}_{j,\tau} = \hat{\boldsymbol{x}}_{j,\tau}\right\}_{j\in N_i}^{\tau\leq t-1}\right], \tag{4.11}$$

i.e., each agent can only disclose its own local estimate to achieve the MMSE performance.

Proof of Theorem 4.2.1: Initially, agent i has access to $\boldsymbol{y}_{i,0}$ only and the MMSE estimator is $\hat{\boldsymbol{x}}_{i,0} = E[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_{i,0} = \boldsymbol{y}_{i,0}]$. At time $t = 1$, the MMSE estimator is given by

$$\hat{\boldsymbol{x}}_{i,1} = E\left[\underline{\mathcal{X}} \middle| \left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 1}, \left\{\underline{\mathcal{Y}}_{j,0} = \boldsymbol{y}_{j,0}\right\}_{j\in N_i}\right]. \tag{4.12}$$

By Lemma 3.1, (4.12) could also be obtained by

$$\begin{aligned}\hat{\boldsymbol{x}}_{i,1} &= E\left[\underline{\mathcal{X}} \middle| \left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 1}, \left\{E[\underline{\mathcal{X}}|\underline{\mathcal{Y}}_{j,0} = \boldsymbol{y}_{j,0}]\right\}_{j\in N_i}\right], \\ &= E\left[\underline{\mathcal{X}} \middle| \left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 1}, \left\{\hat{\mathcal{X}}_{j,0} = \hat{\boldsymbol{x}}_{j,0}\right\}_{j\in N_i}\right].\end{aligned} \tag{4.13}$$

As seen in Fig. 4.2, over a tree network, for $k \in \{1, \cdots, \kappa_i\}$ we have

$$N_i^{(k)} = \bigcup_{j\in N_i} \left(N_i^{(k)} \cap N_j^{(k-1)}\right). \tag{4.14}$$

Note that the sets in (4.14) are disjoint as

$$\left(N_i^{(k)} \cap N_{j_1}^{(k-1)}\right) \cap \left(N_i^{(k)} \cap N_{j_2}^{(k-1)}\right) = \varnothing \tag{4.15}$$

for all $j_1, j_2 \in N_i$ and $j_1 \neq j_2$. Notably, over a tree network, by (4.15), we can partition the collection set defined in (4.1) as follows

$$\begin{aligned}\left\{\boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(k)}} = \Bigg\{ & \left\{\boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(k)}\cap N_{j_1}^{(k-1)}}, \\ & \cdots, \left\{\boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(k)}\cap N_{j_{\pi_i}}^{(k-1)}} \Bigg\}.\end{aligned} \tag{4.16}$$

Let the set of all accessed information by the ith agent at time $t = 2$ be

$$Z_{i,2} \triangleq \left\{ \left\{\boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 2}, \left\{\boldsymbol{y}_{j,\tau}\right\}_{j\in N_i}^{\tau\leq 1}, \left\{\boldsymbol{y}_{k,0}\right\}_{k\in N_i^{(2)}} \right\}.$$

We also define

$$\hat{Z}_{j,i,1} \triangleq \left\{ \overbrace{\left\{\boldsymbol{y}_{k,1}\right\}_{k\in N_i^{(1)}\cap N_j^{(0)}}}^{=\boldsymbol{y}_{j,1}}, \left\{\boldsymbol{y}_{k,0}\right\}_{k\in N_i^{(2)}\cap N_j^{(1)}} \right\}$$

such that by (4.16) we have

$$Z_{i,2} = \left\{\boldsymbol{y}_{i,2}, \hat{Z}_{j_1,i,1}, \cdots, \hat{Z}_{j_{\pi_i},i,1}, Z_{i,1}\right\} \tag{4.17}$$

and

$$\hat{Z}_{j,i,1} = Z_{j,1} \setminus \left\{ \overbrace{\boldsymbol{y}_{j,0}}^{=Z_{j,0}}, \boldsymbol{y}_{i,0} \right\}. \tag{4.18}$$

Hence, by (4.17) and (4.18), we obtain

$$\begin{aligned}\hat{\boldsymbol{x}}_{i,2} = E\Bigg[& \underline{\mathcal{X}} \Big| \left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 2}, \left\{\boldsymbol{\mathcal{Z}}_{j,0} = Z_{j,0}\right\}_{j\in N_i}, \\ & \left\{\boldsymbol{\mathcal{Z}}_{j,1} \setminus \left\{\boldsymbol{\mathcal{Z}}_{j,0}, \underline{\mathcal{Y}}_{i,0}\right\} = Z_{j,1} \setminus \left\{Z_{j,0}, \boldsymbol{y}_{i,0}\right\} \right\}_{j\in N_i} \Bigg].\end{aligned} \tag{4.19}$$

By the following lemma and after some algebra, (4.19) could also be obtained by

$$\begin{aligned}\hat{\boldsymbol{x}}_{i,2} &= E\left[\underline{\mathcal{X}} \Big| \left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 2}, \left\{ E\left[\underline{\mathcal{X}} \middle| \boldsymbol{\mathcal{Z}}_{j,\tau} = Z_{j,\tau}\right] \right\}_{j\in N_i}^{\tau\leq 1} \right], \\ &= E\left[\underline{\mathcal{X}} \Big| \left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 2}, \left\{\hat{\mathcal{X}}_{j,\tau} = \hat{\boldsymbol{x}}_{j,\tau}\right\}_{j\in N_i}^{\tau\leq 1} \right].\end{aligned} \tag{4.20}$$

Lemma 4.2.2: In the context of Lemma 3.1, let $\boldsymbol{\mathcal{M}}_1, \boldsymbol{\mathcal{M}}_2, \boldsymbol{\mathcal{M}}_3 \subset \boldsymbol{\mathcal{M}}$ be disjoint partitions of $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{M}}_4 = \boldsymbol{\mathcal{M}}_2 \cup \boldsymbol{\mathcal{M}}_3$. Then, we obtain

$$\begin{aligned} &E\big[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_1 = M_1, \boldsymbol{\mathcal{M}}_2 = M_2, \boldsymbol{\mathcal{M}}_3 = M_3\big] \\ &\quad = E\Big[\underline{\mathcal{X}}\big|\boldsymbol{\mathcal{M}}_1 = M_1, E[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_2] = E[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_2 = M_2], \\ &\qquad E[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_4] = E[\underline{\mathcal{X}}|\boldsymbol{\mathcal{M}}_4 = M_4]\Big] \end{aligned} \tag{4.21}$$

Proof of Lemma 4.2.2: When we condition $\mathcal{X}$ on the sets $\boldsymbol{\mathcal{M}}_2$ and $\boldsymbol{\mathcal{M}}_3$, we obtain $E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_2 = M_2, \boldsymbol{\mathcal{M}}_3 = M_3\big] = E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_4 = M_4\big]$ and (4.7) yields

$$E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_4 = M_4\big] = E\Big[\mathcal{X}|E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_4\big] = E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_4 = M_4\big]\Big]. \tag{4.22}$$

Since $\boldsymbol{\mathcal{M}}_2 \subset \boldsymbol{\mathcal{M}}_4$, (4.22) leads to

$$\begin{aligned} &E\Big[\mathcal{X}|E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_4 = M_4\big]\Big] \\ &\quad = E\Big[\mathcal{X}|E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_2\big] = E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_2 = M_2\big], \\ &\qquad E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_4\big] = E\big[\mathcal{X}|\boldsymbol{\mathcal{M}}_4 = M_4\big]\Big]. \end{aligned}$$

Then, if we condition on $\boldsymbol{\mathcal{M}}_1$, we get (4.21) and the proof is concluded. □

At time $t = 3$, (4.17) yields

$$\hat{Z}_{j,i,2} = Z_{j,2} \setminus \Big\{Z_{j,1} \cup \overbrace{\big\{Z_{i,1} \setminus \{Z_{i,0} \cup Z_{j,0}\}\big\}}^{=\hat{Z}_{i,j,1}}\Big\}. \tag{4.23}$$

Correspondingly, (4.23) leads to

$$\hat{Z}_{j,i,t} = Z_{j,t} \setminus \big\{Z_{j,t-1} \cup \hat{Z}_{i,j,t-1}\big\}$$

implying that $\hat{Z}_{j,i,t}$ is constructible from $Z_{i,\tau}$ and $Z_{j,\tau}$ for $\tau \leq t$. Hence, by the Lemmas 3.1 and 3.2 for $t > 0$ the MMSE estimator over tree networks is given by (4.11) and the proof is concluded. □

In the sequel, we propose the optimal and efficient distributed online learning (OEDOL) algorithm that achieves the distributed-MMSE performance over tree networks iteratively.

4.2.2 OEDOL Algorithm

We remark that the MMSE estimator $\hat{\boldsymbol{x}}_{i,t}$ linearly depends on $\hat{\boldsymbol{x}}_{i,t-1}$, which has been diffused previously. In order to extract the new information, we need to eliminate the previously received information at each instant on the neighboring agents. This brings in additional computational complexity. On the contrary, agents can just disclose the new information, i.e., $\hat{\boldsymbol{z}}_{i,t}$, after eliminating all the previously exchanged information. Since we are conditioning on the linear combinations of the conditioned variables without effecting their spanned space, i.e., $\hat{\boldsymbol{z}}_{i,t}$ is computable from $\hat{\boldsymbol{x}}_{i,\tau}$ for $\tau \leq t$ and vice versa, we can still achieve the MMSE performance by reduced computational load, yet.

At time t, agent i observes $\boldsymbol{y}_{i,t}$ and receives $\bar{\boldsymbol{z}}_{i,t} = \mathrm{col}\left\{\hat{\boldsymbol{z}}_{j_1,t-1}, \cdots, \hat{\boldsymbol{z}}_{j_{\pi_i},t-1}\right\}$. Here, we determine the content of the received information to extract the innovation within them and utilize this innovation in the update of the local estimate. In particular, the OEDOL algorithm is given by

$$\hat{\boldsymbol{x}}_{i,t} = \boldsymbol{A}_{i,t}\hat{\boldsymbol{x}}_{i,t-1} + \boldsymbol{B}_{i,t}\boldsymbol{y}_{i,t} + \boldsymbol{C}_{i,t}\boldsymbol{w}_{i,t-1}, \tag{4.24}$$

$$\hat{\boldsymbol{\Sigma}}_{i,t} = \boldsymbol{A}_{i,t}\hat{\boldsymbol{\Sigma}}_{i,t-1}, \tag{4.25}$$

where $\boldsymbol{w}_{i,t-1}$ is the innovation term extracted from the received information. $\boldsymbol{A}_{i,t}$, $\boldsymbol{B}_{i,t}$, and $\boldsymbol{C}_{i,t}$ are the corresponding weighting matrices defined as

$$\begin{aligned}\begin{bmatrix}\boldsymbol{B}_{i,t} & \boldsymbol{C}_{i,t}\end{bmatrix} = \; & \hat{\boldsymbol{\Sigma}}_{i,t-1}\tilde{\boldsymbol{H}}_{i,t-1}^{T}\times \\ & \left(\tilde{\boldsymbol{H}}_{i,t-1}\hat{\boldsymbol{\Sigma}}_{i,t-1}\tilde{\boldsymbol{H}}_{i,t-1}^{T} + \tilde{\boldsymbol{G}}_{i,t-1}\right)^{-1},\end{aligned} \tag{4.26}$$

$$\boldsymbol{A}_{i,t} = \boldsymbol{I} - \boldsymbol{B}_{i,t}\boldsymbol{H}_i - \boldsymbol{C}_{i,t}\bar{\boldsymbol{H}}_{i,t-1}, \tag{4.27}$$

where $\tilde{\boldsymbol{H}}_{i,t} = \mathrm{col}\{\boldsymbol{H}_i, \bar{\boldsymbol{H}}_{i,t}\}$, $\tilde{\boldsymbol{G}}_{i,t} = \mathrm{diag}\{\boldsymbol{\Sigma}_{n_i}, \bar{\boldsymbol{G}}_{i,t}\}$ and $\bar{\boldsymbol{G}}_{i,t} = \mathrm{diag}\{\boldsymbol{G}_{i,t}\}$ with

intermediate parameters $\bar{\boldsymbol{H}}_{i,t}$ and $\boldsymbol{G}_{i,t}$ evolving as

$$\bar{\boldsymbol{H}}_{i,t} = \begin{bmatrix} \boldsymbol{B}_{j_1,t}\boldsymbol{H}_{j_1} + \boldsymbol{C}_{j_1,t}\bar{\boldsymbol{H}}_{j_1,t-1} \\ \vdots \\ \boldsymbol{B}_{j_{\pi_i},t}\boldsymbol{H}_{j_{\pi_i}} + \boldsymbol{C}_{j_{\pi_i},t}\bar{\boldsymbol{H}}_{j_{\pi_i},t-1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}_{j_1,t}\bar{\boldsymbol{H}}^{(i)}_{j_1,t-1} \\ \vdots \\ \boldsymbol{C}_{j_{\pi_i},t}\bar{\boldsymbol{H}}^{(i)}_{j_{\pi_i},t-1} \end{bmatrix}, \tag{4.28}$$

$$\boldsymbol{G}_{i,t} = \begin{bmatrix} \boldsymbol{B}_{j_1,t}\boldsymbol{\Sigma}_{n_{j_1}}\boldsymbol{B}^T_{j_1,t} + \boldsymbol{C}_{j_1,t}\bar{\boldsymbol{G}}_{j_1,t-1}\boldsymbol{C}^T_{j_1,t} \\ \vdots \\ \boldsymbol{B}_{j_{\pi_i},t}\boldsymbol{\Sigma}_{n_{j_{\pi_i}}}\boldsymbol{B}^T_{j_{\pi_i},t} + \boldsymbol{C}_{j_{\pi_i},t}\bar{\boldsymbol{G}}_{j_{\pi_i},t-1}\boldsymbol{C}^T_{j_{\pi_i},t} \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}^{(i)}_{j_1,t}\boldsymbol{G}^{(i)}_{j_1,t-1}\left(\boldsymbol{C}^{(i)}_{j_1,t}\right)^T \\ \vdots \\ \boldsymbol{C}^{(i)}_{j_{\pi_i},t}\boldsymbol{G}^{(i)}_{j_{\pi_i},t-1}\left(\boldsymbol{C}^{(i)}_{j_{\pi_i},t}\right)^T \end{bmatrix}. \tag{4.29}$$

We initialize the parameters as $\bar{\boldsymbol{H}}_{j,\tau} = \boldsymbol{0}$ and $\boldsymbol{G}_{j,\tau} = \boldsymbol{0}$ for $\tau < 0$.

The recursion of the innovation parameter $\boldsymbol{w}_{i,t}$ is given by

$$\boldsymbol{w}_{i,t} = \bar{\boldsymbol{z}}_{i,t} - \boldsymbol{D}_{i,t}\hat{\boldsymbol{z}}_{i,t-1} + \boldsymbol{T}_{i,t}\boldsymbol{w}_{i,t-2}, \tag{4.30}$$

where

$$\boldsymbol{D}_{i,t} = \mathrm{col}\left\{\boldsymbol{C}^{(i)}_{j_1,t}, \cdots, \boldsymbol{C}^{(i)}_{j_{\pi_i},t}\right\} \tag{4.31}$$

and

$$\boldsymbol{T}_{i,t} \triangleq \begin{bmatrix} \boldsymbol{C}^{(i)}_{j_1,t}\boldsymbol{C}^{(j_1)}_{i,t-1} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{C}^{(i)}_{j_{\pi_i},t}\boldsymbol{C}^{(j_{\pi_i})}_{i,t-1} \end{bmatrix}. \tag{4.32}$$

Then, the agents disclose

$$\hat{\boldsymbol{z}}_{i,t} = \hat{\boldsymbol{x}}_{i,t} - \boldsymbol{A}_{i,t}\hat{\boldsymbol{x}}_{i,t-1}.$$

The detailed description of the algorithm is provided in Table 4.2.

Remark 4.2.1: In (4.24), the combination matrices $\boldsymbol{A}_{i,t}, \boldsymbol{B}_{i,t}$ and $\boldsymbol{C}_{i,t}$ are independent from the streaming data even though they are time-varying. This implies that they can be computed before-hand and in that case the computational complexity of the iterations is $O\left((\bar{\pi}q)^2\right)$ where $\bar{\pi}$ is the average of the cardinalities of the first-order neighborhoods across the network. Otherwise, the computational complexity of the algorithm is given by $O\left(m(\bar{\pi}q)^3\right)$. On the contrary, the computational complexity of the oracle algorithm is $O\left((mq)^3\right)$. Note

Table 4.2: The description of the OEDOL algorithm.

Algorithm 4.2: The OEDOL Algorithm

Initialization:

For $i = 1$ to m do

$\hat{\boldsymbol{x}}_{i,-1} = \bar{\boldsymbol{x}}$, $\hat{\boldsymbol{\Sigma}}_{i,-1} = \boldsymbol{\Sigma}_x$,

$\bar{\boldsymbol{H}}_{i,\tau} = \boldsymbol{0}$, $\boldsymbol{G}_{i,\tau} = \boldsymbol{0}$, and $\boldsymbol{w}_{i,t} = \boldsymbol{0}$ for $t < 0$

End for

Iterations:

Do for $t \geq 0$

For $i = 1$ to m do

Construction of Weights:

For $j = 1$ to m do

Calculate $\bar{\boldsymbol{H}}_{j,t}$ *and* $\boldsymbol{G}_{j,t}$ *by* (4.28) *and* (4.29).

Determine combination matrices via (4.26), (4.27), (4.31), *and* (4.32).

End for

Construct $\bar{\boldsymbol{z}}_{i,t-1}$ *through received* $\hat{\boldsymbol{z}}_{j,t-1}$ *for* $j \in N_i$

Extraction of Innovation:

$\boldsymbol{w}_{i,t-1} = \bar{\boldsymbol{z}}_{i,t-1} - \boldsymbol{D}_{i,t-1}\hat{\boldsymbol{z}}_{i,t-2} + \boldsymbol{T}_{i,t-1}\boldsymbol{w}_{i,t-3}$

Update:

$\hat{\boldsymbol{x}}_{i,t} = \boldsymbol{A}_{i,t}\hat{\boldsymbol{x}}_{i,t-1} + \boldsymbol{B}_{i,t}\boldsymbol{y}_{i,t} + \boldsymbol{C}_{i,t}\boldsymbol{w}_{i,t-1}$

Disclose $\hat{\boldsymbol{z}}_{i,t} = \hat{\boldsymbol{x}}_{i,t} - \boldsymbol{A}_{i,t}\hat{\boldsymbol{x}}_{i,t-1}$ *to the neighbors.*

End for

that over the tree we have $m - 1$ edges and correspondingly $\bar{\pi}$ is expected to be small. Hence, the optimal diffusion strategy over tree networks also reduces the computational complexity in general (in addition to the substantial reduction in communication). In Table 4.3, we tabulate the computational complexities of the introduced algorithms.

While constructing the spanning tree, we cancel certain communication links in order to avoid multi-path information propagation. However, we also observe that in a fully connected network we can achieve the performance of the ODOL algorithm, i.e., the distributed-MMSE performance, through the disclosure of

Table 4.3: A comparison of the computational complexities of the proposed algorithms.

Algorithm	Without weights	Pre-computed weights
ODOL algorithm	$O\left((mq)^3\right)$	$O\left((mq)^2\right)$
OEDOL algorithm	$O\left(m(\bar{\pi}q)^3\right)$	$O\left((\bar{\pi}q)^2\right)$

local observations. In particular, since all of the agents are connected, each agent can receive the observations across the network directly. Correspondingly, in a fully connected network, we can achieve identical performance with the ODOL algorithm only through the disclosure of the local estimates as stated in the following corollary formally.

Corollary 4.2.1: Consider the distributed estimation framework over a fully connected network. Then, the distributed-MMSE estimator (4.3) could also be obtained by (4.11), where agents disclose local estimates only.

Proof of Corollary 4.2.1: Over a fully connected network, $\kappa_i = 1$ and the MMSE estimator is given by

$$\hat{\boldsymbol{x}}_{i,t} = E\left[\underline{\mathcal{X}}\middle|\left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq t}, \left\{\underline{\mathcal{Y}}_{j,\tau} = \boldsymbol{y}_{j,\tau}\right\}_{j\in N_i}^{\tau\leq t-1}\right]. \tag{4.33}$$

By the Lemma 4.1, we can also obtain (4.33) by (4.13). Here, we have

$$\hat{Z}_{j,i,t} = Z_{j,t} \setminus \{Z_{i,t} \setminus \{\boldsymbol{y}_{i,t}\}\},$$

which yields

$$\begin{aligned}\hat{\boldsymbol{x}}_{i,t} = E\Bigg[\underline{\mathcal{X}}\Big|\left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq t}, \{\boldsymbol{\mathcal{Z}}_{j,\tau} = Z_{j,\tau}\}_{j\in N_i}^{\tau\leq t-1},\\ \Big\{\boldsymbol{\mathcal{Z}}_{j,t-1} \setminus \left\{\boldsymbol{\mathcal{Z}}_{i,t-1} \setminus \underline{\mathcal{Y}}_{i,t-1}\right\}\\ = Z_{j,t-1} \setminus \left\{Z_{i,t-1} \setminus \{\boldsymbol{y}_{i,t-1}\}\right\}\Big\}_{j\in N_i}\Bigg].\end{aligned} \tag{4.34}$$

Hence, by the Lemmas 3.1 and 3.2, (4.34) is also given by (4.11) and the proof is concluded. □

In the following, we enhance the learning rate further for an arbitrary network topology by exploiting the structure of the fully connected sub-networks that we call as "cell".

4.2.3 Tree Networks Involving Cell Structures

We define a "cell structure" as a sub-network in which all agents are connected to each other. Intuitively, considering a cell structure as "single" agent, we can involve the cell (i.e., all the agents in the cell) in the tree such that we can still achieve the distributed-MMSE performance through the disclosure of local estimates (although we may have loops in the cell). We list the features of the cell structures, e.g., seen in Fig. 4.3, as follows:

- Agents out of a cell can connect to at most one of the agents within that cell.
- A cell structure consists of at least 2 agents.
- An agent can belong to more than one cell.
- Two different agents cannot belong to more than one cell at the same time.
- All of the agents belong to at least a cell in a connected network.
- Each agent has also the knowledge of the cells of the other agents.
- Each agent labels its cells from its own and its first order neighbor's point of view. As an example, for the agent i, C_{i,i_1} denotes the cell involving both i and i_1. Note that if the same cell also includes i_2, $C_{i,i_1} = C_{i,i_2}$.

The following theorem shows that we can achieve the performance of the oracle algorithm over tree networks involving cell structures through the disclosure of the local estimates only.

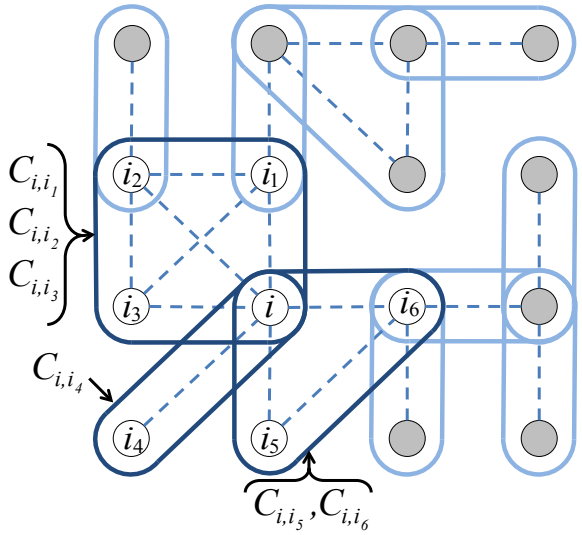


Figure 4.3: An example tree network involving cell structures.

Theorem 4.2.2: Consider the distributed estimation framework over tree networks involving cell structures. Then, the distributed-MMSE estimator (4.3) could also be obtained by (4.11).

Proof of Theorem 4.2.3: We point out that $\hat{Z}_{j,i,t}$ denotes the set of new information received by i over j at time t. Initially, we have $\hat{Z}_{j,i,0} = Z_{j,0} = \{\boldsymbol{y}_{j,0}\}$. By the Lemma 4.1, the MMSE estimator is also given by (4.13) over this network topology. Note that the information received by j at $t = 1$ is given by $Z_{j,1} = \left\{ \boldsymbol{y}_{j,1}, \left\{ \overbrace{\boldsymbol{y}_{k,0}}^{=\hat{Z}_{k,j,0}} \right\}_{k\in N_j}, Z_{j,0} \right\}$, which yields

$$\begin{aligned} Z_{j,i,1} &= Z_{j,1} \setminus \left\{ Z_{j,0} \cup \bigcup_{k\in C_{i,j}} \{\boldsymbol{y}_{k,0}\} \right\}, \\ &= Z_{j,1} \setminus \left\{ Z_{j,0} \cup \bigcup_{k\in C_{i,j}} \hat{Z}_{k,j,0} \right\} \end{aligned}$$

and $\hat{Z}_{j,j,t} = \varnothing$ by definition. Due to the cell structure, we have

$$\begin{aligned} \bigcup_{k\in C_{j,i}} \hat{Z}_{k,j,0} &= \hat{Z}_{i,j,0} \cup \bigcup_{k\in C_{i,j}\setminus j} \hat{Z}_{k,i,0}, \\ &= Z_{i,0} \cup \bigcup_{k\in C_{i,j}\setminus j} Z_{k,0}, \end{aligned}$$

which leads to

$$\hat{\boldsymbol{x}}_{i,2} = E\Bigg[\underline{\mathcal{X}}\Big|\left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{\tau\leq 2}, \{\boldsymbol{\mathcal{Z}}_{j,0} = Z_{j,0}\}_{j\in N_i}, \left\{\boldsymbol{\mathcal{Z}}_{j,1} \setminus \left\{\boldsymbol{\mathcal{Z}}_{j,0} \cup \boldsymbol{\mathcal{Z}}_{i,0} \cup \bigcup_{k\in C_{i,j}\setminus j} \boldsymbol{\mathcal{Z}}_{k,0}\right\}\right\} = Z_{j,1} \setminus \left\{Z_{j,0} \cup Z_{i,0} \cup \bigcup_{k\in C_{i,j}\setminus j} Z_{k,0}\right\}\Bigg\}_{j\in N_i}\Bigg].$$

We point out that $C_{i,j} \subset N_i$ and by the Lemma 3.2 we obtain (4.20). Correspondingly, for $t > 0$ we have

$$\hat{Z}_{j,i,t} = Z_{j,t} \setminus \left\{Z_{j,t-1} \cup \hat{Z}_{i,j,t-1} \cup \bigcup_{k\in C_{i,j}\setminus j} \hat{Z}_{k,i,t-1}\right\}$$

and $\hat{Z}_{j,i,t}$ is constructible by the sets $Z_{j,\tau}$ for $j \in N_i$ and $\tau \leq t$. Hence, for $t > 0$ we obtain (4.11) and the proof is concluded. □

In the sequel, we provide the sub-optimal extensions of the introduced algorithms for practical applications.

4.3 Sub-optimal Approaches

Minimization of the cost measure (4.2) optimally requires excessive computations. We aim to mitigate the problem sub-optimally yet in a computationally efficient approach while achieving comparable performance with the optimal case. As an example, we can approximate the cost measure (4.2) through time-windowing as follows

$$\tilde{\boldsymbol{x}}_{i,t} = \min_{\boldsymbol{x}} E\Bigg[\|\underline{\mathcal{X}} - \boldsymbol{x}\|^2\Big|\left\{\underline{\mathcal{Y}}_{i,\tau} = \boldsymbol{y}_{i,\tau}\right\}^{t-\kappa<\tau\leq t}, \left\{\underline{\mathcal{Y}}_{j,\tau} = \boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(1)}}^{t-\kappa<\tau\leq t-1}, \cdots, \left\{\underline{\mathcal{Y}}_{j,\tau} = \boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(\kappa_i)}}^{t-\kappa<\tau\leq t-\kappa_i}\Bigg].$$

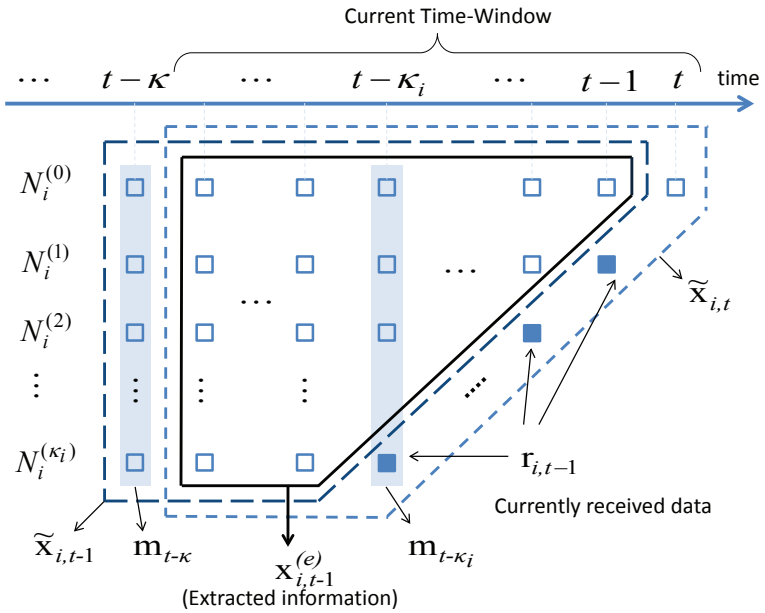


Figure 4.4: Information aggregation illustration. Small squares represent the information across the agents in the corresponding neighborhood and time.

In Fig. 4.4, we illustrate the time-windowing approach over the aggregated observations. We define a memory element $\boldsymbol{m}_t$ denoting the expectation of the zero-mean state conditioned on all observation data time-stamped at t, i.e., $\boldsymbol{m}_t \triangleq E\left[\underline{\mathcal{X}}\middle|\left\{\underline{\mathcal{Y}}_{i,t}=\boldsymbol{y}_{i,t}\right\}_{i\in N}\right]$. We have $\boldsymbol{m}_t = \boldsymbol{M}\boldsymbol{y}_t$ and $\boldsymbol{M} \triangleq \boldsymbol{\Sigma}_x\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{\Sigma}_x\boldsymbol{H}^T+\boldsymbol{\Sigma}_n\right)^{-1}$, where $\boldsymbol{H} \triangleq \text{col}\{\boldsymbol{H}_1,\cdots,\boldsymbol{H}_m\}$, $\boldsymbol{\Sigma}_n \triangleq \text{diag}\{\boldsymbol{\Sigma}_{n_1},\cdots,\boldsymbol{\Sigma}_{n_m}\}$ and $\boldsymbol{y}_t \triangleq \text{col}\{\boldsymbol{y}_{1,t},\cdots,\boldsymbol{y}_{m,t}\}$.

Correspondingly, $\boldsymbol{x}_{i,t-1}^{(e)}$ represents the extracted information from the estimate $\tilde{\boldsymbol{x}}_{i,t-1}$ via the memory element $\boldsymbol{m}_{t-\kappa}$. In particular, $\boldsymbol{x}_{i,t-1}^{(e)}$ is defined as follows

$$\boldsymbol{x}_{i,t-1}^{(e)} \triangleq E\Bigg[\underline{\mathcal{X}}\Big|\left\{\underline{\mathcal{Y}}_{i,\tau}=\boldsymbol{y}_{i,\tau}\right\}^{t-\kappa+1<\tau\leq t-1},\cdots,\left\{\underline{\mathcal{Y}}_{j,\tau}=\boldsymbol{y}_{j,\tau}\right\}_{j\in N_i^{(\kappa_i)}}^{t-\kappa+1<\tau\leq t-\kappa_i-1}\Bigg].$$

This yields $\boldsymbol{x}_{i,t-1}^{(e)} = \boldsymbol{N}_i\boldsymbol{y}_{i,t-1}^{(c)}$ where $\boldsymbol{y}_{i,t-1}^{(c)}$ is the vector collecting all observation data within that window (See Fig. 4.4) and $\boldsymbol{N}_i \triangleq \boldsymbol{\Sigma}_x\hat{\boldsymbol{H}}_i^T\left(\hat{\boldsymbol{H}}_i\boldsymbol{\Sigma}_x\hat{\boldsymbol{H}}_i^T+\hat{\boldsymbol{\Sigma}}_{n_i}\right)^{-1}$. We define $\hat{\boldsymbol{H}}_i \triangleq \text{col}\left\{\boldsymbol{H}_i,\boldsymbol{H}_i,\boldsymbol{H}_i^{(1)},\cdots,\boldsymbol{H}_i,\boldsymbol{H}_i^{(1)},\cdots,\boldsymbol{H}_i^{(\kappa_i)}\right\}$ and $\boldsymbol{H}_i^{(j)} \triangleq \text{col}\left\{\boldsymbol{H}_{j_1},\cdots,\boldsymbol{H}_{j_{\pi_i^{(j)}}}\right\}$. Correspondingly, $\hat{\boldsymbol{\Sigma}}_{n_i} \triangleq \text{diag}\left\{\boldsymbol{\Sigma}_{n_i},\boldsymbol{\Sigma}_{n_i},\boldsymbol{\Sigma}_{n_i}^{(1)},\cdots,\boldsymbol{\Sigma}_{n_i},\boldsymbol{\Sigma}_{n_i}^{(1)},\cdots,\boldsymbol{\Sigma}_{n_i}^{(\kappa_i)}\right\}$

and $\boldsymbol{\Sigma}_{n_i}^{(j)} \triangleq \text{diag}\left\{\boldsymbol{\Sigma}_{n_{j_1}}, \cdots, \boldsymbol{\Sigma}_{n_{j_{\pi_i^{(j)}}}}\right\}$. Then, after some algebra the estimate is given by $\tilde{\boldsymbol{x}}_{i,t-1} = \boldsymbol{K}_i \boldsymbol{m}_{t-\kappa} + \boldsymbol{L}_i \boldsymbol{x}_{i,t-1}^{(e)}$ where

$$\boldsymbol{K}_i \triangleq \left(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{H}\boldsymbol{N}_i\hat{\boldsymbol{H}}_i\right)^{-1} - \left(\boldsymbol{I} - \boldsymbol{N}_i\hat{\boldsymbol{H}}_i\boldsymbol{M}\boldsymbol{H}\right)^{-1}\boldsymbol{N}_i\hat{\boldsymbol{H}}_i,$$
$$\boldsymbol{L}_i \triangleq (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{H})\left(\boldsymbol{I} - \boldsymbol{N}_i\hat{\boldsymbol{H}}_i\boldsymbol{M}\boldsymbol{H}\right)^{-1}.$$

Hence, we obtain the sub-optimal distributed online learning (SDOL) algorithm as

$$\tilde{\boldsymbol{x}}_{i,t} = \boldsymbol{A}_i\tilde{\boldsymbol{x}}_{i,t-1} + \boldsymbol{B}_i\boldsymbol{y}_{i,t} + \boldsymbol{C}_i\boldsymbol{r}_{i,t-1} - \boldsymbol{D}_i\boldsymbol{y}_{t-\kappa}, \tag{4.35}$$

where $\boldsymbol{r}_{i,t-1}$ is a vector consisting of currently received observation data from the neighboring agents. The combination matrices defined as

$$\begin{bmatrix}\boldsymbol{B}_i & \boldsymbol{C}_i\end{bmatrix} = \boldsymbol{\Sigma}_{x,i}\bar{\boldsymbol{H}}_i^T\left(\bar{\boldsymbol{H}}_i\boldsymbol{\Sigma}_{x,i}\bar{\boldsymbol{H}}_i^T + \bar{\boldsymbol{\Sigma}}_{n_i}\right)^{-1}, \tag{4.36}$$
$$\boldsymbol{A}_i \triangleq \left(\boldsymbol{I} - \begin{bmatrix}\boldsymbol{B}_i & \boldsymbol{C}_i\end{bmatrix}\bar{\boldsymbol{H}}_i\right)\boldsymbol{L}_i^{-1}, \tag{4.37}$$
$$\boldsymbol{D}_i \triangleq \boldsymbol{A}_i\boldsymbol{K}_i\boldsymbol{M}, \tag{4.38}$$

where $\boldsymbol{\Sigma}_{x_i} \triangleq (\boldsymbol{I} - \boldsymbol{N}_i\hat{\boldsymbol{H}}_i)\boldsymbol{\Sigma}_x$. In Table 4.4, we tabulate the detailed description of the SDOL algorithm.

In the following we provide several remarks about the practical implementation of the SDOL algorithm.

Remark 4.3.1:

- We point out that in the update (4.35) the matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$, and $\boldsymbol{D}_i$ are independent from the data and they are time invariant. Hence, they can be pre-computed and installed onto the agents. Then, the SDOL algorithm basically takes the linear average of the previous estimate, the current measurement and received data, and observation data at time $t - \kappa$.
- Different from the conventional approaches, the SDOL algorithm requires to memorize previous observations. We note that $q < p/m$ in the model hence storing observation data individually rather than a linear combination, i.e., $\boldsymbol{m}_t$, might be more efficient in terms of memory usage.

Table 4.4: The description of the SDOL algorithm.

Algorithm 4.3: The SDOL algorithm

Initialization:

$\tilde{\boldsymbol{x}}_{i,-1} = \boldsymbol{0}$ for all i

$\boldsymbol{y}_{i,\tau} = \boldsymbol{0}$ for $\tau < 0$ and all i

Calculate combination matrices via (4.36), (4.37) *and* (4.38).

Update:

Do for $t \geq 0$

For $i = 1$ to m do

Construct $\boldsymbol{r}_{i,t-1}$ *through received* $\boldsymbol{r}_{j,t-1}$ *for* $j \in N_i^{(1)}$.

Store $\boldsymbol{r}_{i,t-1}$ *and* $\boldsymbol{y}_{i,t}$ *in the memory accordingly.*

$\tilde{\boldsymbol{x}}_{i,t} = \boldsymbol{A}_i\tilde{\boldsymbol{x}}_{i,t-1} + \boldsymbol{B}_i\boldsymbol{y}_{i,t} + \boldsymbol{C}_i\boldsymbol{r}_{i,t-1} - \boldsymbol{D}_i\boldsymbol{y}_{t-\kappa}$.

Erase $\boldsymbol{y}_{t-\kappa}$ *from the memory.*

Diffuse $\boldsymbol{r}_{i,t}$ *to the neighboring agents.*

End for

- The computational complexity of the DTA algorithm is $O(p^2 + 2pmq)$ (or say $O(p^2)$) and the algorithm requires disclosure of a data vector with $(m-1) \times q < p$ dimension.

- Finally, the steady-state MSE of the SDOL algorithm is given by

$$\text{MSE} = \text{Tr}\left\{\boldsymbol{A}_i\boldsymbol{L}_i\boldsymbol{\Sigma}_{x_i}\right\}.$$

In the sequel, we provide illustrative simulations showing the enhance tracking performance due to the proposed algorithm over several distributed learning examples.

4.4 Illustrative Examples

In this section, we examine the performance of the introduced strategies under different scenarios. Consider a network of $m = 20$ agents where each agent i

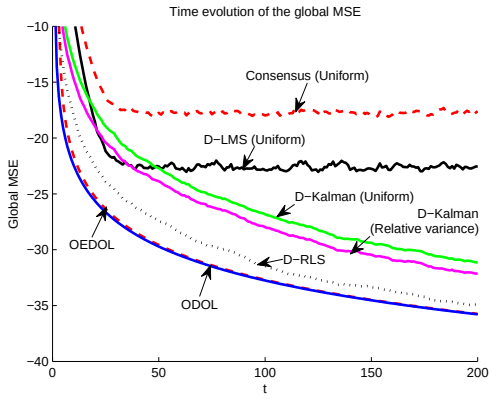


Figure 4.5: Comparison of the global MSE of the algorithms over an arbitrary network where the agents observe the same noise statistics.

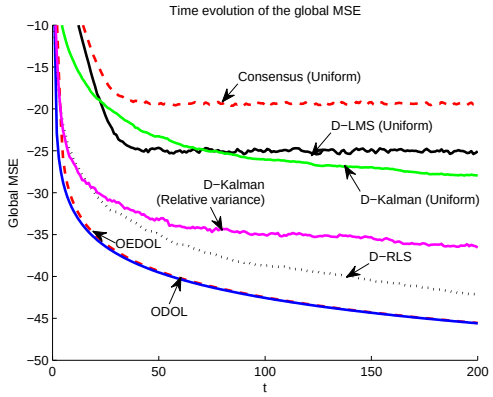


Figure 4.6: Comparison of the global MSE of the algorithms over an arbitrary network where the agents observe different noise statistics.

observes an underlying state $\boldsymbol{x} \in \mathbb{R}^p$ through $y_{i,t} = \boldsymbol{h}_i^T \boldsymbol{x} + n_{i,t}$. The unknown state $\boldsymbol{x}$ is a realization of standard-normal distribution. The observation noise is a zero-mean white Gaussian random process. For comparison, we choose $\boldsymbol{h}_i \in \mathbb{R}^p$ randomly from a standard-normal distribution. We define a global MSE performance measure over a network as $1/m \sum_{i=1}^{m} \text{MSE}(i)$, where $\text{MSE}(i)$ represents the MSE of the ith agent.

We compare the performance of the introduced algorithms with the diffusion implementation of Kalman updates (D-Kalman) [7], diffusion least mean square (D-LMS) [6], diffusion recursive least squares (D-RLS) [8], and consensus algorithms [42]. We utilize several combination rules. In the uniform combination

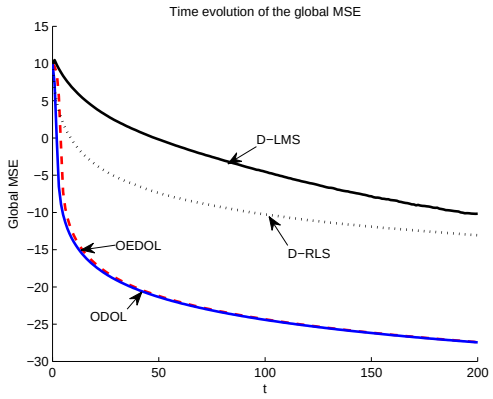


Figure 4.7: MSE performances of the introduced algorithms for $p = 10$.

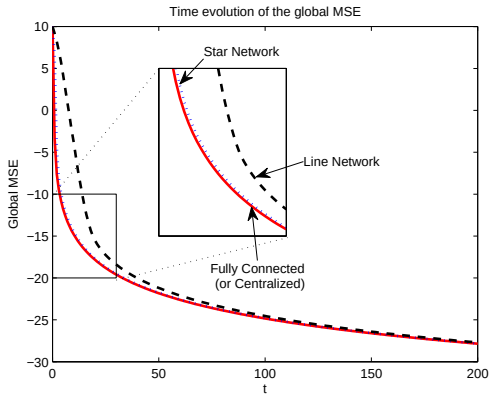


Figure 4.8: Comparison of the global MMSE curves over fully connected, star and line networks.

rule [43], the combination weights given by agent i for neighbor j are chosen as

$$\lambda_{i,j} = \begin{cases} 1/(\pi_i + 1) & \text{if } j \in N_i \cup \{i\} \\ 0 & \text{otherwise} \end{cases}$$

since N_i excludes i. Correspondingly, in the relative variance rule [25, 48], $\lambda_{i,j}$ is given by

$$\lambda_{i,j} = \begin{cases} \dfrac{\sigma_{n_j}^{-2}}{\sum_{k \in N_i \cup \{i\}} \sigma_{n_k}^{-2}} & \text{if } j \in N_i \cup \{i\} \\ 0 & \text{otherwise.} \end{cases}$$

In the D-LMS and consensus algorithms, we set $\mu = 0.1$. In the D-RLS algorithm, we use the Laplacian combination rule [84] for the incremental update and the Metropolis combination rule for the spatial update.

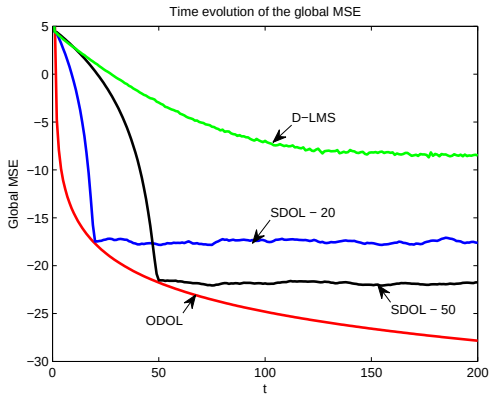


Figure 4.9: Influence of the time-windowing on the global MSE performance.

In Figs. 4.5 and 4.6, we compare the time evolution of the global MSE of all algorithms for space-invariant and space-variant noise statistics, respectively for $p = 1$. We set the standard deviations of the noise parameters from a folded standard normal distribution in the space-variant case. The network topology is chosen arbitrarily and we can readily construct a corresponding spanning tree of that network. As an example, we can construct the spanning tree based on the paths from the most connected agent i to the others, i.e., $i = \arg\max_j \pi_j$, and then we eliminate the multi-paths [79]. The relative-variance cooperation rule that considers the noise statistics to adjust the combination weights performs better than the uniform combination rule for space-variant noise profiles across the network. In Fig. 4.7, we compare the performance of the proposed algorithms for relatively large state vector, i.e., $p = 10$, and space variant noise profiles. In Figs. 4.5, 4.6 and 4.7 we observe that the proposed algorithms achieve superior performance compared to the other algorithms through the optimal weight selection for the MMSE performance. Additionally, the performance of the OEDOL and ODOL algorithms are close to each other even though the OEDOL algorithm operates over the corresponding spanning tree.

In order to examine the influence of the network topology on the MMSE performance, we provide Fig. 4.8. The fully connected network (or the centralized network) achieves the best possible connection among the agents whereas the line network yields the least possible connection. Over a star network, the furthest

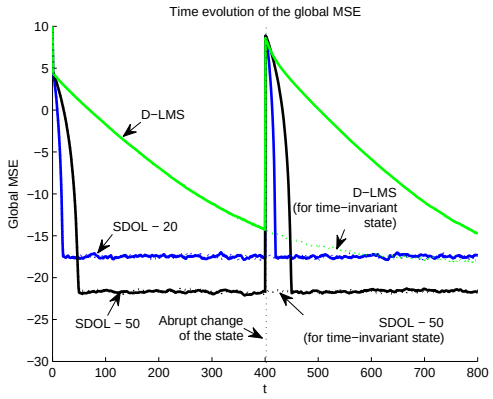


Figure 4.10: Tracking performance of the SDOL algorithms if the state changes abruptly.

distance between the agents is 2 and a single agent, i.e., the so-called pseudo-centralized agent, can access to the information of its neighbors directly. Notably, the star and line networks are the spanning tree of the fully connected network. We observe that the MMSE over the star network is close to the MMSE over the fully connected network. Additionally, the MMSE over even the line network converges to the MMSE over the fully connected network asymptotically.

For practical implementations, we propose the SDOL algorithm. In Fig. 4.9, we examine the influence of the time-windowing depth on the global MSE performance of the SDOL algorithm. We use the SDOL algorithms with time-windowing depths $\kappa = 20$ and $\kappa = 50$. The time evolution of the MSE performance differ from the MMSE behavior because the SDOL algorithms inherently assume that even at initial time, i.e., $t = 0$, agents have legitimate observations from $t = \kappa$ to $t = 0$. By excluding this data in time through sliding time-window, the SDOL algorithms eventually reach the best performance at the corresponding time-windowing depths and maintain this performance afterwards.

Finally, we evaluate the MSE performance of the SDOL algorithm for abruptly changing state. Assume that the state vector changes at time $t = 400$ abruptly. In particular, we have chosen a new realization from the standard normal distribution. In Fig. 4.10, we compare the SDOL algorithms with the D-LMS algorithm. Note that we set the step size of the D-LMS algorithm as $\mu = 0.025$

so that at steady state (if there were no state change) the D-LMS and SDOL with $\kappa = 20$ algorithms achieve the same MSE performance for fair tracking performance comparison. The sliding time-window provides relative robustness against abrupt state changes due to the exclusion of the effect of the previous observations. Hence, the SDOL algorithm can also provide enhance tracking performance in the practical applications.

Chapter 5

Application: Optimal Distributed Learning for Big Data

Along with the increased efficiency and performance improvements of the processing units, there is an enormous demand for the distributed or decentralized control of the autonomous data sources (ADS) [85–90]. In particular, an ADS can generate and process information locally irrespective of any centralized control unit. Through the distributed control of these sources, we can benefit from the natural advantages of cooperation, e.g., fast real-time response and reduced computational load. Hence, the distributed processing of the data has attracted substantial attention in diverse disciplines from services computing to machine learning fields [24, 27, 86–91]. Although there is an extensive literature on the distributed processing, we still have significant and yet unexplored challenges as follows:

Optimality. While processing the data, the ADS formulates an optimization problem based on a certain performance measure. However, the distributed processing strategies only achieve suboptimal performance with respect to that measure in general. Either they seek to achieve the optimal performance asymptotically or can they only aim to optimize an approximation of the performance

measure due to the difficulty of closed form analytical formulations or the excessive computational complexities which make those strategies not implementable for real life applications.

Efficiency. In the distributed control of the ADSs, the ADSs disclose information to the others so that they can enhance their learning performance cooperatively. This brings in an additional performance measure to consider, i.e., the communication load among the ADSs. Hence, the computational efficiency and communication load play crucial roles in the distributed control of the ADSs for real life applications [12, 47, 92, 93].

Online Operation. Since the data volume has grown up unprecedentedly in the era of Big Data, the ADSs should process the information in real time for streaming data rather than collecting all the excessive amount of measurements and processing them in a separate time scale [85, 94]. Different from the batch learning strategies in the online operation we expect the ADSs to operate optimally in a continuous manner rather than achieving an asymptotical convergence to the optimal performance in the limit [95].

Consequently, for Big Data applications, there is a demand for optimal and efficient distributed online learning strategy. In this paper, we introduce novel strategies addressing each of these challenges. Specifically, we consider the optimal state estimation over distributed data sources that has substantial amount of applications in the industry. As an example, consider the control of a collection of buses over an electric grid where the physical state is the voltage magnitude and angle, i.e., a smart grid application [32, 96]. The buses have generation, storage and demand response capabilities. They involve thousands of sensors and actuators for the energy management, which corresponds to excessive volume of data and requires the processing of them in an online manner. These buses are also connected via transmission lines to exchange information among each other and they seek an integrated energy management over the grid.

Contributions. We seek to formulate optimal and efficient distributed online

learning strategies for Big Data applications. To this end we propose a distributed control scheme for the ADSs. In particular, we design which information to disclose and how to utilize that information for the optimal online learning performance in the mean square error (MSE) sense. Over a network, the ADSs observe a noisy version of the underlying state and can share an information with certain ADSs constantly. For that model, we introduce a comprehensive cost measure considering the delay of the information transmission among the ADSs. Correspondingly, in the minimum MSE estimation framework, we formulate the oracle estimator achieving the optimal learning performance with prior information about the statistical profiles and the network topology. We derive a simple iterative oracle algorithm for the case of jointly Gaussian state and noise signals. The central limit theorem implies that the combination of sufficiently large number of realizations of independent random variables has approximately Gaussian distribution [97] hence such a framework widely used in various disciplines, e.g., statistical learning theory [77, 98]. We point out that the algorithms achieve the linear minimum MSE performance for other cases, i.e., if the noise and state statistics are not Gaussian. For certain network structures, we also propose novel algorithms achieving consistently the oracle performance with the disclosure of only a single scalar among the neighboring sources. Note that for an arbitrary network topology, we can construct such network connections by eliminating certain communication links. Additionally, we provide numerical examples demonstrating the significant gains due to the proposed strategies with respect to the state-of-the-art strategies.

5.1 ODOL Algorithm for Big Data

Consider a distributed network of m ADSs. We assume that the graph $G = (V, E)$ corresponding to the network is *connected*. V denotes the set of vertices and E is the set of the edges. The network is partially connected such that certain unordered pairs are not in E, i.e., $(i, j) \notin E$. For each ADS-i, we denote the kth order neighborhood $N_i^{(k)}$ as the set of ADSs whose information could be received

at least after k delays, i.e.,

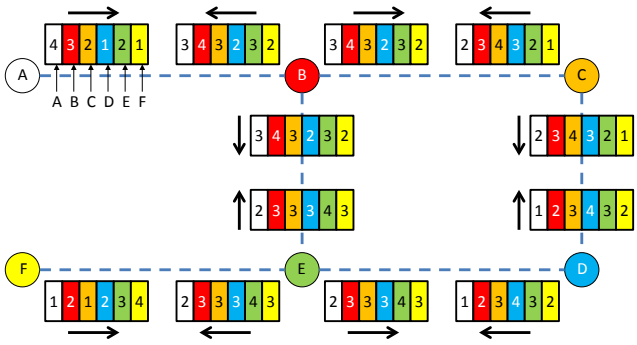
$$N_i^{(k)} \triangleq \left\{ j_1, \cdots, j_{\pi_i^{(k)}} \right\}$$

and $\pi_i^{(k)} = \left| N_i^{(k)} \right|$ is the cardinality of $N_i^{(k)}$. We also assume that $N_i^{(0)} = \{i\}$ and $N_i^{(k)} = \varnothing$ for $k < 0$. Here, the ADSs observe a noisy version of a time-invariant and unknown state $x \in \mathbb{R}$ which is a realization of the Gaussian random variable $\mathcal{X}$ with mean $\bar{x}$ and variance σ_x^2. Each ADS-i observes the state as

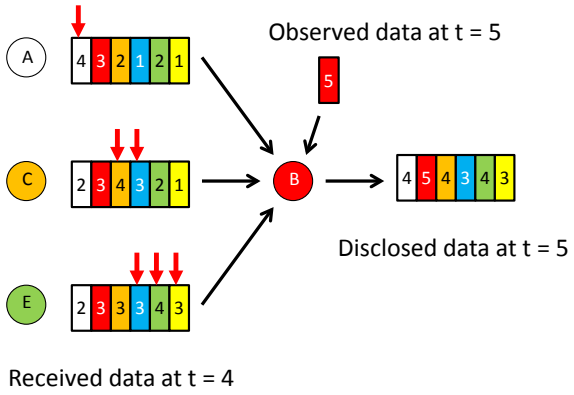
$$y_{i,t} = x + n_{i,t},$$

where $n_{i,t}$ is a realization of a zero-mean white Gaussian noise process $\mathcal{N}_{i,t}$ with variance $\sigma_{n_i}^2$ and correspondingly the observation $y_{i,t}$ is a realization of the random process $\mathcal{Y}_{i,t} = \mathcal{X} + \mathcal{N}_{i,t}$. The noise is also independent from the state and the other noise parameters. Note that the derivations extend to vector case, however, we use the scalar model for notational simplicity. We consider that the variance of the noise signals are known, which can also be readily estimated from the data [5]. Let's assume that at each instant, the ADSs observe the underlying state through $y_{i,t}$, diffuse information to the neighboring ADSs and receive information from them. Naturally, the ADSs can learn the true state, under certain regularity conditions [5], irrespective of cooperation among them, provided that their own measurements are not biased and sufficient to estimate the true state in time. However, through the cooperation of ADSs, we can increase the learning rate significantly [24]. To this end, we aim to find the optimal learning strategy over distributed networks in the MSE sense.

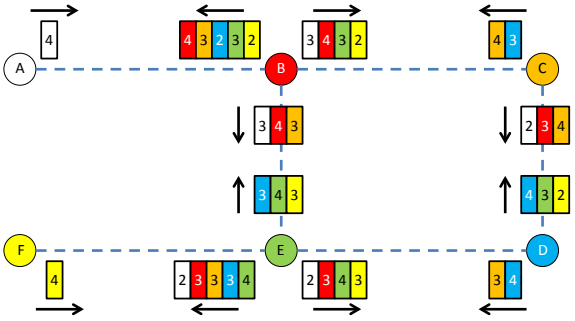
As the oracle approach, we first study the case when there is no limitation on the amount of the disclosed information. ADSs disclose their own measurements and transmit the received information from the neighboring ADSs with time stamps, i.e., the information is ordered (See Fig. 5.1). Through this implementation, each ADS can obtain the measurements of the other ADSs separately in a connected network. We point out that the information on the non-neighboring ADSs could only be received after certain delay. As an example, the disclosed information of $j \in N_i^{(2)}$ reaches to i by passing through two communication links. In particular, this case assumes that each ADS has access to full information from the other ADSs, albeit with certain delay, and corresponds to the "optimal case".



(a) Data exchange at $t = 4$.



(b) Operation over ADS-B.



(c) Removing redundant data exchange

Figure 5.1: Time stamped data exchange over the network of ADSs.

At time t, all the information that the ith ADS has access to is given by

$$\left\{ \{y_{i,\tau}\}^{\tau \leq t}, \{y_{j,\tau}\}_{j \in N_i^{(1)}}^{\tau \leq t-1}, \cdots, \{y_{j,\tau}\}_{j \in N_i^{(\kappa_i)}}^{\tau \leq t-\kappa_i} \right\},$$

where κ_i is the least delay for the information received from the furthest ADS[1]. Here, the collection $\{y_{j,\tau}\}_{j \in N_i^{(k)}}^{\tau \leq t-k}$ is the set of observations from the kth order neighborhood $N_i^{(k)}$. Particularly, the collection is explicitly defined as

$$\{y_{j,\tau}\}_{j \in N_i^{(k)}}^{\tau \leq t-k} \triangleq \Big\{ y_{j_1,t-k}, ..., y_{j_1,0}, \\ ..., y_{j_{\pi_i^{(k)}},t-k}, ..., y_{j_{\pi_i^{(k)}},0} \Big\}. \tag{5.1}$$

Then, the best possible estimate in the MSE sense for each ADS-i is given by the conditional estimate of x given all the accessed information:

$$\hat{x}_{i,t} = E\Big[\mathcal{X} \Big| \{\mathcal{Y}_{i,\tau} = y_{i,\tau}\}^{\tau \leq t}, \{\mathcal{Y}_{j,\tau} = y_{j,\tau}\}_{j \in N_i^{(1)}}^{\tau \leq t-1}, \cdots, \\ \{\mathcal{Y}_{j,\tau} = y_{j,\tau}\}_{j \in N_i^{(\kappa_i)}}^{\tau \leq t-\kappa_i} \Big] \tag{5.2}$$

and we can extend the ODOL algorithm to calculate (5.2) in an iterative way.

Remark 5.1.1: The ODOL algorithm configuration requires the diffusion of m separate information for each ADS (See Fig. 5.1a). However, the communication load is crucial for the applicability of the distributed processing algorithms [12,47]. As seen in Fig. 5.1b, the ADS-B receives multiple versions of the same data with different time stamps, however, uses only the most current one and transmit this data to the neighboring ADSs. This implies that through directive data disclosure we can reduce the redundant communication load as seen in Fig. 5.1c. Furthermore, due to the multi-paths, the same data reaches to an ADS at different instances. By eliminating the multi-paths we can avoid that redundancy and the ADSs can receive each data only once. Additionally, since the state and noise parameters are jointly Gaussian, each ADS linearly combine the received measurements in order to obtain the minimum mean square error (MMSE) estimator. Based on that if there is no multi-path, each ADS could also disclose only that corresponding linear combination, which is a single scalar, instead of transmitting each measurement separately with time stamps. Consequently, we can

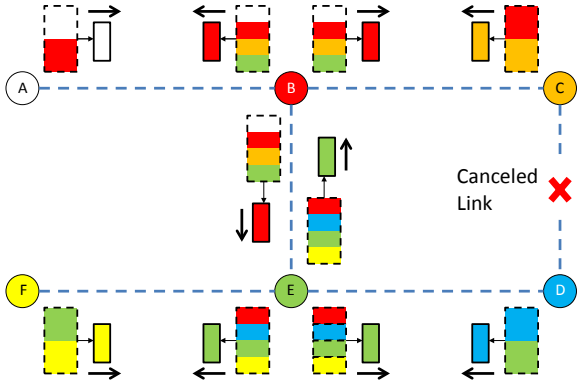
[1]For notational simplicity, we denote $N_i \triangleq N_i^{(1)}$ and $\pi_i \triangleq \pi_i^{(1)}$.

reduce the communication load tremendously. However, instead of the directive diffusion, in the sequel, we demonstrate that the same estimation performance can also be achieved by indirected disclosure of a single scalar.

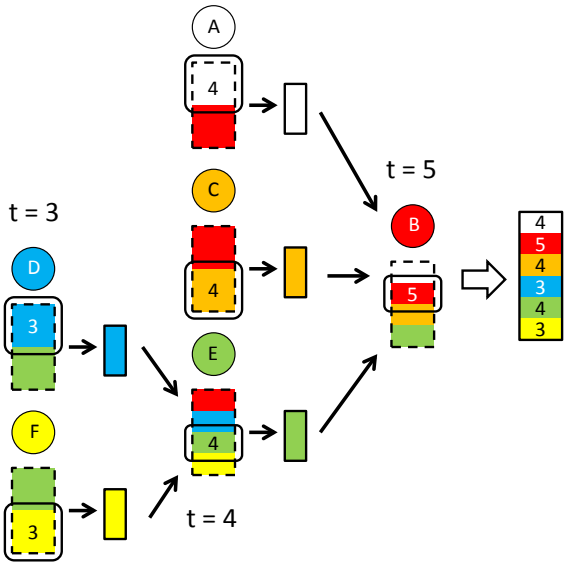
5.2 OEDOL Algorithm for Big Data

We seek to find a configuration in which through the disclosure of a single scalar we can still achieve the optimal learning performance. Consider the scenario given in Fig. 5.1c. The ADS-B receives time stamped information from the neighboring ADSs, i.e., ADS-A, ADS-C and ADS-E. The ADS-B combines received data linearly to learn the underlying state. Hence, ADS-E can simply disclose that linear configuration (which is a single scalar) instead of time stamped information disclosure. However, the measurement of ADS-D reaches ADS-B over two separate path. The ADS-B cannot combine the measurements optimally if both ADS-C and ADS-E disclose the corresponding linear combinations. Importantly, we can overcome this issue by cancelling multi-paths. We also seek indirected disclosure of a single scalar. Such a strategy is crucial for the networks with wireless communication links [24].

A network has a "tree structure" if its corresponding graph is a *tree*, i.e., connected and undirected without any multi-paths [99]. We remark that for an arbitrary network topology we can also construct the spanning tree of the network and eliminate the cycles. In the literature, there exists numerous distributed algorithms for minimum spanning tree construction [79]. In Fig. 5.2a, we construct a corresponding spanning tree by cancelling the communication link between ADS-C and ADS-D. Under that configuration, information disseminates only over a single path and each ADS can learn the underlying state optimally by combining simply the received linear combinations as seen in Fig. 5.2b. In the Theorem 4.2.1, we show that over tree networks we can achieve the optimal performance through the indirected disclosure of the local estimated parameter only.



(a) Single scalar data exchange.



(b) Operation over ADS-B.

Figure 5.2: Indirected single scalar data exchange over the network of ADSs.

We remark that the MMSE estimator $\hat{x}_{i,t}$ linearly depends on $\hat{x}_{i,t-1}$, which has been diffused previously. In order to extract the new information, we need to eliminate the previously received information at each instant on the neighboring ADSs. This brings in additional computational complexity. On the contrary, we can just diffuse the new information, i.e., $\hat{z}_{i,t}$, after eliminating all the previously disclosed information. Since we are conditioning on the linear combinations of the conditioned variables without effecting their spanned space, i.e., $\hat{z}_{i,t}$ is computable from $\hat{x}_{i,\tau}$ for $\tau \leq t$ and vice versa, we can still achieve the oracle performance by reduced computational load, yet. To this end, we can extend the OEDOL algorithm straight-forwardly.

Remark 3.1: The coefficients in the update are independent from the streaming data even though they are time-varying. This implies that they can be computed before-hand and in that case the computational complexity of the iterations is $O(\bar{\pi}^2)$ where $\bar{\pi}$ is the average of the cardinalities of the first-order neighborhoods across the network. Otherwise, the computational complexity of the OEDOL algorithm is given by $O(m\bar{\pi}^2)$. On the contrary, the computational complexity of the ODOL algorithm is $O(m^3)$. Note that over the tree we have $m-1$ edges and correspondingly $\bar{\pi}$ is expected to be small. Hence, the OEDOL algorithm also reduces the computational complexity in general (in addition to the substantial reduction in communication). Additionally, in Section IV, we observe that the oracle algorithm over arbitrary networks and the corresponding spanning tree networks demonstrate similar MSE performance while the spanning tree structure requires significantly reduced communication load due to the eliminated connections.

5.3 Experiments

In this section, we examine the MSE performance of the introduced strategies against the state-of-the-art strategies. Consider a large network of $m = 100$ ADSs where each ADS-i observes an underlying state x through $y_{i,t} = x + n_{i,t}$. The unknown state x is a realization of a zero-mean unit variance Gaussian random

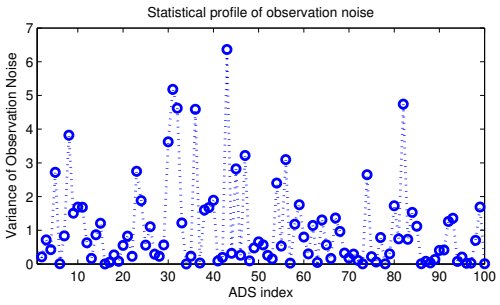


Figure 5.3: Network statistics.

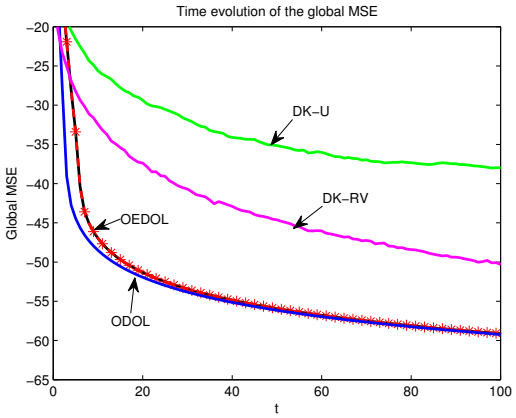


Figure 5.4: Comparison of the global MSE of the algorithms over an arbitrary network of ADSs observing different noise statistics.

variable. The observation noise is a zero-mean white Gaussian random process with randomly chosen standard deviation σ_{n_i} from a folded unit-variance normal distribution (See Fig. 5.3). We define a global MSE performance measure over the network as the uniform average of the MSE of the ADSs. We choose the connections of the network arbitrarily such that the network has 315 communication links. Correspondingly, we construct the spanning tree of that network. Based on the paths from the most connected ADS-i to the others, i.e., $i = \arg\max_j \pi_j$, and then we eliminate the multi-paths [79]. We point out that the spanning tree network has 99 edges. This implies significantly reduced communication demand over the spanning tree network.

In Fig. 5.4, we compare the global MSE performance of the proposed algorithms with the diffusion implementation of Kalman updates [91]. We consider that each ADS performs a discrete Kalman update, diffuses the estimator to the

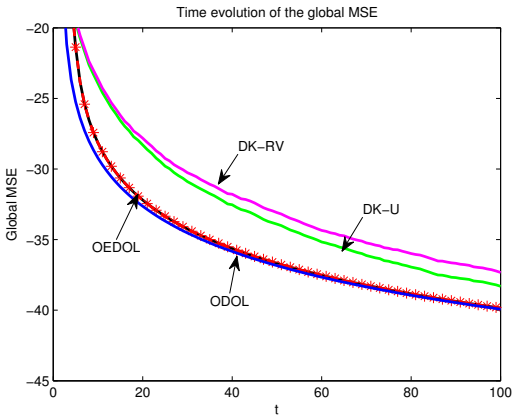


Figure 5.5: Comparison of the global MSE of the algorithms over an arbitrary network of ADSs observing the same statistics.

neighbors, and combines the received estimators linearly through certain combination rules. For the same arbitrary network, we compare the time evolution of the global MSE of all the algorithms in Figs. 5.4 and 5.5. The relative-variance cooperation rule that considers the noise statistics to adjust the combination weights performs better than the uniform combination rule due to space-variant noise profiles across the network. Additionally, we examine the performance of the algorithms for the space-invariant noise statistics. To this end, we use unit-variance noise signals at each ADS. In Fig. 5.5, we observe that different from the configuration of space-variant noise statistics the uniform cooperation rule outperforms the relative-variance rule. However, in both cases, the OEDOL outperforms the other algorithms as expected. Additionally, the ODOL and OEDOL demonstrate similar learning performance while OEDOL reduces the communication load to 1%, i.e., from 100 time-stamped scalar to a single scalar.

Chapter 6

Conclusion

In the diffusion based distributed estimation strategies, the communication load increases far more in the large networks or highly connected network of nodes. Hence, the compressive diffusion approach plays an essential role in achieving comparable convergence performance to the full diffusion configurations while reducing the communication load tremendously. We provide a complete performance analysis for the compressive diffusion strategies. We analyze the mean-square convergence, the steady-state behavior and the tracking performance of the scalar and single-bit diffusion approaches. The numerical examples show that the theoretical analysis model the simulated results accurately. Additionally, we introduce the confidence parameter concept, which adds one more freedom of dimension to the combination rule in order to improve the convergence performance. When we adapt the confidence parameter using the well-known adaptive mixture algorithms, we observe enormous enhancement in the convergence performance of the compressive diffusion strategies even for relatively long filter lengths.

In addition to that the distributed algorithms have attracted significant attention due to their wide spread applicability to highly complex structures from biological systems to social and economical networks. However, there are still challenges for disclosure and utilization of information among agents. We introduce the ODOL algorithm achieving the MMSE performance for Gaussian

state and noise statistics and the LMMSE performance for arbitrary statistical profiles. We propose a consensus-like iterative algorithm for streaming data over an arbitrary network topology via the aggregation of information at each agent rather than propagation of information through the disclosure of local estimates. Importantly, the disclosure of the local estimate is sufficient only over the certain network topologies, e.g., over the introduced tree network involving cell structures. Moreover, the aggregation of information based approach can require excessive communication load. In order to reduce the communication load, we exploit the network structures. We introduce the OEDOL algorithm that achieves the MMSE performance through the diffusion of information. We also observe that the MSE performance of the ODOL and OEDOL algorithms are close to each other over even a highly connected arbitrary network while the OEDOL provides reduced communication load in general. Finally, we introduce time-windowing approach for practical applications due to the reduced complexity. The SDOL algorithm possesses consensus-like iterations with time-invariant combination weights that should be calculated only once and the SDOL algorithm provides enhanced tracking performance for changing state due to the sliding of time-window.

Bibliography

[1] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, 2013.

[2] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, 2002.

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

[4] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proc. Int. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, vol. 4, pp. 2033–2036 vol.4, 2001.

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

[6] 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.

[7] 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, 2010.

[8] 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.

[9] F. S. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2069–2084, 2010.

[10] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *Signal Processing, IEEE Transactions on*, vol. 60, no. 12, pp. 6217–6234, 2012.

[11] X. Zhao, S.-Y. Tu, and A. H. Sayed, "Diffusion adaptation over networks under imperfect information exchange and non-stationary data," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3460–3475, 2012.

[12] M. O. Sayin and S. S. Kozat, "Single bit and reduced dimension diffusion strategies over distributed networks," *IEEE Signal Processing Letters*, vol. 20, no. 10, pp. 976–979, 2013.

[13] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[14] R. G. Baraniuk, V. Cevher, and M. B. Wakin, "Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 959–971, 2010.

[15] R. Arablouei, S. Werne, and K. Dogancay, "Partial-diffusion recursive least-squares estimation over adaptive networks," in *IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 89–92, 2013.

[16] R. Arablouei, S. Werner, Y. F. Huang, and K. Dogancay, "Distributed least mean square estimation with partial diffusion," *IEEE Transactions on Signal Processing*, vol. 62, no. 2, pp. 472–483, 2014.

[17] R. Arablouei, S. Werner, Y. F. Huang, and K. Dogancay, "Adaptive distributed estimation based on recursive least-square and partial diffusion,"

IEEE Transactions on Signal Processing, vol. 62, no. 14, pp. 3510–3522, 2014.

[18] S. Chouvardas, K. Slavakis, and S. Theodoridis, "Trading off complexity with communication costs in distributed adaptive learning via Krylov subspaces for dimensionality reduction," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 257–273, 2013.

[19] S. Xie and H. Li, "Distributed LMS estimation over networks with quantised communications," *International Journal of Control*, vol. 86, no. 3, pp. 478–492, 2013.

[20] A. Ribeiro, G. B. Giannakis, and S. I. Roumeliotis, "SOI-KF: Distributed Kalman filtering with low-cost communications using the sign of innovations," *IEEE Transactions on Signal Processing*, vol. 54, no. 12, pp. 4782–4795, 2006.

[21] H. Sayyadi and M. R. Doostmohammadian, "Finite-time consensus in directed switching network topologies and time-delayed communications," *Scientia Iranica*, vol. 18, pp. 75–85, February 2011.

[22] J. N. Tsitsiklis, *Problems in decentralized decision making and computation.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[23] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. AC-31, no. 9, pp. 803–812, 1986.

[24] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, 2013.

[25] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, 2010.

[26] S. Kar, J. M. F. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," *IEEE Transactions on Information Theory*, vol. 58, no. 6, pp. 3575–3605, 2012.

[27] S. Shahrampour, A. Rakhlin, and A. Jadbabaie, "Online learning of dynamic parameters in social networks," in *Neural Information Processing Systems*, pp. 2013–2021, 2013.

[28] A. Jadbabaie, P. Molavi, A. Sandroni, and A. Tahbaz-Salehi, "Non-Bayesian social learning," *Games and Economic Behavior*, vol. 76, no. 1, pp. 210–225, 2012.

[29] D. Acemoglu and A. Ozdaglar, "Dynamic games and applications," *Opinion dynamics and learning in social networks*, vol. 1, no. 1, pp. 3–49, 2011.

[30] S. Camazine, J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*. Princeton, N.J.: Princeton University Press, 2003.

[31] M. Jackson, *Social and Economic Networks*. Princeton, N.J.: Princeton University Press, 2008.

[32] S. Kar, G. Hug, J. Mohammadi, and J. M. F. Moura, "Distributed state estimation and energy management in smart grids: A consensus + innovations approach," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 6, pp. 1022–1038, 2014.

[33] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[34] I. Schizas, G. Mateos, and G. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365–2382, 2009.

[35] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[36] D. Shah, "Gossip algorithms," in *Foundations and Trends in Networking*, Boston, M.A.: Now Publishers, 2009.

[37] A. Jadbabaie, J. Lin, 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, 2003.

[38] A. Nedic and A. Ozdaglar, "Cooperative distributed multi-agent optimization," in *Convex Optimization in Signal Processing and Communications* (Y. Eldar and D. Palomar, eds.), pp. 340–386, Cambridge, U.K.: Cambridge University Press, 2009.

[39] J. W. Lee, S. E. Kim, W. J. Song, and A. H. Sayed, "Spatio-temporal diffusion strategies for estimation and detection over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4017–4034, 2012.

[40] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012.

[41] J. Chen and A. H. Sayed, "Distributed Pareto optimization via diffusion strategies," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 205–220, 2013.

[42] S. Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, 2012.

[43] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus and flocking," in *Proc. Joint 44th IEEE Conf. Decision Control Eur. Control Conf. (CDC-ECC), Seville, Spain, Dec.*, pp. 2996–3000, 2005.

[44] D. S. Scherber and H. C. Papadopoulos, "Locally constructed algorithms for distributed computations in ad-hoc networks," in *Proceedings of Information Processing Sensor Networks (IPSN)*, pp. 11–19, 2004.

[45] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[46] 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, 2010.

[47] M. O. Sayin and S. S. Kozat, "Compressive diffusion strategies over distributed networks for reduced communication load," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5308–5323, 2014.

[48] S.-Y. Tu and A. H. Sayed, "Optimal combination rules for adaptation and learning over networks," in *Proceedings of 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 317–320, 2011.

[49] C. K. Yu and A. H. Sayed, "A strategy for adjusting combination weights over adaptive networks," in *Proceedings of 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4579–4538, 2013.

[50] J. Fernandez-Bes, J. Arenas-Garcia, and A. H. Sayed, "Adjustment of combination weights over adaptive diffusion networks," in *Proceedings of 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6409–6413, 2014.

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

[52] T. C. Aysal, B. N. Oreshkin, and M. J. Coates, "Accelerated distributed average consensus via localized node state prediction," *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1563–1576, 2009.

[53] C. C. Moallemi and B. V. Roy, "Consensus propagation," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4753–4766, 2006.

[54] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.

[55] 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, 2010.

[56] E. Barker and J. Kelsey, "Recommendation for random number generation using deterministic random bit generators," *NIST SP800-90A*, 2012.

[57] J. Joutsensalo and T. Ristaniemi, "Synchronization by pilot signal," in *Proc. Int. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, pp. 2663–2666, 1999.

[58] F. T. Castoldi and M. L. R. Campos, "Application of a minimum-disturbance description to constrained adaptive filters," *IEEE Signal Processing Letters*, vol. 20, no. 12, pp. 1215–1218, 2013.

[59] M. A. Donmez, H. A. Inan, and S. S. Kozat, "Adaptive mixture methods based on Bregman divergences," *Digital Signal Processing*, vol. 23, pp. 86–97, 2013.

[60] A. H. Sayed, T. Y. Al-Naffouri, and V. H. Nascimento, "Energy conservation in adaptive filtering," in *Nonlinear Signal and Image Processing: Theory, Methods, and Applications* (K. E. Barner and G. R. Arce, eds.), Florida: CRC Press, 2003.

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

[62] T. Y. Al-Naffouri and A. H. Sayed, "Transient analysis of data-normalized adaptive filters," *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 639–652, 2003.

[63] R. Price, "A useful theorem for nonlinear devices having gaussian inputs," *IEEE Transactions on Information Theory*, vol. 4, no. 2, pp. 69–72, 1958.

[64] E. McMahon, "An extension of price's theorem (corresp.)," *IEEE Transactions on Information Theory*, vol. 10, no. 2, pp. 168–168, 1964.

[65] T. Koh and E. J. Powers, "Efficient methods of estimate correlation functions of Gaussian processes and their performance analysis," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 4, pp. 1032–1035, 1985.

[66] X. Zhao and A. H. Sayed, "Combination weights for diffusion strategies with imperfect information exchange," in *IEEE International Conference on Communications*, pp. 398–402, 2012.

[67] S.-Y. Tu and A. H. Sayed, "Adaptive networks with noisy links," in *IEEE Global Telecommunications Conference*, pp. 1–5, 2011.

[68] J. Arenas-Garcia, V. Gomez-Verdejo, and A. R. Figueiras-Vidal, "New algorithms for improved adaptive convex combination of LMS transversal filters," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 6, pp. 2239–2249, 2005.

[69] J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Transactions on Signal Processing*, vol. 54, no. 3, pp. 1078–1090, 2006.

[70] M. T. M. Silva and V. H. Nascimento, "Improving the tracking capability of adaptive filters via convex combination," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3137–3149, 2008.

[71] S. S. Kozat, A. T. Erdogan, A. C. Singer, and A. H. Sayed, "Steady state MSE performance analysis of mixture approaches to adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 58, pp. 4050–4063, August 2010.

[72] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of back propagation learning," in *From Natural to Artificial Neural Computation* (J. Mira and F. Sandoval, eds.), Springer Berlin Heidelberg, 1995.

[73] J. J. Xiao, A. Riberio, Z. Q. Luo, and G. B. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 27–41, 2006.

[74] F. S. Cattivelli and A. H. Sayed, "Diffusion detection over adaptive networks using diffusion adaptation," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 1917–1932, 2011.

[75] M. O. Sayin and S. S. Kozat, "Single-bit and reduced dimension diffusion strategies over distributed networks," *IEEE Signal Processing Letters*, vol. 20, no. 10, pp. 976–979, 2013.

[76] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Springer, 1992.

[77] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1979.

[78] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Reading, MA: Addison-Wesley, 1990.

[79] B. Y. Wu and K.-M. Chao, *Spanning Trees and Optimization Problems (Discrete Mathematics and Its Applications)*. New York: CRC Press, 2004.

[80] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 1, pp. 66–71, 1983.

[81] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. Philadelphia, PA: SIAM, 2000.

[82] M. Elkin, "Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem," in *Proceedings of 36th ACM Symposium on Theory of Computing (STOC)*, pp. 331–340, 2004.

[83] M. Khan, G. Pandurangan, and V. S. A. Kumar, "Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 1, pp. 124–139, 2009.

[84] R. Olfati-Saber and R. M. Murray, "Ieee transactions on automatic control," *Consensus problems in networks of agents with switching topology and time-delays*, vol. 49, no. 9, pp. 1520–1533, 2004.

[85] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[86] B. S. Vidyalakschimi, R. K. Wong, and C.-H. Chi, "Decentralized trust driven access control for mobile content sharing," in *Proceedings of IEEE International Congress on Big Data*, pp. 239–246, 2013.

[87] Q. He, J. Yan, Y. Yang, R. Kowalczyk, and H. Jin, "A decentralized service discovery approach on peer-to-peer networks," *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 64–75, 2013.

[88] S.-P. L. ad M.-H. Wong, "Data allocation in scalable dsitrbuted database systems based on time series forecasting," in *Proceedings of IEEE International Congress on Big Data*, pp. 17–24, 2013.

[89] J. Li, H. Wang, S. U. Khan, Q. Li, and A. Y. Zomaya, "A fully distributed scheme for discovery of semantic relationships," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 457–469, 2013.

[90] A. Mohaisen, H. Tran, A. Chandra, and Y. Kim, "Trustworthy distributed computing on social networks," *IEEE Transactions on Services Computing*, vol. 7, no. 3, pp. 333–345, 2014.

[91] F. S. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2069–2084, 2010.

[92] A. Barker, J. B. Weissman, and J. I. van Hemert, "Reducing data transfer in service-oriented architectures: The circulate approach," *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 437–449, 2012.

[93] B. N. Thyamagondlu, V. W. Chu, and R. K. Wong, "A bandwidth-conscious caching scheme for mobile devices," in *Proceedings of IEEE International Congress on Big Data*, pp. 78–85, 2013.

[94] L. Song, C. Tekin, and M. van der Schaar, "Online learning in large-scale contextual recommender systems," *IEEE Transactions on Services Computing*, vol. PP, no. 99, 2015.

[95] Y. Achbany, I. J. Jureta, S. Faulkner, and F. Fouss, "Continually learning optimal allocations of services to tasks," *IEEE Transactions on Services Computing*, vol. 1, no. 3, pp. 141–154, 2008.

[96] Y.-F. Huang, S. Werner, J. Huang, N. Kashyap, and V. Gupta, "State estimation in electric power grids: Meeting new challenges presented by the requirements of the future grid," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 33–43, 2012.

[97] J. A. Rice, *Mathematical Statistics and Data Analysis.* Belmont, CA: Duxbury Press., 2007.

[98] H. V. Poor, *An Introduction to Signal Detection and Estimation.* New York: Springer-Verlag, 1994.

[99] F. R. K. Chung, *Spectral Graph Theory.* Providence, RI: American Mathematical Society, 1997.