DISSERTATION

Distributed Averaging in Wireless Sensor Networks

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften

unter der Leitung von
Ao. Univ.-Prof. Dr. G. Matz
Institute of Telecommunications

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik

von
Valentin Schwarz
Simon-Denk-Gasse 2/28
1090 Wien

Wien, im April 2014

Die Begutachtung dieser Arbeit erfolgte durch:

1. **Ao. Univ.-Prof. Dipl.-Ing. Dr. G. Matz**
   Institute of Telecommuncations
   Technische Universität Wien

2. **Prof. Dr. Romain Couillet**
   Telecommunication Department
   École supérieure d’électricité (Supélec)

"Es ist nicht genug, zu wissen,
man muss auch anwenden;
es ist nicht genug, zu wollen,
man muss auch tun."

JOHANN WOLFGANG VON GOETHE

Abstract

Distributed computation emerged in computer science already some decades ago. Miniaturization led to efficient and low-cost designs of wireless sensor nodes and thus to a revival of distributed algorithms in the novel application area of wireless sensor networks. Many distributed algorithms have been developed for wireless sensor networks, but some are not efficient, i.e., they require a lot of communication or computational power at each sensor. A simple algorithm that runs efficiently in such networks is distributed averaging. We consider two distributed averaging schemes in detail: consensus propagation and average consensus. Both require only local communication and each sensor obtains the desired average in an iterative fashion. The main contribution of this thesis is to investigate their performance and introduce practical relevant extensions.

The first contribution in our work is an extension of consensus propagation that tracks the average of time-varying measurements. Our approach is shown to be stable and the performance is compared to an existing dynamic version of average consensus. A transformation of the update equations to the frequency domain allows us to study the behavior of both dynamic averaging algorithms through their transfer functions. We also introduce a linear filter to combat the delay inherent to dynamic consensus propagation. Moreover, we propose an efficient medium access protocol which is motivated by the well known ALOHA protocol and implement it in consensus propagation scenarios. Finally, it is shown how to use consensus propagation for field estimation through solving a least squares problem via distributed averaging. Numerical results that use random geometric graphs to model realistic wireless sensor networks verify our findings.

The next part of the thesis considers novel weight design methods for average consensus to improve performance. We start with analyzing the stability of the already known Metropolis-Hastings weights, where we derive novel conditions for the convergence. Furthermore, we present a method to design weights that achieve the best per-step mean squared error improvement assuming that the initial correlation of the measurements is known. These weights need to be computed every iteration and hence they are hard to use in practice. However, they provide a good performance benchmark for other weights that vary over time. One such weight design that we propose is called weight morphing. This method switches between two different weights (more precisely, one weight design morphs into another), by trying to use the better performing weight for the corresponding situation adaptively. In particular, we morph Metropolis-Hasting weights, which perform well for uncorrelated measurements, and the weights with the best possible asymptotic behavior, which yield good convergence for correlated measurements. The last contribution of this part of the work introduces blending with average consensus. Motivated by physics, we add an advective flow to the diffusive structure of average consensus. These flows are similar to the behavior of fluids in a blender. Besides a detailed derivation of the theory, we prove

the convergence of our method. Finally, we numerically compare all our weight designs in random geometric graphs.

The last part of our work addresses the behavior of average consensus when nodes are mobile. We derive generic lower and upper bounds on the mean-squared error, which verify the beneficial impact of mobility. Additionally, we provide a closed form expression for the lower bound in random geometric graphs with random hopping and compare it to numerical results.

Kurzfassung

Verteilte Algorithmen sind bereits seit Jahrzehnten ein Forschungsthema in der Informatik. In der Elektrotechnik hat die Miniaturisierung dazu geführt, dass man mittlerweile preiswerte und effiziente drahtlose Sensornetze herstellen kann. Somit wurde ein neues, relativ unerforschtes Anwendungsgebiet für verteilte Algorithmen erschlossen. In letzter Zeit wurden daher einige Algorithmen entwickelt, die speziell für drahtlose Sensornetze konzipiert sind, wobei manche dieser Algorithmen noch immer zu viele Ressourcen wie Leistung oder Zeit benötigen und somit nicht umsetzbar sind. Ein einfacher effizient in Sensornetzen anwendbarer Algorithmus ist die verteilte Mittelwertbildung. Wir behandeln zwei Varianten in dieser Arbeit: Consensus Propagation und Average Consensus. Beide Algorithmen benötigen nur drahtlose Kommunikation zwischen benachbarten Sensoren und der Mittelwert wird iterativ an jedem Knoten ermittelt. In dieser Dissertation befassen wir uns mit der Weiterentwicklung dieser Algorithmen und analysieren den erbrachten Nutzen.

Im ersten Teil dieser Arbeit stellen wir eine dynamische Version von Consensus Propagation vor, mit der man zeitlich veränderliche Messwerte mitteln kann. Wir beweisen die Stabilität dieses Algorithmus und vergleichen dessen Konvergenzverhalten mit dem von Average Consensus, von dem es bereits eine dynamische Variante gibt. Dies gelingt uns zum Teil durch die Analyse der Übertragungsfunktionen beider Algorithmen im Frequenzbereich. Außerdem zeigen wir, dass man durch lineare Filter eine zeitliche Verzögerung von Consensus Propagation beinahe gänzlich beseitigen kann. Damit Consensus Propagation in der Praxis funktioniert, muss man sich auch Gedanken über ein Kommunikationsprotokoll machen. Hierfür präsentieren wir eine Modifikation des bekannten ALOHA-Protokolls, welches relativ gut und vor allem einfach funktioniert. Zuletzt betrachten wir die verteilte Feldrekonstruktion als Anwendungsfall der Mittelwertbildung. Hierbei wird Feldrekonstruktion als verteiltes kleinste Quadrate Problem formuliert, welches durch eine simple Adaption von Consensus Propagation gelöst werden kann. Numerische Simulationen zeigen abschließend die Leistungsfähigkeit der Algorithmen, wobei das drahtlose Sensornetzwerk mittels geometrischer Zufallsgraphe modelliert wird.

Der nächste Teil dieser Arbeit befasst sich mit dem Entwurf der Gewichts-Koeffizienten bei Average Consensus, welche die Effizienz und die Konvergenzgeschwindigkeit erhöhen. Als erstes untersuchen wir die bereits bekannten Metropolis-Hastings Gewichte und leiten neue Bedingungen für die Konvergenz her. Danach stellen wir eine Methode vor, mit der man die Gewichte in jedem Zeitschritt optimiert, um die bestmögliche Fehlerreduktion zu erreichen. Leider ist diese Methode in der Praxis nicht einfach realisierbar, man kann sie jedoch als Vergleichsmaß für andere zeitvariante Gewichts-Entwürfe heranziehen. In diesem Zusammenhang schlagen wir eine Methode vor, bei der ein Gewichts-Design in ein anderes dynamisch übergeführt wird. Das führt zu verbesserter Konvergenz, weil man immer jene Gewichte verwenden kann, welche für die aktuelle

Korrelationsstruktur der zu mittelnden Werte besonders geeignet sind (z.B. Metropolis-Hastings Gewichte für unkorrelierte Daten). Weiters stellen wir eine Möglichkeit vor mit der man ein Geschwindigkeitsfeld, welches dem einer Flüssigkeit in einem Mixer gleicht, Average Consensus überlagert und damit die Mittelwertbildung beschleunigt. Für diesen Ansatz zeigen wir auch, welche Eigenschaften erfüllt sein müssen, damit das Gesamtsystem konvergiert. Schlussendlich vergleichen wir das Konvergenzverhalten all unserer neuen Gewicht-Designs anhand von geometrischer Zufallsgraphen.

Im letzten Teil dieser Arbeit widmen wir uns mobilen Szenarien. Darunter verstehen wir, wenn sich Sensoren während der Mittelwertbildung bewegen. Wir leiten zur Effizienzanalyse eine obere und untere Konvergenzschranke her und zeigen damit, dass sich Mobilität fast immer positiv auf die Konvergenzgeschwindigkeit auswirkt. Zuletzt berechnen wir noch analytisch die untere Schranke für ein spezielles Bewegungsmodell, bei dem sich die Sensoren sprunghaft bewegen und vergleichen das Ergebnis mit numerischen Simulationen.

Contents

1 Introduction 1

1.1 Averaging in Wireless Sensor Networks . . . 3
1.2 Scope of Work and Contributions . . . 5
1.3 Outline of the Thesis . . . 7

2 Preliminaries 9

2.1 Wireless Sensor Networks . . . 10
  2.1.1 Graph Theory . . . 10
  2.1.2 Network Topologies . . . 11
2.2 Distributed Averaging . . . 13
  2.2.1 Consensus Propagation . . . 14
  2.2.2 Average Consensus . . . 15
  2.2.3 Nonlinear Average Consensus . . . 17
  2.2.4 Dynamic Average Consensus . . . 17
  2.2.5 MSE Definition for Distributed Averaging . . . 18
2.3 Partial Differential Equations . . . 20
2.4 Low-Pass Fields . . . 21
  2.4.1 Sampling on a Regular Grid . . . 21
  2.4.2 Irregular Sampling . . . 22
  2.4.3 Construction of a Low-Pass Field . . . 22
  2.4.4 Correlation Between Two Points . . . 23

3 Distributed Dynamic Averaging 25

3.1 Static Consensus Propagation . . . 26
3.2 Dynamic Consensus Propagation . . . 27
  3.2.1 Incidence Matrix . . . 28
  3.2.2 State Space Model . . . 29
  3.2.3 Stability and Dynamic Behavior of Dynamic CP . . . 30
  3.2.4 Frequency Domain Analysis . . . 31
  3.2.5 Linear Prediction and Equalization . . . 32

3.3 State Space Model for Dynamic Average Consensus 34
3.4 Consensus Propagation in WSN 35
  3.4.1 Broadcasting 35
  3.4.2 Medium Access of CP 36
3.5 Application: Field Reconstruction 40
  3.5.1 Subspace Model 40
  3.5.2 Measurement Model 41
  3.5.3 Distributed Least-Squares 42
  3.5.4 WLS Approach 43
3.6 Numerical Results 44
  3.6.1 Static Averaging 44
  3.6.2 Dynamic Averaging 46
  3.6.3 Distributed Field Reconstruction 49
3.7 Conclusions 49
3.A Proofs 50
  3.A.1 Diagonal Dominance of CP 50
  3.A.2 Proof of Proposition 1 51
  3.A.3 Bounds on the κ Messages 52

4 Weight Design Methods for Average Consensus 53
4.1 Convergence Conditions for Constant and Metropolis-Hastings Weights 54
  4.1.1 Sufficient Conditions for Convergence 54
  4.1.2 Necessary Conditions for Convergence 55
  4.1.3 Discussion 57
  4.1.4 Numerical results 57
4.2 Optimum Symmetric Weights 59
  4.2.1 MSE Derivation 59
  4.2.2 MSE Minimization 60
  4.2.3 Uncorrelated Measurements 63
  4.2.4 Numerical Solver 63
  4.2.5 Numerical Results 66
4.3 Weight Morphing 67
  4.3.1 General Method 68
  4.3.2 Global Deterministic Morphing 69
  4.3.3 Local Adaptive Morphing 69
  4.3.4 Interpretation 70
  4.3.5 Dynamic Case 70
  4.3.6 Numerical Results 71
4.4 Blending with Advection-Diffusion 72
  4.4.1 Basic Theory 73
  4.4.2 Discretization of the Advection-Diffusion Equation 73
  4.4.3 Relation Between AC and Advection-Diffusion 79
  4.4.4 Design of the Velocity Field 79
  4.4.5 Convergence Analysis 80

4.4.6 Blending in WSN with Arbitrary Topology 81
4.4.7 Convergence Analysis for Arbitrary Topologies 82
4.4.8 Numerical Results 83
4.5 Comparison and Conclusion 86
4.5.1 MSE Convergence for Static Averaging 86
4.5.2 MSE Performance of Dynamic Averaging 87
4.5.3 Conclusions 89
4.A Alternative Convergence Analysis 90

5 Average Consensus with Mobile Nodes 97
5.1 AC in Mobile WSN 98
5.1.1 Definition of the MSE 98
5.1.2 Upper Bound on the MSE 99
5.1.3 Lower Bound on the MSE 100
5.1.4 Mean Frobenius Norm 101
5.1.5 Static WSN 102
5.1.6 Mobile WSN 103
5.2 Numerical Results 112
5.3 Conclusion 114
5.A Expectation of the Frobenius Norm 115
5.A.1 Diagonal Elements 115
5.A.2 Off-diagonal Elements 119

6 Conclusions and Outlook 123
6.1 Conclusions 124
6.2 Outlook 124

Bibliography 127

1

Introduction

Wireless sensor networks (WSN) gained a lot of interest during the last decades, since the range of potential application seems to be almost unlimited. Many of those require specific research to achieve the prescribed tasks. Moreover, the necessary research is often not only located in just one research field, but rather in many different areas, e.g., signal processing, wireless communications, and also hardware design. Hence, WSN became a promising and forward-looking research field.

Before describing the contributions of this thesis, we want to discuss WSN in general, in particular their components, the assembly of the individual sensors, and applications [1, 2].

A WSN consists of several sensors which are distributed. The communication between the sensors is performed in a wireless fashion. The sensors or sensor nodes serve for measurements, e.g., a physical quantity such as temperature or air pressure. Then, the measured quantities are used to infer something, for instance the location of the heat source or the air pressure distribution. This step is preferable done in a cooperative fashion, which means that the measurements of more than one sensing unit is used and the processing load is also distributed among the sensors. The latter should reduce the computational load at each unit and also not require too much wireless communication power. Finally the derived result may be distributed such that it is available at more than just one unit. So far it is seen that many different challenges arise in WSN, but nevertheless, each application requires different specification of the WSN, and hence the needs and requirements vary significantly. Before we discuss some applications, we briefly review the major components of each sensor, cf. Fig. fig:sensor node.

First of all, measuring a quantity requires a sensing device. Some years ago, analog sensors combined with a measurement amplifier and converter were used such that the analog data can be captured by an analog-digital converter. All these components require space, power, and careful circuit development. In recent years, sensors for various tasks were engineered, which integrate all the previous mentioned components and are even of small size. Moreover, the costs of these sensors are also low and since the measured data of many of them is accessible via a digital bus, they can be used in a flexible manner (e.g., sensors can be added or replaced at each node). Usually high-speed synchronous measurement conversion of data in WSN applications is not

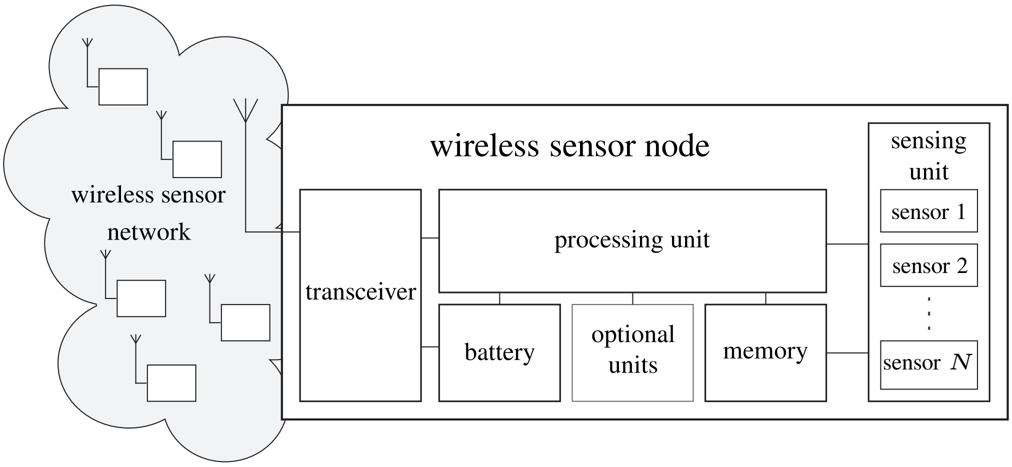


Figure 1.1: *Block diagram of a sensor node.*

necessary. Therefore, no complex application specific sensors are necessary and these new simple sensors meet the requirements perfectly.

The next important component is the processing unit and working memory. Since every new consumer electronics device is already ”smart” the price of a processing unit has decreased whilst the available computational power increased. Additionally, a variety of processors is available where each is suitable for a different type of task. But there can be nothing done against the fact that the consumed power rises with the processing power. Thus, the processor has to be chosen carefully and a trade-off between power consumption and processing abilities (speed, flexibility, ...) has to be made. The working memory is necessary to store the measurements and other data. Although chip based memory is easily and almost freely available, it requires space, power, and also engineering effort to adapt the hardware.

The power consumption brings us to the next component, the battery. Since each sensor has to be supplied with energy, it has to be equipped with a power source. In some cases it may be sufficient and possible that the energy which is available at or next to the nodes is used, e.g., power cable, but even the sun energy may be used via solar cells (just to mention one energy harvesting method). Most commonly, however, sensors are equipped with (rechargeable) batteries. A lot of progress was made by developing batteries based on Lithium which allowed to increase the power density significantly. Hence, novel batteries enable the usage of (powerful) processors which appeared impossible in WSN years ago. But battery power is still a limiting factor.

Finally, we have the communication unit. It is used by the processor to share the measurement or other information with neighboring wireless sensors. There exist integrated circuits that only need a few external parts to form a fully functional radio frequency front-end and there are complete transmission protocol stacks available, such that there is no need for developing sophisticated transmission schemes. With these tools, many challenging wireless communication issues in WSN can be solved. In some applications, however, it is still necessary to come up with a tailored wireless communication front-end to reduce the communication and computation overhead for the sake of saving battery power. It is worth noting that every value which is transmitted has to be quantized and precision obviously affects transmission power.

After this brief description of the sensor units in a WSN we discuss some (potential) applications. The driving forces for WSN are industrial and military applications. But also medical and environmental applications can be mastered through WSN. In industry, WSN can for instance be used to detect isolated inclusions and defects in materials such as concrete, but also in the exploration of raw materials WSN can be useful. But many industrial applications, however, still use wired sensor networks, due to reliability and transmission bandwidth. When it comes to military applications, WSN are often dedicated for target tracking/identification and surveillance. Here, the entire system has to be fault tolerant and reliable. Moreover, in many cases the systems need to be dynamic, i.e., the measurements vary over time, the targets and even the sensors move. Even in environmental applications the measured quantities may vary over time. For instance a WSN can determine the humidity distribution of a certain area and control irrigation, but also target tracking can be used to find a contaminating source. Finally, in medical applications sensors can be distributed on the human skin to observe the distribution of the body temperature. Of course there exist many more applications, and it may will be that many applications are not even conceived.

The variety of application requires different algorithms to share and process the data and infer the desired outcome. If there exists no constraints on the amount of communication, each sensor forwards its measurement to a central node, which does the processing and then spreads the result if necessary. In many cases, however, the communication is limited and moreover there should not exist a prioritized node (e.g., a failure of this sensor causes a breakdown of the entire system), and therefore local computation is performed. Hence, in many cases this scheme also shares the total processing load between the nodes. But a major challenge is to reformulate an algorithm such that it can be executed in a distributed fashion. Moreover the precision of the result may suffer, because the variables between the sensors have to be quantized sufficiently and sometimes the convergence of the distributed algorithm is slower than the centralized version. A promising and often used method to obtain a distributed algorithm is to break a centralized algorithm apart, and use well proven distributed algorithms for solving each part. But it is also possible that the resulting distributed version of an algorithm requires more resources than its centralized equivalent and therefore the advantages and disadvantages have to be compared carefully.

1.1 Averaging in Wireless Sensor Networks

Wireless sensor networks (WSN) became attractive in the signal processing community recently, since this research field enables many promising real-world applications (e.g. real-time area surveillance, environmental monitoring, and distributed localization). Several books (e.g. [1] and [2]) provide a good introduction through a comprehensive overview of the theory and applications of WSN. In this thesis we concentrate on WSNs that compute the average of the sensor measurements in a fully distributed manner. We consider homogeneous networks where each sensor node faces the same computational effort and asymptotically achieves the same result. The execution of the averaging algorithm has to comply with the constraint that only local information (i.e., local measurements and messages from the neighboring sensors) is available and can be used.

The most popular method for distributed averaging, named average consensus (AC), originates in the thesis of Tsitsiklis [3]. Further developments and a comprehensive discussion of distributed computation can be found in [4]. AC received a lot of attention with many papers considering this approach and providing extensions.

An extensive survey on AC can be found in [5]. A special case of AC is the gossip algorithm (or pairwise averaging) [6] which allows for asynchronous communications.

Consensus propagation (CP), introduced in [7] and [8], is an entirely different approach for distributed averaging. It is based on belief propagation (BP) which was initially proposed in [9] and afterwards extended to a powerful tool for many research fields. The goal of belief propagation is to compute a marginal posterior distribution (belief) by passing messages in a graph, that models the inference problem. An extensive introduction to BP can be found in [10]. BP converges to exact marginal for tree graphs, but for general graphs it does not necessarily converge. For the special case of Gaussian belief propagation (GaBP), i.e., all random variables are jointly Gaussian distributed, several works [11, 12] consider convergence conditions for general (loopy) graphs. In [11] they show that for settings where diagonal dominance of the information matrix is satisfied, the mean of the Gaussian marginal distribution obtained with GaBP converges to the true mean; this is not true, however, for the covariance, which converges but not generally to the correct matrix. In [12] the authors use a walk-sum framework to show convergence of the mean. The thesis of Bickson [13] provides an introduction to GaBP and describes many applications, such as solving a linear system of equations (see also [14] for more information). Since CP can also be viewed as an application of GaBP and the conditions of [11, 12] are fulfilled inherently due to the structure of CP, it will always converge to the true mean. Moreover, the authors of [7, 8] provide a more specific convergence proof for CP which is based on the algorithmic structure. Very recently Zhang et al. published in [15] a new approach to solve CP and also more general problems through coordinate-descent message-passing that requires slightly more local processing, but less communication.

In the following we discuss useful existing extension and applications of CP and AC related to WSNs. In its elementary form AC requires a fully synchronized operation, which is hard to achieve in WSNs, especially if only simple sensor nodes are available. The gossip version of [6] circumvents this problem by being fully asynchronous, but it has the drawback of being slower in terms of convergence than the generic form of AC. A comparison of the gossip algorithm and AC in the context of WSNs can be found in [16]. Alternatively, protocols to run AC asynchronously are provided in [17], but these are more suitable for stable networks such as the Internet, rather than for WSNs. CP, in contrast, can be either performed synchronously or asychronously. The latter requires total asynchronism of the messages which ensures that in the limit all messages are updated infinitely often (the exact definition can be found in [4]). Besides asynchronous processing the efficient usage of the physical medium plays an very important role for saving energy in WSNs. Here, the pairwise processing of the gossip algorithm [6] is inefficient, but a broadcasting extension described in [18] resolves this deficiency. In contrast to AC, where the generic form incorporates broadcasting, CP exchanges individual messages between neighboring nodes and therefore the transmit energy scales with the number of edges, even in synchronous settings. Another method to save transmit energy is to increase the convergence speed of the averaging algorithm.

Several works consider weight optimization techniques for AC. An extensive overview of methods that optimize the asymptotic convergence of AC is provided in [19]. A drawback of many of those techniques is that the network topology needs to be a priori known. Even though it is possible to assign edge weights to CP, this has not been considered in the literature so far. A different method to accelerate AC is proposed in [20] where a predictor for the inner states is designed. Modifying the AC weights over time can also improve the convergence speed [21]. Another aspect in WSNs is the medium access. In particular, low cost sensors with simple radio frequency modules will cause collisions and interference. So far random link failures in AC are considered in [22, 23]. To prevent collisions the authors of [24] consider a protocol for collision-free transmission in WSN. A different method is to study the SINR required in WSN (in [25] this is done for random geometric graphs) to achieve error-free transmission with appropriate channel coding. The analysis of the effect of interference on the convergence speed of AC is done in [26].

Continuous-time dynamic extensions of AC were proposed in [27, 28], which extend the field of possible applications significantly. A more practical discrete-time version of dynamic AC was proposed in [29]. Clearly, instead of using a dynamic version of distributed averaging, it is also possible to run the static version of distributed averaging for each sampling step.

The computation of the mean is not the only application for distributed averaging. Generally speaking all problems involving a sum where the summands are distributed over the WSN can be solved via distributed averaging. Algorithms which use this technique are distributed computation of the joint likelihood function [30, 31], distributed ML [32], field estimation [33], and distributed control [34], just to mention a few.

1.2 Scope of Work and Contributions

In this thesis we consider three main topics: dynamic distributed averaging, weight design methods for AC, and AC with mobile nodes. In the part where we study dynamic distributed averaging we extend CP for dynamic scenarios, i.e., the sensors' measurements vary over time and hence the corresponding algorithm has to compute the time-varying average. Such an extension already exists for AC [29] but as our numerical results show, it is also reasonable to establish a CP based approach. Along with the novel algorithm comes a state-space representation of the linearized version of dynamic CP which allows to compare its transfer behavior with the dynamic version of AC. In addition we consider a medium access method for CP and theoretically analyze potential collisions of the messages which are transferred between the nodes. As a practical application, we show how to solve distributed field estimation with dynamic CP.

The second part considers novel weight designs for AC, which improve the convergence performance. First of all we investigate the convergence properties of a generalized version of Metropolis-Hastings (MH) weights. Then we introduce time-varying weights for AC and propose two approaches. One considers a per-step MSE optimum weight design, whereas the other lets one weight design morph into another heuristically. Numerical results verify that our novel methods achieve superior convergence speed. The last weight design is based on physical advection-diffusion processes and inspired by the work of [35]. The idea is to emulate a physical flow with static weights.

Finally, in the last part we introduce mobility of the nodes when AC is applied. We derive analytical bounds on the convergence. For further investigations, we consider a simple motion model, for which we derive closed form results of our bounds. Our theoretical findings and also numerical results confirm the intuition that mobility can improve the convergence speed.

We next describe our contributions in more detail.

Distributed dynamic averaging

- *Dynamic consensus propagation.* Initially proposed in our own work [36,37], we show in this thesis how to extend CP such that it allows averaging of time-varying measurements. A novel formulation of the incidence matrix allows us to derive a state space model. This model enables analytical studies of the stability and of the dynamic behavior. In addition we transform the system into the frequency domain and compare the transfer function with dynamic AC. Finally, we show that appending FIR filters can improve the performance (e.g. compensate a delay with linear prediction).

- *Medium access with distributed averaging.* We provide a broadcast scheme for CP (proposed in [36]) that significantly reduces the transmission power in WSN. A related scheme was independently developed in the thesis of Bickson [13]. In the case of AC the authors of [18] introduced a broadcast scheme. Then we extend our medium access method [38] which is based on ALOHA [39] and present theoretical statistical findings. Our numerical simulations show that our medium access method with collisions (i.e., some messages do not arrive at the receiver) even outperform the perfectly synchronized version of CP.

- *Distributed field reconstruction.* A potential application of distributed averaging is distributed field reconstruction. As shown in [33], we reformulate the task as a least squares problem by introducing a basis expansion. Then, CP is adopted to solve the distributed least squares problem.

Weight design methods for average consensus

- *Convergence of generalized Metropolis-Hastings weights.* During our investigations, we were not able to find a satisfying proof that AC with MH weights from [40] becomes unstable if and only if the graph is regular and bipartite. A version of the MH weights which is stable for all types of graphs was used in [41], but we were not able to find a theoretical analysis. Therefore, we introduce generalized MH weights (cf. [42]) and provide a detailed analysis. Additionally we present numerical results that verify that choosing the conservative version as in [41] causes a decrease of the performance in various WSN settings.

- *Per-step MSE optimum weight design.* We propose a weight design method that optimizes the overall convergence of AC in the mean-square sense which we published in [43]. This establishes an ultimate performance benchmark for other weight designs. The result is related to the work of [44] where the authors optimize weights to achieve a maximal MSE decrease for uncorrelated states. We additionally provide results for correlated states, which is practically useful and necessary since the AC iterations render the states increasingly correlated. In contrast to [44], we further derive closed-form solutions for the optimal weights for the case of correlated and uncorrelated states. Finally, we show numerical simulations that illustrate the performance of the proposed weight design.

- *Weight morphing.* Motivated by [21] we present a novel nonlinear method for designing the AC weights introduced in our work [45]. Our method involves a local linear combination of differently designed weights where the coefficients in the linear combination depend nonlinearly on the previous states/measurements, thereby rendering the whole design nonlinear. Our approach is valid for both the static and the dynamic case of AC. We demonstrate by means of numerical simulations that our method outperforms existing weight designs in terms of convergence speed and residual error.

- *Weight design motivated by advection.* Inspired by [35] we derive an improved AC weight design (published in [46]) based on the advection-diffusion process for realistic WSN scenarios. More specifically, we use a discretization for advection-diffusion processes that differs from [35] and yields improved performance. For the case where sensors are arranged on a uniform grid, we propose new velocity fields and provide a stability analysis for the resulting AC algorithm (following [47]). Then we generalize our approach for arbitrary network topologies and present an optimization approach for the advection component of the AC weights. Finally, we illustrate the performance gains of our AC weight designs via numerical simulations.

The impact of mobility on average consensus

- *Performance bounds.* In related works, the effect of node mobility on gossip algorithms was studied analytically in [48] and motion dynamics in WSN is considered in the context of diffusion algorithms in [49]. With regard to AC, static WSN with dynamically switched links have been investigated in [22]. In contrast to those works, we derive an analytical lower and upper bound on the convergence for AC with mobile nodes (we initially presented the lower bound in [50]). We consider a setting which we can model through stationary Markovian evolving graphs [51] where the statistical graph structure is always preserved.

- *Closed form results for hopping mobility.* We derive a closed form solution for our lower-bound given a random hopping mobility scheme. We present the expression for the bound and a detailed derivation, which was missing in [50] due to the page limitation.

1.3 Outline of the Thesis

- In **Chapter 2** we provide the reader with a short introduction to the necessary theoretical background for the subsequent chapters, i.e., we introduce WSN, distributed averaging algorithms, partial differential equations, and the sampling of low-pass fields.

- **Chapter 3** focuses on dynamic distributed averaging. In particular, we present a dynamic extension of CP and investigate its state space model. Moreover we introduce our medium access method for distributed averaging, show how distributed averaging can be applied to field estimation, and finally we numerically compare CP with average consensus.

- **Chapter 4** considers weight design method for AC. First we provide a proof for the convergence of MH weights, then we show how to exploit the statistics of the measurements to obtain step-wise optimum weights. Inspired by this idea, we present our heuristic weight morphing scheme that yields superior performance compared to traditional static weight design methods. Finally, we show how to apply the idea of advection with average consensus in WSN which improves the performance significantly.

- In **Chapter 5** we theoretically study the impact of moving nodes when we apply average consensus. More specifically we derive bounds for the convergence and underline our results with numerical simulations.
- **Chapter 6** provides a conclusion of the thesis and gives an outlook for potential research areas that arise due to the findings of this thesis.

2

Preliminaries

IN this chapter we provide a brief introduction to concepts which are necessary for the understanding of the main contributions of this work. We want to emphasize that this introduction deals only with the most relevant background information and moreover we try to present it in a compact fashion such that it does not distract the reader from the main contributions in the next chapters. Hence, we refer to the cited literature if a more detailed insight is desired.

First of all, we start with a brief introduction of wireless sensor networks (WSN), which includes the basics of graph theory and network topologies used throughout our work. Then we discuss the general idea of distributed averaging followed by a short summary of two specific algorithms, i.e., consensus propagation (CP) and average consensus (AC). In the context of AC, its nonlinear and dynamic extension are also considered. For the performance quantification of distributed averaging, we define a proper metric that is based on the mean squared error (MSE). Furthermore we briefly discuss partial differential equations and their relevance in the field of distributed averaging. This chapter is finalized with a discussion of low-pass fields including an explanation how to sample them.

2.1 Wireless Sensor Networks

A wireless sensor network consists of several sensing nodes that can communicate with each other. By now very small sensor nodes are available, which are also cheap in production. Moreover, these sensor nodes can be easily adapted to run simple applications (cf. [52]). Advanced scenarios require specific nodes, designed for the corresponding needs. Applications of WSN are for instance: health applications, environmental applications, industrial applications, home applications, and military applications (in [1] a more comprehensive description can be found). In all those applications we can distinguish between centralized and fully distributed sensor networks. In the first case most of the nodes only sense, preprocess, and then transmit their data to one ore more fusion centers, which do the final processing task. The fusion centers usually possess a more powerful processing unit than the sensing nodes since they have to solve more complex mathematical problems. Such a setting is for instance considered in a clustered sampling scenario [53]. In a fully distributed setting, however, each node performs the same tasks. In practice of course it may happen that the importance of some nodes is higher, which can be for instance caused by topological issues.The advantage of a WSN with almost equal prioritized nodes is that they are more fault tolerant (i.e., a loss of a fusion center is not possible) and the result is automatically available at each sensor. A drawback, however, is that some algorithms cannot be distributed efficiently and hence there is a lot of communication between the nodes necessary and the computational complexity at each node is also high. In this work we only consider the fully distributed setting.

To fulfill the prescribed task, each sensor has to consist of at least the following units: the sensing unit, the processing and memory unit, the transceiver unit, and the energy unit [54]. In some applications it is also necessary that the sensors have to infer their own geographic position, for which an extra unit may be necessary (e.g., a GPS module) or algorithms can be used to perform self-localization [55]. We are not going to specify the individual units, but we consider that the output of the sensing unit is given and used by the processing unit to generate messages that are transmitted. In some parts of this work, we assume that the sensor positions are known but we do not consider a specific method.

2.1.1 Graph Theory

The communication unit allows each sensor to share information with other sensors. The ability of nodes to interact suggests to model the sensor network in terms of a graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes with $I = |\mathcal{V}|$ being the number of nodes. The edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the wireless communication links in the network. Topological properties of $\mathcal{G}$ induced by wireless communication are discussed in the next section.

A graph can either consist of directed or undirected edges. In this work undirected graphs are considered unless stated otherwise. The edge set can be represented by the adjacency matrix $\mathbf{A}$, where $[\mathbf{A}]_{ij} = 1$ if and only if $(i,j) \in \mathcal{E}$ [56]. The adjacency matrix becomes symmetric for undirected graphs since $(i,j) \in \mathcal{E}$ implies $[\mathbf{A}]_{ij} = [\mathbf{A}]_{ji} = 1$. We only consider simple graphs which exclude multiple edges between nodes and self-loops (i.e., the diagonal elements of $\mathbf{A}$ are zero).

A different matrix to describe a graph is the Laplacian, defined as $\mathbf{L} = \mathrm{diag}\{d_1, d_2, \ldots, d_I\} - \mathbf{A}$. Here, $d_i = |\mathcal{N}(i)|$ denotes the degree of node i, i.e., the numbers of neighbors, where $\mathcal{N}(i) = \{j \mid (i,j) \in \mathcal{E}\}$ defines the neighbor set of node i; furthermore, $\mathrm{diag}\{\cdot\}$ converts a vector into a diagonal matrix or extracts the diagonal of a matrix into a vector. It can be shown that the Laplacian always has an eigenvalue equal to zero, see [57] for more details. The multiplicity of the zero eigenvalue indicates the number of distinct components of the graph, i.e., a multiplicity of one indicates a connected graph consisting of one component.

The incidence matrix [56] of a graph represents the relation between nodes and edges. A commonly used definition of the incidence matrix which is denoted in this work as $\mathbf{B}$ contains $+1$ for outgoing edges and -1 for incoming edges. The graph Laplacian can be obtained from this incidence matrix as $\mathbf{L} = \mathbf{B}\mathbf{B}^T$. In the case of undirected graphs the sign is irrelevant and therefore is sometimes discarded. In Section 3.2.1 we will introduce an alternative definition which is tailored to our specific needs.

An extension of ordinary graphs are the so called weighed graphs $\mathcal{G}_\mathbf{w}$, where $\mathbf{w}$ is the weight vector of size $|\mathcal{E}|$ and contains a weight for each edge. We assume simple and undirected weighted graphs with strictly positive weights, i.e. $[\mathbf{w}]_l > 0$. In the case when $\mathbf{w} = \mathbf{1}$, weighted graphs reduce to ordinary graphs as defined in the previous paragraphs.

The adjacency matrix for a weighted graph $\mathcal{G}_\mathbf{w}$ is denoted as $\mathbf{A}_\mathbf{w}$ and the element $[\mathbf{A}_\mathbf{w}]_{ij}$ is defined through the corresponding edge (i,j) denoted as l. If $(i,j) \in \mathcal{E}$ then $[\mathbf{A}_\mathbf{w}]_{ij} = [\mathbf{A}_\mathbf{w}]_{ji} = [\mathbf{w}]_l$, zero otherwise. The definition of the neighborhood remains the same and the degree of node i is given by

$$d_\mathbf{w}(i) = \sum_l |[\mathbf{B}]_{il}| \, [\mathbf{w}]_l \, .$$

To be consistent, the definition of the weighted Laplacian is $\mathbf{L}_\mathbf{w} = \mathrm{diag}\{d_\mathbf{w}(1), d_\mathbf{w}(2), \ldots, d_\mathbf{w}(I)\} - \mathbf{A}_\mathbf{w}$. The weighted Laplacian always has an eigenvalue equal to zero, which we also show in Section 4.1.1, and again, the number of zero eigenvalues equals the number of connected components [58]. The definition of the incidence matrix $\mathbf{B}_\mathbf{w}$ has to ensure that $\mathbf{L}_\mathbf{w} = \mathbf{B}_\mathbf{w}\mathbf{B}_\mathbf{w}^T$ and hence we have $\mathbf{B}_\mathbf{w} = \mathbf{B}\,\mathrm{diag}\{\sqrt{\mathbf{w}}\}$. Therefore, we can also compute the Laplacian of a weighted graph by using the incidence matrix of the corresponding unweighted graph as $\mathbf{L}_\mathbf{w} = \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T$. Finally, another important graph property is the bipartiteness [56]. It describes whether a graph can be separated into two distinct components, where there exist only edges between nodes that are not in the same component. It can be shown that a graph is bipartite if there exists an eigenvalue of the signless Laplacian $|\mathbf{L}|$ that is zero; see [59] for more details.

2.1.2 Network Topologies

The sensors of the WSN are deployed in a region $\mathcal{A}$, according to a random or deterministic distribution. The positions of the sensors are denoted $\mathbf{r}_i$, $i = 1, \ldots, I$. The graph topology defines the structure of the graph $\mathcal{G}$ and the associated matrices.

Random Geometric Graphs

An intuitive approach to model a WSN is through a random geometric graph [60]. In random geometric graphs the node deployment is random based on an underlying spatially uniform distribution. Any two nodes i and j are connected if they lie within a prescribed range r, i.e., if $\|\mathbf{r}_i - \mathbf{r}_j\|_2 < r$. This corresponds to a scenario in which each sensor node has the same transmit power and can transmit over a certain distance predominantly via free space propagation. Moreover, if sensor i can reach sensor j then this implies that also sensor j can reach

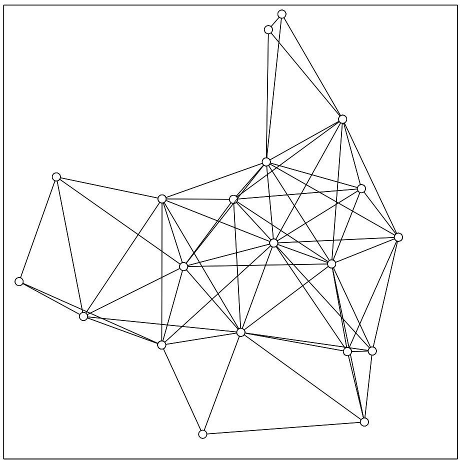


Figure 2.1: *Realization of a random geometric graph with* $I = 100$ *and* $c = 2$.

sensor i, according to the reciprocity of wireless channels. As the number of nodes I and the area of the region $|\mathcal{A}|$ tend to infinity with a fixed $\chi = \frac{I-1}{|\mathcal{A}|}$, the degree distribution for a random geometric graph converges to a Poisson distribution with parameter $r^2\pi\chi$. For the sake of simplicity we define the normalized communication range (or connectivity) as $c = r\sqrt{\chi}$ which yields a Poisson parameter $c^2\pi$. Hence, the expected number of neighbors of each node is given by

$$\mathrm{E}\{|\mathcal{N}(\cdot)|\} = c^2\pi\,.$$

It is seen that the connectivity c represents the local structure of a random geometric graph. A realization of a random geometric graph is shown in Fig. 2.1. We mainly concentrate on random geometric graphs throughout this work because we think they are versatile and realistic for WSN.

Scale-free Graphs

Another interesting network topology is given by scale-free graphs. In such graphs the degree statistics adhere to an exponential distribution, which leads to a few highly connected nodes (also called "hub" nodes) and many nodes with few neighbors. Prominent instances of scale-free graphs are the world wide web and social networks. A method to construct such graphs was proposed by Barabási and Albert (cf. [61]) as follows: Initially a small arbitrarily connected graph $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ is chosen. In each iteration of the algorithm an additional node is added and connected to $\xi \geq 1$ nodes of the graph, where ξ is a parameter of the algorithm. The new node is more likely connected to a node with a higher degree than to a node with a lower degree. As soon as the desired number of nodes I is reached the algorithm stops. The expected number of edges of a Barabási-Albert graph is $|\mathcal{E}| = |\mathcal{E}_0| + \xi(I - |\mathcal{V}_0|) \approx \xi\, I$. One drawback of this algorithm is that it cannot be directly used in the context of WSN, because the construction method does not take any topological and wireless transmission issues into account. To obtain a scale-free structure in WSN long range connections need to be established through high power communication links or through heterogenous communication, such as additional wires or multi-hop transmissions.

Regular Graphs

In a WSN were the nodes are distributed according to a lattice and each sensor has the same communication range the resulting graph (neglecting the boundary) has the structure of a deterministic regular graph. In general regular graphs each node has the same number of neighbors. The edges between the nodes can also be chosen at random which then results in a random regular graph (see [62] for more details). For any regular graph we have $|\mathcal{E}| = \frac{1}{2}dI$, where d denotes the degree of each node.

Periodic Graphs

These type of graphs avoid any boundaries because it is assumed that the nodes are placed on a toroidal surface. Hence, such graphs possess a spatial periodicity. All the three aforementioned graph types can be constructed assuming that the nodes lie on a torus which helps to confirm theoretical results empirically since boundary effects do not occur.

2.2 Distributed Averaging

In this section we provide a brief introduction of distributed averaging in a WSN. First of all, it is assumed that each sensor in the network measures a physical quantity at discrete time instants. We consider a discrete-time model since it has more practical relevance than a continuos-time model. The measurement of sensor i at (discrete) time n is denoted by $s_i[n]$. For simplicity, measurement quantization is ignored. Quantization would certainly influence the performance of the proposed methods, but we do not consider it throughout this work. The goal of distributed averaging is to determine the time-varying spatial average

$$\bar{s}[n] = \frac{1}{I}\sum_{i=1}^{I} s_i[n]$$

at each sensor node by exchanging information in the sensor network. Distributed averaging is a part of more sophisticated algorithms that compute more general functions of the measurements (see Section 3.5 for an example). The algorithms which are considered in this work allow messages to be exchanged only between neighboring sensors, i.e., sensors that can directly communicate with each other.

We can distinguish between static and dynamic distributed averaging:

- in the static case it is assumed that $s_i[n]$ stays constant during the computation of the average, i.e., averaging is performed within a sampling period;
- in the dynamic case averaging is done online and new measured values are directly involved in the averaging process.

In other words, with static distributed averaging an instance of the algorithm is executed at each temporal sampling point. If the measurements stay constant ($s_i[n]$ is time invariant), it is not necessary to start additional instances. Therefore, it is reasonable to omit the time index n when static averaging is considered.

For convenience we stack the measurements into a measurement vector $\mathbf{s}[n] = (s_1[n]\ s_2[n]\ \ldots\ s_I[n])^T$ and can therefore write the averaging problem as $\bar{s}[n] = \frac{1}{I}\mathbf{1}^T\mathbf{s}[n]$, where $\mathbf{1}$ is the all-ones vector. Finally, as already mentioned before, the goal is that the average $\bar{s}$ is available at each node, which can be described with

$$\bar{\mathbf{s}}[n] = \mathbf{1}\bar{s}[n] = \frac{1}{I}\mathbf{1}\mathbf{1}^T\mathbf{s}[n] = \mathbf{J}\mathbf{s}[n]\,,$$

where $\mathbf{J} \triangleq \frac{1}{I}\mathbf{1}\mathbf{1}^T$ is the orthogonal projection matrix on to the span of $\mathbf{1}$. In case a graph is not connected, i.e., there exist at least two components of the graphs which are not linked through any edge, it is obvious that any distributed averaging algorithm does not work properly. In particular, the global averaging separates into subproblems where distributed averaging is performed in each component independently.

In the remainder of this section we introduce distributed averaging algorithms, where we assume WSN topologies that do not change over time.

2.2.1 Consensus Propagation

Consensus Propagation (CP) was introduced in [7] and [8]. It is a distributed algorithm for averaging numbers, which consists of iterations that can be performed either asynchronously or synchronously. First, we give a brief overview of the theory behind CP and an explanation of the message update equations.

It is known that the average $\bar{s} = \frac{1}{I}\sum_i s_i$ is the solution to the least-squares minimization problem

$$\bar{s} = \arg\min_{\theta} \|\mathbf{1}\theta - \mathbf{s}\|_2^2\,.$$

In a connected graph, this problem can be rewritten as a constrained minimization problem with equality constraints induced by the graph:

$$\begin{aligned}\bar{\mathbf{s}} &= \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \mathbf{s}\|_2^2 \\ &\text{subject to } \theta_i = \theta_j \quad \forall (i,j) \in \mathcal{E}\,.\end{aligned} \tag{2.1}$$

The side constraint ensures that the estimates are equal for all neighbors and hence in a connected graph all nodes obtain the same result. The next step approximates the problem such that it can be solved via Gaussian belief Propagation (GaBP) [11], i.e.,

$$\hat{\bar{\mathbf{s}}} = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{\theta} - \mathbf{s}\|_2^2 + \beta \sum_{(i,j)\in\mathcal{E}} q_{ij}(\theta_i - \theta_j)^2 \right\}\,, \tag{2.2}$$

where $\beta > 0$ and $q_{ij} = q_{ji} > 0$. Note that in general $\hat{\bar{\mathbf{s}}} \neq \bar{\mathbf{s}}$. However, as shown in [7] and [8], $\lim_{\beta\to\infty} \hat{\bar{\mathbf{s}}} = \mathbf{1}\bar{s}$ and hence the approximation yields arbitrarily accurate results with increasing β.

It is easy to transform the objective function in problem (2.2) into a product of exponential families (Gaussian structure) and then apply belief propagation, in particular the sum-product algorithm [63] (or equivalently for the Gaussian case the max-product algorithm). This leads to the estimate of $\bar{s}$ in an iterative manner. Since the construction of (2.2) enforces only local dependencies, i.e., each local estimate should be as close to the measurement as possible and the local estimates of neighboring nodes should be identical, the resulting algorithm only demands message exchanges between neighboring nodes. The advantage of GaBP is that the messages between nodes are Gaussians, which are completely characterized by their mean and variance. Hence, only two quantities need to be exchanged, which with CP are the mean and the inverse variance (denoted as μ and κ, respectively). For synchronous processing the following message update equations are performed for all sensors at iteration k:

$$\kappa_{i\to j}[k] = \frac{1+\sum\limits_{v\in\mathcal{N}(i)\setminus j}\kappa_{v\to i}[k-1]}{1+\frac{1}{\beta q_{ij}}\left(1+\sum\limits_{v\in\mathcal{N}(i)\setminus j}\kappa_{v\to i}[k-1]\right)},$$

$$\mu_{i\to j}[k] = \frac{s_i+\sum\limits_{v\in\mathcal{N}(i)\setminus j}\mu_{v\to i}[k-1]\kappa_{v\to i}[k-1]}{1+\sum\limits_{v\in\mathcal{N}(i)\setminus j}\kappa_{v\to i}[k-1]}.$$

Here, i is the source node and j is the destination node. It is seen that β and q_{ij} are the only parameters that influence the behavior of the algorithm (assuming a fixed network topology). Moreover the algorithm enforces each node to generate $|\mathcal{N}(i)|$ individual messages for its neighbors which is inefficient since the broadcast property of the wireless medium is ignored.

CP is based on GaBP, which does not necessarily require synchronous processing. This asynchronous processing can be directly applied to CP and requires that messages that are not updated keep their previous values, i.e., $\kappa_{i\to j}[k]=\kappa_{i\to j}[k-1]$ and $\mu_{i\to j}[k]=\mu_{i\to j}[k-1]$. But it is necessary that the κ and the μ messages are exchanged together.

The estimate of $\bar{s}$ at node i in iteration k is given by

$$\hat{\bar{s}}_i[k] = \frac{s_i+\sum\limits_{v\in\mathcal{N}(i)}\mu_{v\to i}[k-1]\kappa_{v\to i}[k-1]}{1+\sum\limits_{v\in\mathcal{N}(i)}\kappa_{v\to i}[k-1]}.$$

Since CP is an application of GaBP, the convergence conditions of GaBP apply. In [7, 8] however, the authors show that the individual messages converge to a fixed point even for asynchronous operation. All in all we have $\lim\limits_{n\to\infty}\hat{\bar{s}}_i[k]=\bar{s}$ for all i as β goes to infinity. In practice β has to be finite and hence the optimization problem (2.1) will not be solved exactly.

To date no general results regarding the convergence speed of CP have been obtained. In [7] the authors presented a bound on the convergence speed for the case of random regular graphs, but the bound is very loose.

2.2.2 Average Consensus

Average consensus (AC), initially proposed in [3], is a completely different approach to CP for distributed averaging. The only similarity with CP is that both involve the exchange of messages. With AC, each sensor initially copies its measurement into an internal state $x_i[0]$, i.e., $\mathbf{x}[0]=\mathbf{s}$. The state vector $\mathbf{x}[k]=\big(x_1[k]\; x_2[k]\; \ldots\; x_I[k]\big)^T$ is updated at every iteration, taking into account the states of the neighboring nodes:

$$x_i[k+1] = w_{ii}\,x_i[k] + \sum_{j\in\mathcal{N}(i)} w_{ij}\,x_j[k]\,.$$

Here, $w_{ij} \in \mathbb{R}$ denotes suitable weights (discussed below). This algorithm applies also to undirected graphs (see [64]), but we will restrict our discussion to directed graphs, i.e., all subsequent statements do not necessarily hold for undirected graphs. In a more compact form AC can be written as

$$\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k] \,. \tag{2.3}$$

Here the weight matrix is defined by $[\mathbf{W}]_{ij} = w_{ij}$, where $w_{ij} = 0$ unless $(i,j) \in \mathcal{E}$ or $i = j$. For undirected graphs the weights are in general symmetric, $w_{ij} = w_{ji}$, and hence we have $\mathbf{W} = \mathbf{W}^T$. To ensure convergence to the true average, the weight matrix has to satisfy the following properties: (i) $\mathbf{W}\mathbf{1} = \mathbf{1}$ and (ii) $\rho\{\mathbf{W} - \mathbf{J}\} < 1$. The first condition states that the all-ones vector is an eigenvector of $\mathbf{W}$ with associated eigenvalue 1. Since $\mathbf{W} = \mathbf{W}^T$, we also have $\mathbf{1}^T\mathbf{W} = \mathbf{1}^T$ and hence $\frac{1}{I}\mathbf{1}^T\mathbf{x}[k+1] = \frac{1}{I}\mathbf{1}^T\mathbf{W}\mathbf{x}[k] = \frac{1}{I}\mathbf{1}^T\mathbf{x}[k] = \frac{1}{I}\mathbf{1}^T\mathbf{x}[0] = \bar{s}$, i.e., the correct average is always preserved. The second condition forces the average to be the fixed point.

In the case when $\mathbf{W}$ is symmetric and the weights are positive, it is possible to express the weight matrix in terms of the Laplacian of the weighted graph $\mathcal{G}_{\mathbf{w}}$ where $\mathbf{w}$ corresponds to the AC weights. We can write

$$\mathbf{W} = \mathbf{I} - \mathbf{L}_{\mathbf{w}} = \mathbf{I} - \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T \,, \text{ with } \; [\mathbf{w}]_l > 0 \quad \forall l \,, \tag{2.4}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{B}$ the incidence matrix defined in Section 2.1.1. This definition of the weight matrix implicitly fulfills the first convergence condition due to the properties of $\mathbf{L}_{\mathbf{w}}$. Finally, even for $\mathbf{w}$ containing (some) negative weights the condition $\mathbf{W}\mathbf{1} = \mathbf{1}$ holds.

In the following we discuss different weight design methods.

Constant Weights

A trivial and simple approach is to assign to each edge the same weight α. Hence, by inserting $\mathbf{w} = \alpha\mathbf{1}$ we can rewrite (2.4) as

$$\mathbf{W}^{\mathrm{CW}} = \mathbf{I} - \alpha\mathbf{L} \,.$$

To ensure convergence, α has to fulfill $0 < \alpha < \frac{1}{\max\{d_i\}}$ for all $i \in \mathcal{V}$ (the proof is provided in [5]). Thus, α has to be positive and smaller than the maximum reciprocal the degree of the graph. In the case when the graph is not bipartite, it is a good choice to set $\alpha = \frac{1}{\max\{d_i\}}$. We discuss this matter in more detail in Section 4.1.

Metropolis-Hastings Weights

A common choice with favorable initial convergence performance is given by the local-degree weights which stem from the Metropolis-Hastings (MH) algorithm in Markov chain theory [40]:

$$w_{ij}^{\mathrm{MH}} = \begin{cases} \frac{1}{\max\{d_i, d_j\}} \,, & \text{for } (i,j) \in \mathcal{E}, \\ 0 \,, & \text{for } (i,j) \notin \mathcal{E} \text{ and } i \neq j, \\ 1 - \sum_{j \neq i} w_{ij}^{\mathrm{MH}} \,, & \text{for } i = j. \end{cases}$$

Here, d_i denotes the degree (the number of neighbors) of node i. A sufficient convergence condition for AC with these MH weights is that the graph is not bipartite (cf. [19]). The modified MH weights $w_{ij}^{\mathrm{MH}} = \frac{1}{\max\{d_i, d_j\}+1}$ always lead to convergence. We provide a detailed study of the convergence of AC with MH weights in Section 4.1.

Asymptotic Optimum Weights

Different approaches which optimize the asymptotic convergence speed were introduced in [19]. One of these methods obtains the symmetric weights that provide the fastest asymptotic convergence speed (which coincides with the optimum per step convergence with no prior knowledge of the measurements for undirected graphs) through solving a convex optimization problem. To this end we denote the corresponding weight matrix as $\mathbf{W}^{\text{CVX}}$. The optimization problem reads

$$\mathbf{W}^{\text{CVX}} = \arg\min_{\mathbf{W}} \rho\{\mathbf{W} - \mathbf{J}\}\,,$$

where the constraints are that the resulting weight matrix $\mathbf{W}^{\text{CVX}}$ has the same sparsity pattern as the corresponding adjacency matrix with the exception that the diagonal elements are non-zero and that $\mathbf{W}^{\text{CVX}}\mathbf{1} = \mathbf{1}$.

This optimization problem can be reformulated for symmetric $\mathbf{W}$ and using (2.4) as

$$\mathbf{W}^{\text{CVX}} = \arg\min_{\mathbf{w}} \left\|\mathbf{I} - \mathbf{B}\,\text{diag}\{\mathbf{w}\}\mathbf{B}^T - \mathbf{J}\right\|_2\,.$$

In [19] the authors provide a solver for this optimization problem. These weights cannot be efficiently designed in a distributed manner, but they provide a good baseline for the achievable performance.

2.2.3 Nonlinear Average Consensus

In practice it can be observed that the CVX weights provide an excellent asymptotic convergence rate, but compared to MH weights, they perform poor in the transient phase in settings with uncorrelated measurements. To improve the transient behavior of CVX, a nonlinear AC algorithm was proposed in [21]. With this method, the nonlinear (NL) state update is given by

$$\mathbf{x}[k+1] = \mathbf{W}^{\text{NL}}[k]\,\mathbf{x}[k]\,,$$

where the weight $\mathbf{W}^{\text{NL}}[k]$ depends on the current states and hence renders the updates time-varying and nonlinear. More specifically, the edge weights are defined as

$$w_{ij}^{\text{NL}}[k] = \begin{cases} w_{ij}^{\text{CVX}} f\big(x_i[k] - x_j[k]\big)\,, & \text{for } i \neq j, \\ 1 - \sum_{j \neq i} w_{ij}^{\text{NL}}[k]\,, & \text{for } i = j. \end{cases}$$

Here $f(\cdot)$ is a suitable nonlinear function, chosen as $f(u) = \frac{\tanh(\theta_1 u)\theta_2}{u}$ with $\theta_1\theta_2 \leq 1$ in [21]. The performance provided by the nonlinear weight design unfortunately depend on these θ parameters which have to be manually adapted to the specific scenario. Therefore, this heuristic approach might not always increase the convergence speed.

2.2.4 Dynamic Average Consensus

In [28] and [29] dynamic extensions of AC for time-varying measurements $s_i[n]$ were proposed. Due to our discrete time setting, we will stick to the first-order dynamic AC algorithm described in [29]. The dynamic AC updates are given by

$$x_i[n+1] = w_{ii}\,x_i[n] + \sum_{j \in \mathcal{N}(i)} w_{ij}\,x_j[n] + (s_i[n] - s_i[n-1]).$$

Initially $x_i[0]$ is set to $s_i[-1]$ for all i. The states $x_i[n+1]$ are an estimate of $\bar{s}[n]$. In vector notation the algorithm reads

$$\mathbf{x}[n+1] = \mathbf{W}\mathbf{x}[n] + (\Delta\mathbf{s})[n]\,, \tag{2.5}$$

where $(\Delta\mathbf{s})[n] = \mathbf{s}[n] - \mathbf{s}[n-1]$ is the first order difference. Note that the term $(\Delta s_i)[n]$ basically serves as a simple means to predict the measurement at time $n+1$. For the case of temporally constant measurements, $(\Delta\mathbf{s})[n] = \mathbf{0}$ and (2.5) reduces to the static AC (2.3). For more details regarding dynamic AC we refer to [29]. An extension of this algorithm uses higher-order differences but entails an increased communication overhead between the sensors. The optimal order depends on the amount of time-variation in the measurements.

The conditions imposed on the weight matrix in the static case (see Section 2.2.2) ensure

$$\mathbf{1}^T\mathbf{x}[n] = \mathbf{1}^T\mathbf{s}[n]\,,$$

which in turn implies that dynamic AC achieves an asymptotically vanishing error provided that $(\Delta s_i)[n]$ is independent of i (cf. [29]). Unfortunately, the latter condition is not satisfied in the case of noisy measurements and consequently the mean-square error (MSE) achieved with dynamic AC is rather high.

2.2.5 MSE Definition for Distributed Averaging

An important measure of the performance of distributed averaging is the MSE. The MSE itself is simple to derive, but its normalization is not unique. The normalization in particular is crucial for the comparison of different measurement settings. In the following we give the definition of the MSE, compare some normalization methods, and argue why we are choosing one of them for our work.

The squared error at node i at time n (in a static setting we have $k=n$ and $\bar{s}[n]$ is time invariant) for a given measure average $\bar{s}[n]$ is defined as:

$$\mathrm{SE}_i[n] = |x_i[n] - \bar{s}[n]|^2\,.$$

Furthermore, averaging over all nodes yields

$$\mathrm{ASE}[n] = \frac{1}{I}\sum_i |x_i[n] - \bar{s}[n]|^2\,.$$

It is obvious that $\mathrm{ASE}[n]$ depends on a certain measurement and on the graph topology. For that purpose it is reasonable to do Monte-Carlo experiments to suppress statistical outliers. Hence, we obtain for C Monte-Carlo experiments the mean average squared error:

$$\mathrm{MSE}[n] = \frac{1}{CI}\sum_c\sum_i \left|x_i^{(c)}[n] - \bar{s}^{(c)}[n]\right|^2\,,$$

where c denotes a certain scenario. For some investigations it is necessary to separate between graph and measurement scenarios (cf. Section 5.1.3) which is omitted in the following for simplicity.

The expectation over the different scenarios (letting C going to infinity) is then defined as

$$\mathrm{E}\{\mathrm{MSE}[n]\} = \mathrm{E}_c\left\{\frac{1}{I}\sum_i \left|x_i^{(c)}[n] - \bar{s}^{(c)}[n]\right|^2\right\}\,.$$

It is possible to do the following derivation

$$\begin{aligned}\mathrm{E}\{\mathrm{MSE}[n]\} &= \mathrm{E}_c\left\{\frac{1}{I}\left(\sum_i \left(x_i^{(c)}[n]\right)^2\right) - \left(\bar{s}^{(c)}[n]\right)^2\right\} \\ &= \frac{1}{I}\sum_i \mathrm{E}\left\{\left(x_i^{(c)}[n]\right)^2\right\} - \frac{1}{I^2}\mathrm{E}\left\{\left(\sum_i s_i^{(c)}[n]\right)^2\right\} \\ &= \frac{1}{I}\operatorname{tr}\{\mathbf{R}_{x[n]}\} - \frac{1}{I^2}\mathbf{1}^T\mathbf{R}_{s[n]}\mathbf{1}\,,\end{aligned}$$

which provides us with a stochastic interpretation of the MSE. First, $\mathbf{R}_{x[n]}$ is the correlation matrix of the state vector and $\mathbf{R}_{s[n]}$ is the correlation matrix of the measurements, both at time n. If consensus is reached, we obtain a correlation matrix $\mathbf{R}_{x[n]}$ where all elements are $\mathrm{E}\{\bar{s}^2[n]\}$.

In the next step we discuss normalization methods for the MSE, which should preferable allow to compare different settings. A trivial method for static settings is to normalize the MSE with the initial MSE

$$\mathrm{E}_c\left\{\frac{1}{I}\sum_i |s_i - \bar{s}|^2\right\}.$$

Such a normalization yields a weighted MSE of 1 (or 0dB) initially. Unfortunately it only represents the deviation from the initial measurements and does not involve the magnitude of the measurements. It is possible to extend this normalization to fit also for dynamic scenarios, by taking the expectation also over time n and obtain

$$w_1 = \frac{1}{I}\operatorname{tr}\{\mathbf{R}_s\} - \frac{1}{I^2}\mathbf{1}^T\mathbf{R}_s\mathbf{1}\,,$$

where $\mathbf{R}_s$ is the time-independent correlation matrix of the measurement vector, i.e., $[\mathbf{R}_s]_{ij} = \mathrm{E}_{c,n}\{s_i^{(c)}(n)s_j^{(c)}(n)\}$.

A normalization with the power of the mean takes the magnitude of the measurements into account,

$$w_2 = \mathrm{E}_{c,n}\left\{\bar{s}^2[n]\right\} = \frac{1}{I^2}\mathbf{1}^T\mathbf{R}_s\mathbf{1}\,,$$

but it is seen that adding an offset to the measurements leads to a completely different result of the normalized MSE while a change of the variance of the measurements has no effect.

To this end, it is reasonable to use the following normalization, since it involves both of the previous mentioned methods and can be therefore seen as a compromise

$$w_3 = \frac{1}{I}\sum_i P_{s_i} = \frac{1}{I}\operatorname{tr}\{\mathbf{R}_s\} = w_1 + w_2\,,$$

where $P_{s_i} = \mathrm{E}_{c,n}\left\{\left(s_i^{(c)}[n]\right)^2\right\}$.

To conclude, we define the normalized empirical MSE in the following as

$$\mathrm{WMSE}[n] = \frac{\sum_c\sum_i\left|x_i^{(c)}[n] - \bar{s}^{(c)}[n]\right|^2}{\frac{1}{N}\sum_n\sum_c\sum_i\left(s_i^{(c)}[n]\right)^2}\,. \tag{2.6}$$

An analogous result for the expected normalized MSE can be derived

$$\mathrm{E}\{\mathrm{WMSE}[n]\} = \frac{\mathrm{tr}\{\mathbf{R}_{x[n]}\} - \frac{1}{I}\mathbf{1}^T\mathbf{R}_{s[n]}\mathbf{1}}{\mathrm{tr}\{\mathbf{R}_s\}} \,.$$

When the statistic of the measurements is know, the denominator of (2.6) can be interchanged with the one of the expected version, i.e.,

$$\mathrm{WMSE}[n] = \frac{\frac{1}{C}\sum_c \sum_i \left| x_i^{(c)}[n] - \bar{s}^{(c)}[n] \right|^2}{\mathrm{tr}\{\mathbf{R}_s\}} \,,$$

which we will use throughout this work in our numerical simulations.

2.3 Partial Differential Equations

The AC algorithms has parallels to diffusion processes described through partial differential equations, which we will consider in more detail in Section 4.4.1. In [35] it is moreover shown that a discretized advection-diffusion process can also be used for averaging and applied similarly to AC (see Section 2.2.2). For the necessary background knowledge we give a brief introduction on partial differential equations and the advection-diffusion process subsequently.

For the modeling of many physical processes it is necessary to use differential equations, since the observed dynamics can be only described through change rates of continuos quantities. If the underlying process depends only on one quantity, simple differential equations can be used, but as soon as more variables are involved, partial differential equations have to be used. Some of these equations can be even solved in closed form, in particular if the equations are linear and if the side constraints are idealized (cf. [65]), but in general, especially if close to real systems are considered, approximation techniques have to be used (see [66] for more details). Important side constraints arise due to the boundaries of the system where the physical processes occur. Neumann and Dirichlet boundary conditions model a lot of the occurring phenomena at boundaries, but we refer to [67] for further information.

A physical process which can be formulated through partial differential equations is the advection-diffusion process [68]. Therefore, we proceed with a brief explanation of this process which is a combination of diffusion and advection (i.e., a bulk flow induced by an external velocity field).

To this end we consider a quantity $x(\mathbf{r}, t)$, with $\mathbf{r}$ and t denoting space and time, respectively. The diffusion is quantified by the diffusion coefficient D which can vary in space (e.g. in homogenous media) and time, but we assume that it is constant. The velocity vector $\mathbf{v}$ represents the advection, i.e., it describes the direction and magnitude of the velocity which affects bulk. In the following we consider $\mathbf{v}(\mathbf{r})$ to be constant over time, but it may vary over space. Under the assumption that there are neither sources nor sinks, $x(\mathbf{r}, t)$ is governed by the advection-diffusion equation

$$\frac{\partial x}{\partial t} = D\nabla^2 x - \mathbf{v} \cdot \nabla x \,, \tag{2.7}$$

where we omitted the arguments due to simplicity. Additionally the equation presumes that the flow created by the velocity field is incompressible, which means that there are no points where the velocity field starts or ends (i.e., $\nabla \cdot \mathbf{v} = 0$). The first term on the right-hand side of (2.7) is the diffusion part and the second term characterizes the advection.

If we assume boundary conditions which ensure that nothing of the quantity x gets lost, the diffusion process will enforce that the quantity $x(\mathbf{r}, t)$ becomes independent of $\mathbf{r}$, the advection process, however, does only provide an additional mixing, which helps to reach this equality state. In Section 4.4 of this work, we show the impact of using the idea of advection-diffusion in AC scenarios.

2.4 Low-Pass Fields

We consider spatial/temporal low-pass fields $f(\mathbf{r}, t)$ which are generated by filtering white Gaussian noise. First, we derive expressions for general sharp low-pass spectra, which can be directly applied on Gaussian noise. Finally we derive expressions for the correlations between two arbitrary points of the resulting field.

As already mentioned the field is represented by $f(\mathbf{r}, t)$. The total dimensionality is given by the number of spatial dimensions plus one (for the temporal dimension). For simplicity we assume that all dimensions are independent and therefore the field can be decomposed into one-dimensional (1D) factors that are 1D fields themselves. In the case of two spatial dimensions we have $f(\mathbf{r}, t) = f(r_1, r_2, t) = f_1(r_1) f_2(r_2) f_3(t)$. Hence, it is reasonable that in the following we only consider 1D spatial fields that can be easily extended to higher dimensional spatial/temporal fields.

To be able to compute the field with computers efficiently we consider periodic fields that yield a discrete spectrum. The one sided filter length is denoted as $K \in \mathbb{N}$ which yields $2K+1$ frequency components after filtering. To ensure real-valued fields, the spectral coefficients $c_k \in \mathbb{C}$ have to be conjugate symmetric, i.e., $c_k = c^*_{-k}$. First of all we assume 1D fields with length $L \in \mathbb{R}^+$.

2.4.1 Sampling on a Regular Grid

In the case of regular sampling we have I points regular distributed on the interval $[0, L]$ (we assume that the distance between the points is $\frac{L}{I}$ and therefore the first point is at 0 and the last one at $L - \frac{L}{I}$). This yields a sampling interval of $T = \frac{L}{I}$ and a spatial sampling frequency of $f_\mathrm{s} = \frac{I}{L}$. Therefore, we can describe this field in the frequency-domain if the maximum frequency component is $\frac{f_\mathrm{s}}{2}$, which is reasonable since we are considering low-pass fields, i.e., $f_\mathrm{c} \ll \frac{f_\mathrm{s}}{2}$, where f_c denotes the cut-off frequency of the low-pass filter.

The field at sampling point i reads

$$f_i = \sum_{k=-K}^{K} c_k e^{\iota 2\pi \frac{k}{I} i} ,$$

where $K \le \frac{I}{2} - 1$ and we assume that I is even. Moreover $\iota = \sqrt{-1}$ denotes the imaginary unit. When the frequency coefficients are i.i.d. complex Gaussian, i.e., $c_k \sim \mathcal{CN}(0, \sigma^2)$ than we obtain a low-pass field with maximum normalized frequency (cut-off frequency) $\theta_\mathrm{c} = \frac{K}{I}$. This frequency is less than $\frac{1}{2}$ and the normalization is the sampling frequency f_s. The minimum frequency equals $\frac{1}{I}$ which equals one period in the interval $[0, L]$.

In the next step we introduce oversampling. The oversampling factor a has to fulfill $0 < a \le 1$ and $\frac{1}{a} \in \mathbb{N}$. If $a = 1$ we have no oversampling. In total we have $\frac{I}{a}$ sampling points within $[0, \tilde{L}]$ and a sampling frequency of $\tilde{f}_s = \frac{f_s}{a}$. The field is given by

$$f_i = \sum_{k=-K}^{K} c_k e^{\iota 2\pi \frac{ak}{I} i} .$$

To be consistent with the previous result, we only consider an interval of length $L = a\tilde{L}$ in which we have I sampling points. Again, for random frequency coefficients c_k and $K \leq \frac{I}{2a} - 1$ (we assume that $\frac{I}{2a} \in \mathbb{N}$) we obtain a low-pass field with normalized cut-off frequency $\tilde{\theta}_c = \frac{aK}{I}$ (normalized with $f_s = \frac{I}{L}$). It is seen that the minimum cut-off frequency with oversampling is $\frac{a}{I}$. Hence, oversampling allows smaller frequencies as with normal sampling with a scaling determined by a.

2.4.2 Irregular Sampling

In this section we consider a continuous 1D spatial field which is sampled irregularly. The interval is again $[0, L]$ and the sampling positions are denoted as r_i. To obtain similar results we normalize the frequency with $f_{\text{ref}} = \frac{1}{r_{\text{exp}}}$ where r_{exp} denotes the expected distance between two sampling points. If the sampling positions are distributed randomly regarding a uniform distribution (the first point lies at 0 and the last one at L) we have $r_{\text{exp}} = \frac{L}{I}$ and hence $f_{\text{ref}} = \frac{I}{L}$, which equals the sampling frequency for regular sampling. Thus we have the same normalization. The field on sampling position r_i is defined as

$$f(r_i) = \sum_{k=-\infty}^{\infty} c_k e^{\iota 2\pi \frac{k}{L} r_i} .$$

Due to low-pass character we assume that all $c_k = 0$ when $|k| > K$ and hence,

$$f(r_i) = \sum_{k=-K}^{K} c_k e^{\iota 2\pi \frac{k}{L} r_i} ,$$

the coefficients fulfill the same properties as for regular sampling and therefore we obtain a low-pass field with absolute cut-off frequency $f'_c = \frac{K}{L}$. A normalization with f_{ref} yields $\theta'_c = \frac{K r_{\text{ref}}}{L}$.

Again, we are using the concept of oversampling which allows to generate low-pass fields with arbitrary low-frequencies. The oversampling factor a has to fulfill $0 < a \leq 1$ and $a \in \mathbb{R}$. The field at the sampling positions is obtained as

$$f(r_i) = \sum_{k=-K}^{K} c_k e^{\iota 2\pi \frac{ak}{L} r_i} .$$

The normalized (again with respect to f_{ref}) cut-off frequency is $\tilde{\theta}'_c = \frac{aK r_{\text{ref}}}{L}$. The minimum achievable frequency is $\frac{a r_{\text{ref}}}{L}$ and determined by the oversampling factor a.

2.4.3 Construction of a Low-Pass Field

To construct a spatial low-pass field efficiently in matrix/vector form we first stack the frequency coefficients into a vector $\mathbf{c} = (c_{-K} \dots c_{-1}\, c_0\, c_1 \dots c_K)^T$ where c_0 defines the DC component. The coefficients are generated randomly for each scenario regarding the aforementioned rules. Then the sampled exponential functions are placed in a matrix $\mathbf{U}_1$ where $[\mathbf{U}_1]_{ik} = e^{\iota 2\pi \frac{ak}{I-1} i}$ (for regular sampling) and $[\mathbf{U}_1]_{ik} = e^{\iota 2\pi \frac{ak}{L} r_i}$ (for irregular sampling). The one in the index denotes that the sampling is in the first spatial direction. The spatial field is now simply generated as

$$\mathbf{f} = \mathbf{U}_1 \mathbf{c}\,.$$

A two-dimensional (2D) spatial field can be generated by combining the exponential functions of both directions (r_1 and r_2) in a way that we obtain all possible combinations ($(2K+1)^2$ in total). Hence, we construct a coefficient matrix $\mathbf{C}$ of size $(2K+1)\times(2K+1)$. To obtain a real valued field the coefficients have conjugate symmetry regarding the center, i.e., $c_{kj} = c^*_{(-k)(-j)}$. Then we generate the sampling matrices for both dimensions separately, i.e. $\mathbf{U}_1$ and $\mathbf{U}_2$. Finally those matrices are combined as $\mathbf{U} = (\mathbf{U}_1 \otimes \mathbf{1}^T) \circ (\mathbf{1}^T \otimes \mathbf{U}_2)$, where $\circ$ denotes the direct product. To that end, the field computes as

$$\mathbf{f} = \mathbf{U}\mathbf{c}\,, \tag{2.8}$$

where we define $\mathbf{c} = \mathrm{vec}\{\mathbf{C}\}$. It is seen that $\mathbf{U}$ contains the sampled Fourier basis, i.e., the columns are the basis vectors sampled at the corresponding positions.

This procedure can be extended to generate higher dimensional fields (fields that for instance include also the temporal dimension).

2.4.4 Correlation Between Two Points

The correlation between two sampled field points is given by

$$\mathrm{E}\{f(r_1)f(r_2)\} = \mathrm{E}\left\{\sum_{k=-K}^{K} c_k e^{\iota 2\pi \frac{ak}{L} r_1} \sum_{k=-K}^{K} c_k e^{\iota 2\pi \frac{ak}{L} r_2}\right\}$$

since the coefficients are distributed independently with the constraint that $c_k = c^*_{-k}$ we have $\mathrm{E}\{c_k c_l\} = \delta_{k+l} P_{c_k}$ with $P_{c_k} = \mathrm{E}\{c_k c_k^*\}$. This yields

$$\begin{aligned}\mathrm{E}\{f(r_1)f(r_2)\} &= \sum_{k=-K}^{K} P_{c_k} e^{\iota 2\pi \frac{ak}{L}(r_1 - r_2)} \\ &= P_{c_0} + \sum_{k=1}^{K} 2P_{c_k} \cos\left(2\pi \frac{ak}{L}(r_1 - r_2)\right) \\ &= P_{c_0} + \sum_{k=1}^{K} 2P_{c_k} \cos\left(2\pi \frac{ak}{L}\Delta r\right)\,.\end{aligned}$$

If P_{c_k} is constant, in particular $P_{c_k} = P_c$ for all k we obtain

$$\mathrm{E}\{f(r_1)f(r_2)\} = P_c + P_c \sum_{k=1}^{K} 2\cos\left(2\pi \frac{ak}{L}\Delta r\right)\,,$$

which is a Dirichlet kernel. The power of any sampling point is given by setting Δr zero and get $(2K+1)P_c$.

For the case when $a \ll 1$ we can assume that the field is not periodic. For that purpose we have to consider the discrete-time Fourier transform to obtain the field values

$$\mathrm{E}\{f(r_1)f(r_2)\} = \mathrm{E}\left\{\int_{-f_\mathrm{c}'}^{f_\mathrm{c}'} C(f)e^{\iota 2\pi f r_1} df \int_{-f_\mathrm{c}'}^{f_\mathrm{c}'} C(f)e^{\iota 2\pi f r_2} df\right\} ,$$

where $f_c' = \frac{\theta_c'}{2\pi}$ denotes the cut-off frequency. Since the spectrum is conjugate symmetric and also distributed independently we have $\mathrm{E}\{C(f_i)C(f_j)\} = \delta(f_i - f_j)S(f_i)$ where $S(f_i)$ denotes the power spectral density [69] at f_i and hence,

$$\mathrm{E}\{f(r_1)f(r_2)\} = \int_{-f_\mathrm{c}'}^{f_\mathrm{c}'} S(f)e^{\iota 2\pi f(r_1 - r_2)} df \,.$$

Finally we assume that the power spectral density is constant, i.e., $S(f) = P_S$ and therefore we obtain

$$\mathrm{E}\{f(r_1)f(r_2)\} = 2\theta_\mathrm{c}' P_S \operatorname{sinc}(2\theta_\mathrm{c}'\Delta r) = 2\frac{aK}{L} P_S \operatorname{sinc}\left(2\frac{aK}{L}\Delta r\right) .$$

with $\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$. The factor $\frac{a}{L}$ (= sampling frequency) between the powers of the two cases comes from sampling in the frequency spectrum. For the 2D case the correlation in each direction is computed separately (due to orthogonality) and then multiplied.

3

Distributed Dynamic Averaging

THIS chapter deals with distributed dynamic averaging, the first major contribution of this thesis. We start with reformulating CP such that it can track time-varying signals. For a further performance analysis, we derive a state space representation of dynamic CP and compute the corresponding transfer function. One important key ingredient is a novel formulation of the incidence matrix that allows to establish the linear state space model. Based on the state space model, we will study the theoretical behavior of dynamic CP, in particular we provide a theorem that states stability. Moreover, the linear state space model allows to analyze the performance in the frequency domain through a multiple-input and multiple-output transfer function. The extension of dynamic CP by adding linear filters at each node is natural in this context and done subsequently.

The next part considers the state space model and frequency domain analysis of dynamic AC, which allows us on the one hand to study the weight design in dynamic scenarios and on the other hand to compare the transfer characteristics of dynamic AC and dynamic CP.

Furthermore, the practical implementational issue of medium access of (dynamic) CP is discussed in this chapter. To that end, we consider a simple ALOHA like protocol and derive theoretical expressions for the collision probabilities in WSN with random geometric graphs. Then we also study the performance of our protocol with CP in WSN.

We apply distributed averaging to the problem of distributed field reconstruction. To this end, we show how to adapt CP to reconstruct spatial fields in WSN. After the computation, each node possesses a sufficient accurate estimation of the entire field that was sampled by the individual sensors.

This chapter is finalized with numerical results that verify the theoretical findings and provide insights into the convergence behavior of (dynamic) CP and AC.

3.1 Static Consensus Propagation

In this section we briefly provide the basics of static CP [7] and [8] (more details can be found in Section 2.2) and we present a new theoretical aspect of the convergence. Moreover we show how to apply the convergence conditions of Gaussian Belief Propagation (GaBP) to CP.

We consider that our sensor network is represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes/sensors and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The total number of nodes is denoted by $I = |\mathcal{V}|$ and the number of edges is $E = |\mathcal{E}|$. Two nodes i and j are connected if $(i,j) \in \mathcal{E}$. The neighborhood of node i is defined as $\mathcal{N}(i) = \{j \mid (i,j) \in \mathcal{E}\}$ and the degree is $d_i = |\mathcal{N}(i)|$. The adjacency matrix $\mathbf{A}$ describes the connectivity pattern of a graph, i.e., $[\mathbf{A}]_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and zero otherwise. The graph Laplacian $\mathbf{L}$ can be obtained through the adjacency matrix as $\mathbf{L} = \mathrm{diag}\{\mathbf{A}\mathbf{1}\} - \mathbf{A}$. More details regarding graph theory are presented in Section 2.1.1.

In static averaging it is the goal to compute $\bar{s} = \frac{1}{I} \sum_i s_i$ where s_i denotes the measurement at sensor i. Assuming sensor networks, the measurements are distributed, since each sensor knows only its own measurement. As already discussed in the previous chapter, CP is based on solving the following optimization in a distributed manner:

$$\bar{s} = \arg\min_{\theta} \|\mathbf{1}\theta - \mathbf{s}\|_2^2 \,.$$

Taking the the network structure into account, the optimization problem reads,

$$\hat{\mathbf{s}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \mathbf{s}\|_2^2 + \beta \sum_{(i,j)\in\mathcal{E}} q_{ij}(\theta_i - \theta_j)^2 \,, \tag{3.1}$$

where $\beta > 0$ and $q_{ij} > 0$ (for all $(i,j) \in \mathcal{E}$) represent constants that have an impact on the precision of the result and on the convergence speed. In particular, in [7] it is shown that CP will converge to the true mean, as $\beta \to \infty$ (which is impossible in practice). Therefore, it is reasonable to introduce a bound on the MSE after CP has converged for a given β and q_{ij}:

Proposition 1. *Let* $\mathbf{\Gamma}$ *be the* $I \times I$ *matrix defined by* $\boldsymbol{\theta}^T \mathbf{\Gamma} \boldsymbol{\theta} = \sum_{\forall (i,j)\in\mathcal{E}} q_{ij}(\theta_i - \theta_j)^2$ *and let* λ_i, $i = 1, \dots, I$, *denote the sorted eigenvalues of* $\mathbf{\Gamma}$. *For* $\beta > 0$ *the error* $\epsilon^2 \triangleq \|\bar{\mathbf{s}} - \hat{\mathbf{s}}\|^2 / \|\mathbf{s}\|^2$ *can be bounded as* $\epsilon^2 \leq \sum_{i=2}^{I} \frac{1}{(1+\beta\lambda_i)^2}$.

The proof is provided in Appendix 3.A.2. The bound of the proposition can be relaxed to $\epsilon^2 \leq \frac{I-1}{(1+\beta\lambda_2)^2}$. For the case when $q_{ij} = 1$ for all $(i,j) \in \mathcal{E}$ we have $\mathbf{\Gamma} = \mathbf{L}$ and further $\lambda_2 \geq \frac{1}{2E\,\mathrm{diam}\{\mathcal{G}\}}$ (here, $\mathrm{diam}\{\mathcal{G}\}$ is the diameter of $\mathcal{G}$, see [70] for details). The error bound can thus be further simplified as $\epsilon^2 \leq \frac{I-1}{(1+\frac{\beta}{2E\,\mathrm{diam}\{\mathcal{G}\}})^2}$.

The unconstrained convex optimization problem (3.1) can be easily transferred in a form such that it can be represented via Gaussian Markov random field [71] and hence the average can be inferred through Gaussian belief propagation. Since the structure of the Gaussian Markov random field is the same that of the network graph, the necessary message exchanges happen on the available wireless communication links. The Gaussian structure allows that only the mean μ and the precision κ (inverse variance) of the corresponding distribution need to be transmitted.

It is straightforward to derive the message update equations of CP. The equations for synchronous processing at iteration k are:

$$\kappa_{i\to j}[k] = \frac{1 + \sum\limits_{v\in\mathcal{N}(i)\backslash j} \kappa_{v\to i}[k-1]}{1 + \frac{1}{\beta q_{ij}}\left(1 + \sum\limits_{v\in\mathcal{N}(i)\backslash j} \kappa_{v\to i}[k-1]\right)}, \tag{3.2}$$

$$\mu_{i\to j}[k] = \frac{s_i + \sum\limits_{v\in\mathcal{N}(i)\backslash j} \mu_{v\to i}[k-1]\kappa_{v\to i}[k-1]}{1 + \sum\limits_{v\in\mathcal{N}(i)\backslash j} \kappa_{v\to i}[k-1]}. \tag{3.3}$$

Here, i denotes the source node and j the destination node.

The estimate of $\bar{s}$ at node i in iteration k is given by,

$$\hat{\bar{s}}_i[k] = \frac{s_i + \sum\limits_{v\in\mathcal{N}(i)} \mu_{v\to i}[k-1]\kappa_{v\to i}[k-1]}{1 + \sum\limits_{v\in\mathcal{N}(i)} \kappa_{v\to i}[k-1]}. \tag{3.4}$$

These update equation can also be processed asynchronously, which follows also from GaBP, see Section 2.2.1 and [7] for more details.

In the case of synchronous processing, it is seen in (3.2) and (3.3) that for each neighboring node a different message is generated and for that purpose $|\mathcal{N}(i)|$ individual messages have to be transmitted in each iteration by node i. In Section 3.4.1 we present a method such that only one message needs to be transmitted in a broadcast fashion to reduce the required power.

As already discussed, CP is an application of GaBP and therefore the convergence conditions directly apply. In particular, in [11] they show that a sufficient condition for the convergence of GaBP is given by the diagonal dominance of the information matrix of the underlying problem. A detailed discussion can be found in [11]. Using the results from [11] we can state the following proposition:

Proposition 2. *Consensus Propagation according to* (3.1) *always satisfies the condition for diagonal dominance [11].*

The proof is provided in the Appendix 3.A.1.

Besides the diagonal dominance, the authors of [7] provide a convergence proof, which is tailored for the update equations of CP. Moreover they derive a bound on the convergence speed for random regular graphs, but the bound is too loose and hence impractical.

3.2 Dynamic Consensus Propagation

Consensus Propagation (CP) was initially proposed for static averaging. In this chapter we propose a dynamic version of CP and prove its stability.

Inspection of the structure of the CP updates (3.2) and (3.3) reveals that the μ-messages and the estimates are linear functions of the measurements and of the μ-messages of the neighbors. If the κ-messages are small, the measurements and the μ-messages from the neighbors have almost the same influence. In the course of the CP iterations the κ-messages increase and will converge. This decreases the influence of the measurements and therefore the estimate of the average is mostly defined by the μ-message, which contain the information of all other nodes. The idea with dynamic CP now is to keep the magnitude of the κ-messages moderate (β has to be small) and replace the static with the dynamic measurements. Hence, the μ-messages (which also represent the past) do not dominate, but it is still challenging to find the optimum tradeoff between tracking ability and the decreased accuracy because of small values of β (see Proposition 1). In summary, the update equations (3.2) of the κ-messages remain unchanged and the μ-messages and the local estimates are computed as:

$$\mu_{i\to j}[n] = \frac{s_i[n] + \sum\limits_{v\in\mathcal{N}(i)\setminus j} \mu_{v\to i}[n-1]\kappa_{v\to i}[n-1]}{1 + \sum\limits_{v\in\mathcal{N}(i)\setminus j} \kappa_{v\to i}[n-1]}, \tag{3.5}$$

$$\hat{s}_i[n] = \frac{s_i[n] + \sum\limits_{v\in\mathcal{N}(i)} \mu_{v\to i}[n-1]\kappa_{v\to i}[n-1]}{1 + \sum\limits_{v\in\mathcal{N}(i)} \kappa_{v\to i}[n-1]}. \tag{3.6}$$

If the measurements $s_i[n]$ vary fast over time, it is also possible to iterate these updates $M \geq 1$ times between two measurement instants. To this end, we substitute $s_i[n]$ in (3.5) and (3.6) with $s_i[\lfloor n/M \rfloor]$. If we let M tend to infinity we end up with concatenated static CP instances.

Even though we only considered synchronous dynamic CP so far, it is also possible to apply it in an asynchronous manner. Asynchronous processing is preferable in real WSN applications (see Section 3.4 for more details), but it should be ensured that the average update frequency is the same for all edges. But for simplicity we will only consider the synchronous version in the following.

3.2.1 Incidence Matrix

The incidence matrix of a graph describes the relation between nodes and edges. In the case of message-passing algorithms on graphs with undirected edges, each undirected edge is replaced with two oppositely directed edges, thereby doubling the number of edges. Correspondingly we define two different incidence matrices $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{B}}'$, for the outgoing edges and for the incoming edges, respectively. The elements are defined as $b_{il} = 1$ if and only if the directed edge l starts at node i and $b'_{il} = 1$ if edge l ends at node i. Both incidence matrices have size $I \times E_\text{d}$ where $E_\text{d} = 2|\mathcal{E}|$ and denotes the number of directed edges. Since to each incoming edge there is a corresponding outgoing edge, the two incidence matrices can be shown to be related as $\tilde{\mathbf{B}} = \tilde{\mathbf{B}}'\mathbf{P}$ with $\mathbf{P}$ being a symmetric permutation matrix. The incidence matrix is related to the adjacency and Laplacian matrix as: $\mathbf{A} = \tilde{\mathbf{B}}\mathbf{P}\tilde{\mathbf{B}}^T$ and $\mathbf{L} = \tilde{\mathbf{B}}\tilde{\mathbf{B}}^T - \tilde{\mathbf{B}}\mathbf{P}\tilde{\mathbf{B}}^T$.

A more commonly used definition of the incidence matrix which is denoted in this work as $\mathbf{B}$ contains $+1$ for outgoing edges and -1 for incoming edges.The graph Laplacian can be obtained from this incidence matrix as $\mathbf{L} = \mathbf{B}\mathbf{B}^T$. In the case of undirected graphs the sign is irrelevant and therefore is sometimes discarded. However, in our context the definition of the two incidence matrices $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{B}}'$ is more flexible.

3.2.2 State Space Model

To study the properties of dynamic CP, we formulate and analyze a corresponding state space model in which the μ-messages constitute the states. For the derivation of the state space model it is necessary to express the update equations (3.5) and (3.6) in vector form. For that purpose we define $\boldsymbol{\mu}[n]$ and $\boldsymbol{\kappa}[n]$ as the $E_{\mathrm{d}} \times 1$ stacked message vectors which are computed as described subsequently and $\hat{\mathbf{s}}[n]$ is the vector of the time-varying estimates of the average. Since each node sends messages to all its neighbors, it is convenient to use the incidence matrix $\tilde{\mathbf{B}}$ (and corresponding permutation matrix $\mathbf{P}$) defined in Section 3.2.1. We start with the κ-messages where (3.2) (for simplicity we assume that $q_{ij} = 1, \forall i, j$) can be reformulated as

$$\boldsymbol{\kappa}[n] = \mathrm{diag}^{-1}\left\{\mathbf{1} + \frac{1}{\beta}\left(\mathbf{1} + \mathbf{H}\boldsymbol{\kappa}[n-1]\right)\right\}\left(\mathbf{1} + \mathbf{H}\boldsymbol{\kappa}[n-1]\right) ,$$

where $\mathrm{diag}\{\cdot\}$ denotes a diagonal matrix containing the elements of the argument vector and the auxiliary matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \mathbf{P}\big(\tilde{\mathbf{B}}^T\tilde{\mathbf{B}} - \mathbf{I}\big).$$

The element $h_{kl} = 1$ only if edge k ends at the node where edge l starts. With the definitions of $\mathbf{P}$ and $\tilde{\mathbf{B}}$ from Section 3.2.1 the μ-message update (3.5) in vector form reads

$$\boldsymbol{\mu}[n] = \mathrm{diag}^{-1}\{\mathbf{1} + \mathbf{H}\boldsymbol{\kappa}[n-1]\}\,\Big(\mathbf{P}\tilde{\mathbf{B}}^T\mathbf{s}[n] + \mathbf{H}\,\mathrm{diag}\{\boldsymbol{\kappa}[n-1]\}\boldsymbol{\mu}[n-1]\Big) , \tag{3.7}$$

and depends on the κ-messages.

Using the κ- and μ-vectors the computation of the estimates $\hat{\mathbf{s}}[n]$ (cf. (3.6)) can be rewritten as

$$\hat{\mathbf{s}}[n] = \mathrm{diag}^{-1}\{\mathbf{1} + \tilde{\mathbf{B}}\boldsymbol{\kappa}[n-1]\}\big(\mathbf{s}[n] + \tilde{\mathbf{B}}\,\mathrm{diag}\{\boldsymbol{\kappa}[n-1]\}\boldsymbol{\mu}[n-1]\big). \tag{3.8}$$

Our goal is to formulate a linear multiple-input multiple-output state space model, where the measurement vector $\mathbf{s}[n]$ is the input and $\hat{\mathbf{s}}[n]$ is the output. Unfortunately, all update equations depend nonlinearly on the κ-messages. Hence, incorporating the κ-messages into the state vector does not result in a linear system.

A remedy to this problem is to assume that the κ-messages have already converged, i.e., $\kappa = \kappa[n]$ with $n \gg 1$, when setting up the state space model (the convergence proof of the κ-messages is provided in [8]). This is possible since the κ-messages only depend on themselves and not on the measurements nor the μ-messages. Hence, in practice they can even be computed distributedly before the averaging starts. Additionally, since we use small values of β with dynamic CP, the κ-messages converge after a few iterations (cf. [36]). A method to accelerate the κ convergence is to initialize them with the upper bound provided in Appendix 3.A.3. It follows that no κ-messages need to be exchanged during the dynamic CP iterations. For fixed (converged) κ-messages and using (3.7), (3.8), the state space model for dynamic CP is obtained as

$$\boldsymbol{\mu}[n] = \boldsymbol{\Phi}\boldsymbol{\mu}[n-1] + \boldsymbol{\Gamma}\mathbf{s}[n] \tag{3.9a}$$

$$\hat{\mathbf{s}}[n] = \mathbf{C}\boldsymbol{\mu}[n-1] + \mathbf{D}\mathbf{s}[n]\,. \tag{3.9b}$$

The matrices $\mathbf{\Phi}$ ($E_\mathrm{d} \times E_\mathrm{d}$), $\mathbf{\Gamma}$ ($E_\mathrm{d} \times I$), $\mathbf{C}$ ($I \times E_\mathrm{d}$), and $\mathbf{D}$ ($I \times I$) can be obtained from (3.7) and (3.8):

$$\begin{aligned}
\mathbf{\Phi} &= \mathrm{diag}^{-1}\{\mathbf{1} + \mathbf{H}\boldsymbol{\kappa}\}\,\mathbf{H}\,\mathrm{diag}\,\{\boldsymbol{\kappa}\}\ , \\
\mathbf{\Gamma} &= \mathrm{diag}^{-1}\{\mathbf{1} + \mathbf{H}\boldsymbol{\kappa}\}\,\mathbf{P}\tilde{\mathbf{B}}^T\,, \\
\mathbf{C} &= \mathrm{diag}^{-1}\Big\{\mathbf{1} + \tilde{\mathbf{B}}\boldsymbol{\kappa}\Big\}\,\tilde{\mathbf{B}}\,\mathrm{diag}\,\{\boldsymbol{\kappa}\}\,, \\
\mathbf{D} &= \mathrm{diag}^{-1}\Big\{\mathbf{1} + \tilde{\mathbf{B}}\boldsymbol{\kappa}\Big\}\,.
\end{aligned} \tag{3.10}$$

The matrix $\mathbf{\Phi}$ denotes the state-transition matrix between the μ states. The matrix $\mathbf{\Gamma}$ projects all the measurements of the sensors to the messages to the neighbors. The $\mathbf{C}$ and $\mathbf{D}$ matrices generate the estimates from the messages and the input. The analysis of the state space model in the next section uses the structure of $\mathbf{\Phi}$ in (3.10) to show stability of dynamic CP.

3.2.3 Stability and Dynamic Behavior of Dynamic CP

The stability of dynamic CP can be argued with the help of the convergence study of static CP, since it is shown in [7] that for any initial μ-messages they converge when the κ-messages have already converged. Beyond that, the local estimates converge to the true mean, even in an asynchronous setting.

Based on the state space model asymptotic stability [72] of dynamic CP can be shown. Asymptotic stability means that if the system has no input, any possible inner states will converge to zero as time goes to infinity, i.e., $\lim_{n\to\infty} \boldsymbol{\mu}_n = \mathbf{0}$.

Theorem 1. *For fixed non-negative $\boldsymbol{\kappa}$, the CP state space formulation* (3.9) *is asymptotically stable for any graph $\mathcal{G}$.*

Proof. According to [72], a discrete-time system is asymptotically stable if and only if $|\lambda_i(\mathbf{\Phi})| < 1$, where $\lambda_i(\mathbf{\Phi})$ denotes the eigenvalues of $\mathbf{\Phi}$. The elements of $\mathbf{\Phi}$ can be shown to be given by (cf. (3.10))

$$\phi_{ij} = [\mathbf{\Phi}]_{ij} = \frac{h_{ij}\kappa_j}{1 + \sum_l h_{il}\kappa_l}\,.$$

The definition of the 0/1-valued incidence matrix $\mathbf{B}$ implies that the matrix $\tilde{\mathbf{B}}^T\tilde{\mathbf{B}}$ has non-negative elements and all its diagonal elements equal 1. Hence, $\tilde{\mathbf{B}}^T\tilde{\mathbf{B}} - \mathbf{I}$ has non-negative elements as well which in turn implies $h_{ij} \geq 0$ and further $\phi_{ij} \geq 0$. It follows that (cf. [73, Sections 2.3.1 and 2.3.2])

$$\|\mathbf{\Phi}\|_\infty \leq \max_i \sum_j |\phi_{ij}| = \max_i \sum_j \frac{h_{ij}\kappa_j}{1 + \sum_l h_{il}\kappa_l} < 1.$$

Since the spectral radius $\rho(\mathbf{\Phi}) = \max_i \{|\lambda_i(\mathbf{\Phi})|\}$ is upper bounded by any matrix norm, we have $|\lambda_i(\mathbf{\Phi})| \leq \rho(\mathbf{\Phi}) \leq \|\mathbf{\Phi}\|_\infty < 1$, thereby establishing asymptotic stability. □

From the fact that the system is asymptotically stable, it can be concluded that it is also bounded input bounded output stable (see [72]).

If we want to investigate the dynamic behavior, it is necessary to analyze the eigenvalues of the system matrix $\mathbf{\Phi}$. Smaller eigenvalues ensure that the system reacts faster to changes, which is in fact preferable. Hence, it is reasonable to bound the largest eigenvalues as already done in the proof of Theorem 1,

$$\max|\lambda_i| \leq \max_i \frac{\sum_j h_{ij}\kappa_j}{1+\sum_l h_{il}\kappa_l} = \frac{\max_i \sum_j h_{ij}\kappa_j}{1+\max_i \sum_l h_{il}\kappa_l},$$

where the term $\max_i \sum_j h_{ij}\kappa_j$ can be upper bounded using the fact that $\kappa_i \leq \beta$ (equality is established if the degree of the graph would tend to infinity) and the sum consists of at least $d_{\text{max}}-1$ elements (where d_{max} denotes the maximum degree of the given graph). Hence, we have $\max_i \sum_j h_{ij}\kappa_j \leq (d_{\text{max}}-1)\beta$ and

$$\max|\lambda_i| \leq \frac{(d_{\text{max}}-1)\beta}{1+(d_{\text{max}}-1)\beta},$$

which allows us conclude that a large β and graphs with high degree tend to decrease the convergence speed of the system. On the other hand, a small β imposes a larger error (cf. Proposition 1), therefore there is a trade off between accuracy and tracking ability.

3.2.4 Frequency Domain Analysis

The state space model in (3.9) allow us to apply system-theoretic analysis tools to study dynamic CP. In particular, the $I \times I$ multiple-input multiple-output (frequency-domain) transfer function matrix of the entire system can be expressed as [72]

$$\mathbf{G}(\theta) = \mathbf{C}\big(e^{\imath 2\pi\theta}\mathbf{I} - \mathbf{\Phi}\big)^{-1}\mathbf{\Gamma} + \mathbf{D}, \tag{3.11}$$

where $\theta \in \left(-\frac{1}{2}, \frac{1}{2}\right]$ is the normalized frequency. This transfer function matrix consists of I^2 individual transfer functions $g_{ij}(\theta) = [\mathbf{G}(\theta)]_{ij}$ where each function describes the frequency-domain dependence of the estimate $\hat{\bar{s}}_i[n]$ at node i on the measurement $s_j[n]$ at node j. Since the true average depends on the measurements as $\bar{s}[n] = \frac{1}{I}\mathbf{1}\mathbf{1}^T\mathbf{s}[n]$ the ideal (but non-realizable) transfer function is given by $\mathbf{G}_{\text{ideal}}(\theta) = \frac{1}{I}\mathbf{1}\mathbf{1}^T$. Such a transfer function cannot be achieved for graphs that are not fully connected, since the measurement need some time to propagate through the network and therefore at least a delay will occur. Nevertheless it is desirable to obtain a transfer function matrix which is close to $\mathbf{G}_{\text{ideal}}(\theta)$.

A simple global characterization of the amplitude transfer behavior is given by

$$A(\theta) = \frac{1}{I^2}\sum_{i=1}^{I}\sum_{j=1}^{I}\big|g_{ij}(\theta)\big|.$$

This quantity represents the average amplitude response of the system which ideally should be close to $\frac{1}{I}$. The global delay characteristics of the system can be described through the (average) group delay

$$\tau(\theta) = -\frac{1}{I^2}\sum_{i=1}^{I}\sum_{j=1}^{I}\frac{\mathrm{d}}{\mathrm{d}\theta}\arg\{g_{ij}(\theta)\}.$$

In the ideal case, each individual group delay is zero and hence $\tau(\theta)$ is zero as well. In practice this is unrealistic, however, it suffices that $\frac{\mathrm{d}}{\mathrm{d}\theta}\arg\{g_{ij}(\theta)\}$ is constant in the frequency band of the observed signal. Methods to mitigate undesired delays are discussed in the next session.

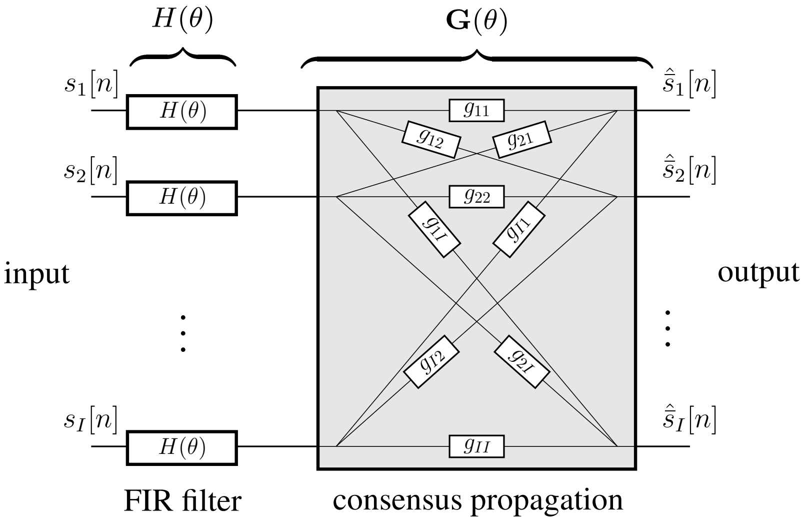


Figure 3.1: *Consensus propagation when the input is filtered.*

3.2.5 Linear Prediction and Equalization

In this section we provide two methods to adapt the performance of dynamic CP. The first method tries to mitigate the delay and the second method aims for a closer to optimum amplitude and phase behavior. For both methods we introduce an additional local filter with transfer function $H_i(\theta)$ at each node i which filters the input signal before dynamic CP is applied. For simplicity, we only consider identical FIR filters $H(\theta) = \sum_{l=0}^{L} h_l e^{-\iota 2\pi\theta n}$ for all nodes, where $L+1$ is the filter length and h_l are the filter coefficients. Hence, the transfer function of the filter plus dynamic CP is $\mathbf{F}(\theta) = \mathbf{G}(\theta)H(\theta)$. A block diagram of the entire system is seen in Fig. 3.1.

Since all filters are equal and linear, they can be either applied to the measured data (as it is shown in Fig. 3.1) or to the output of the system. We only consider the first case in the following. The filtered input $\tilde{\mathbf{s}}[n]$ of CP therefore equals

$$\tilde{\mathbf{s}}[n] = \big(\mathbf{s}[n]\ \mathbf{s}[n-1]\ \ldots\ \mathbf{s}[n-L]\big)\,\mathbf{h}\,,$$

with $\mathbf{h} = [h_0\, h_1\ \ldots\ h_L]^T$. As for dynamic CP, we can formulate a state space model of the concatenated system. To that end, the state vector $\boldsymbol{m}$ is augmented with the $L-1$ previous measurement vectors:

$$\tilde{\boldsymbol{\mu}}[n] = \big(\boldsymbol{\mu}^T[n]\ \mathbf{s}^T[n-1]\ \ldots\ \mathbf{s}^T[n-L]\big)^T\ .$$

The extended state space model then reads

$$\tilde{\boldsymbol{\mu}}[n+1] = \tilde{\mathbf{\Phi}}\tilde{\boldsymbol{\mu}}[n] + \tilde{\mathbf{\Gamma}}\mathbf{s}[n]\,, \tag{3.12a}$$

$$\hat{\tilde{\mathbf{s}}}[n+1] = \tilde{\mathbf{C}}\tilde{\boldsymbol{\mu}}[n] + \tilde{\mathbf{D}}\mathbf{s}[n]\,, \tag{3.12b}$$

with the matrices

$$\tilde{\boldsymbol{\Phi}} = \begin{array}{c} \left(\begin{array}{c:c:c} \boldsymbol{\Phi} & \multicolumn{2}{c}{\mathbf{h}'\otimes\boldsymbol{\Gamma}} \\ \hdashline \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hdashline \mathbf{0} & \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} & \mathbf{0} \end{array}\right) \begin{array}{l} \}\ E_\mathrm{d} \\ \}\ I \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\}\ (L-1)I \end{array} \\ \underbrace{\qquad}_{E_\mathrm{d}}\ \underbrace{\qquad\qquad}_{(L-1)I}\ \underbrace{\qquad}_{I} \end{array}$$

$$\tilde{\boldsymbol{\Gamma}} = \left(\begin{array}{c} h_0\boldsymbol{\Gamma} \\ \hdashline \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} \\ \hdashline \mathbf{0} \end{array}\right) \begin{array}{l} \}\ E_\mathrm{d} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\}\ (L-1)I \\ \}\ I \end{array}$$

$$\tilde{\mathbf{C}} = \left(\begin{array}{c:c} \mathbf{C} & \mathbf{h}'\otimes\boldsymbol{\Gamma} \end{array}\right), \qquad \underbrace{\quad}_{E_\mathrm{d}}\ \underbrace{\qquad\quad}_{LI}$$

$$\tilde{\mathbf{D}} = h_0\mathbf{D}\,,$$

with $\mathbf{h}' = (h_1\,h_2\,\ldots\,h_L)^T$. Again, the entire system is stable. This follows because dynamic CP is stable according to Theorem 1, any FIR filter is stable, and the concatenation of two stable systems is also stable.

Linear Prediction

To mitigate the delay inherent to dynamic CP, we suggest to use linear prediction with horizon v. This is feasible because in many practical applications the measurements smoothly change over time which can be modeled in terms of a low-pass process. To design the predictor the second-order statistics of the measurements have to be known. The input sequence of the ith predictor at time n is $s_i[n]\ldots s_i[n-L]$. Through the known correlations $\mathrm{E}\{s_i[n]s_i[n-k]\}$ and $\mathrm{E}\{s_i[n+v]s_i[n-k]\}$ for all $0 \le k \le L$ (we assume that the process is wide-sense stationary and hence the correlations are independent on the time n) it is possible to derive the filter coefficients through solving the Yule-Walker equations [74]. Simulation results for dynamic CP with linear prediction are shown in Section 3.6.3.

Optimal FIR Filter Design

Here, the goal is to design $\mathbf{h}$ such that the overall transfer function $\mathbf{F}(\theta)$ is close to $\frac{1}{I}\mathbf{1}\mathbf{1}^T$ apart from a delay τ_0. This leads to the objective function

$$f(\mathbf{h}) = \left\| \mathbf{F}(\theta) - \frac{1}{I}\mathbf{1}\mathbf{1}^T e^{\imath 2\pi\tau_0\theta} \right\| W(\theta)\,, \tag{3.13}$$

where $W(\theta)$ is a weight function whose range equals the interval $[0,1]$. The norm can be either the l_2 or the l_∞ norm. According to the results of [75], the objective function with both norms is convex. The l_2 optimization can be solved in closed form.

3.3 State Space Model for Dynamic Average Consensus

We next derive a state space model for dynamic AC. The state vector $\boldsymbol{\theta}[n]$ here contains the last measurement $\mathbf{s}[n-1]$ and also $\mathbf{x}[n]$, i.e., $\boldsymbol{\theta}[n] = \left(\mathbf{x}^T[n]\,\mathbf{s}^T[n-1]\right)^T$. The state space representation of (2.5) then reads

$$\begin{aligned}\boldsymbol{\theta}[n] &= \bar{\boldsymbol{\Phi}}\boldsymbol{\theta}[n-1] + \bar{\boldsymbol{\Gamma}}\mathbf{s}[n]\,,\\ \hat{\bar{\mathbf{s}}}[n] &= \bar{\mathbf{C}}\boldsymbol{\theta}[n-1] + \bar{\mathbf{D}}\mathbf{s}[n]\,,\end{aligned}$$

where the matrices $\bar{\boldsymbol{\Phi}}$, $\bar{\boldsymbol{\Gamma}}$, $\bar{\mathbf{C}}$ and $\bar{\mathbf{D}}$ are defined as

$$\bar{\boldsymbol{\Phi}} = \left(\begin{array}{c|c}\mathbf{W} & \mathbf{I}\\ \hline \mathbf{0} & \mathbf{0}\end{array}\right), \qquad \bar{\boldsymbol{\Gamma}} = \left(\begin{array}{c}\mathbf{I}\\ \hline \mathbf{I}\end{array}\right),$$

$$\bar{\mathbf{C}} = \left(\begin{array}{c|c}\mathbf{W} & \mathbf{I}\end{array}\right), \qquad \bar{\mathbf{D}} = \mathbf{I}\,.$$

As for CP we can use this state space model to obtain the transfer function of the entire system similarly to (3.11). For dynamic AC the transfer function can be simplified because the block structure of $\boldsymbol{\Gamma}$ allows to invert $\left(e^{\iota 2\pi\theta}\mathbf{I} - \boldsymbol{\Phi}\right)^{-1}$ analytically and yields

$$\mathbf{G}(\theta) = \left(1 - e^{-\iota 2\pi\theta}\right)\left(\mathbf{I} - \mathbf{W}e^{-\iota 2\pi\theta}\right)^{-1}\,.$$

We next bound the error between the transfer function and the ideal averaging function $\mathbf{G}_{\text{ideal}} = \frac{1}{I}\mathbf{1}\mathbf{1}^T$ where we use the eigenvalue decomposition $\mathbf{W} = \sum_i \lambda_i \mathbf{v}_i\mathbf{v}_i^T$ and assume that the eigenvalues λ_i are sorted. We obtain

$$\begin{aligned}\left\|\mathbf{G}(\theta) - \frac{1}{I}\mathbf{1}\mathbf{1}^T\right\|_2 &= \left\|\sum_i \frac{1 - e^{-\iota 2\pi\theta}}{1 - e^{-\iota 2\pi\theta}\lambda_i}\mathbf{v}_i\mathbf{v}_i^T - \frac{1}{I}\mathbf{1}\mathbf{1}^T\right\|_2\\ &= \left\|\frac{1 - e^{-\iota 2\pi\theta}}{1 - e^{-\iota 2\pi\theta}}\frac{1}{I}\mathbf{1}\mathbf{1}^T - \frac{1}{I}\mathbf{1}\mathbf{1}^T + \sum_{i\geq 2}\frac{1 - e^{-\iota 2\pi\theta}}{1 - e^{-\iota 2\pi\theta}\lambda_i}\mathbf{v}_i\mathbf{v}_i^T\right\|_2\\ &= \left\|\sum_{i\geq 2}\frac{1 - e^{-\iota 2\pi\theta}}{1 - e^{-\iota 2\pi\theta}\lambda_i}\mathbf{v}_i\mathbf{v}_i^T\right\|_2 \leq \sum_{i\geq 2}\left|\frac{1 - e^{-\iota 2\pi\theta}}{1 - e^{-\iota 2\pi\theta}\lambda_i}\right|\left\|\mathbf{v}_i\mathbf{v}_i^T\right\|_2\\ &= \sum_{i\geq 2}\left|\frac{1 - e^{-\iota 2\pi\theta}}{1 - e^{-\iota 2\pi\theta}\lambda_i}\right| = \sum_{i\geq 2}\frac{2|\sin\pi\theta|}{\sqrt{1 + \lambda_i^2 - 2\lambda_i\cos 2\pi\theta}} = \epsilon_{\text{dyn}}(\theta)\,.\end{aligned}$$

It is seen that the error becomes zero at $\theta = 0$, where the transfer function becomes zero, which is due to the fact that dynamic AC involves the first-order difference $\mathbf{s}[n]-\mathbf{s}[n-1]$ which is zero for constant measurements. But the left and right limit at zero is the average as $\lim_{\theta\to 0\pm}\mathbf{G}(\theta) = \frac{1}{I}\mathbf{1}\mathbf{1}^T$. It follows that we obtain almost the true average for small values of θ with dynamic AC. If we want to reduce the error for small frequencies the structure of $\epsilon_{\text{dyn}}(\theta)$ requires that the eigenvalues are close to -1 (i.e., the problem $\max\sum_{i\geq 2}(\lambda_i - 1)^2$ is convex and can be solved through standard solvers as CVX [76]), which we have included in our numerical simulations in Section 3.6.2. But the optimization towards -1 yields large errors at high frequencies. Hence, if the source signal contains high frequency components (e.g., a signal containing white noise) the performance is poor. But it also affects the transient phase, since it can be modeled as the reaction of the system to a step function (which contains high frequency components). Therefore, this approach has limited practical relevance. To summarize, the eigenvalues should be as small as possible for good transient performance, but they should not be too small to be able to follow faster signals. Numerical integration over the error with an application dependent cost function will return the optimum range for the eigenvalues.

3.4 Consensus Propagation in WSN

3.4.1 Broadcasting

As we have seen in Section 3.1, CP involves bilateral message exchange among neighbors. This is a major difference to AC, where a sensor transmits the same message to all its neighbors. In this section we show that such a broadcast protocol is also possible for CP without affecting the convergence behavior. The ability of transmitting the same message to all neighboring sensors enables broadcasting in wireless scenarios, thereby reducing the required transmit power and rendering CP more suitable for applications in WSN.

In the following we derive broadcast CP for static scenarios. A similar protocol can be developed for our proposed dynamic version. First, the partial sums in the message update equations (3.2) and (3.3) can be replaced with sums over all incoming messages minus the messages from the destination node:

$$\kappa_{i\to j}[k] = \frac{\big(1+\sum_{u\in\mathcal{N}(i)}\kappa_{u\to i}[k-1]\big) - \kappa_{j\to i}[k-1]}{1+\frac{1}{\beta}\Big(\big(1+\sum_{u\in\mathcal{N}(i)}\kappa_{u\to i}[k-1]\big) - \kappa_{j\to i}[k-1]\Big)}\,,$$

$$\mu_{i\to j}[k] = \frac{\big(s_i+\sum_{u\in\mathcal{N}(i)}\mu_{u\to i}[k-1]\kappa_{u\to i}[k-1]\big) - \mu_{j\to i}[k-1]\kappa_{j\to i}[k-1]}{\big(1+\sum_{u\in\mathcal{N}(i)}\kappa_{u\to i}[k-1]\big) - \kappa_{j\to i}[k-1]}\,.$$

These update equations can be separated into two steps, where the first reads

$$\tilde{\kappa}_i[k] \;=\; 1 \;+ \sum_{u\in\mathcal{N}(i)} \kappa_{u\to i}[k-1]\,, \tag{3.14a}$$

$$\tilde{\mu}_i[k] \;=\; s_i \;+ \sum_{u\in\mathcal{N}(i)} \mu_{u\to i}[k-1]\,\kappa_{u\to i}[k-1]\,. \tag{3.14b}$$

We call this step the pre-processing step and the auxiliary variables $\tilde{\kappa}_i[k]$ and $\tilde{\mu}_i[k]$ are broadcasted to all neighbors of node i. The auxiliary messages only depend on the $k-1$ messages but have the time index k, this choice is arbitrarily, but we assume a change in the time instance during each transmission step. In the second step each node that receives the messages applies the post processing step

$$\kappa_{i\to j}[k] = \frac{\tilde{\kappa}_i[k] - \kappa_{j\to i}[k-1]}{1+\frac{1}{\beta}\big(\tilde{\kappa}_i[k] - \kappa_{j\to i}[k-1]\big)}\,, \tag{3.15a}$$

$$\mu_{i\to j}[k] = \frac{\tilde{\mu}_i[k] - \mu_{j\to i}[k-1]\kappa_{j\to i}[k-1]}{\tilde{\kappa}_i[k] - \kappa_{j\to i}[k-1]}\,. \tag{3.15b}$$

Here, each node subtracts the previous message from the received messages in order to ensure that we have finally the sum over all messages except the previous (virtually) transmitted message. Hence, not only the

network model	E$\{\eta\}$
random r-regular graphs	r
scale-free graphs	2ξ
random geometric graphs	$c^2\pi$

Table 3.1: *Mean energy saving factor of broadcast CP for different network models.*

target node has to do the post-processing, also the transmitting node needs to do the post-processing step for all its neighbors, to be able to obtain the correct messages if it receives a subsequent message. Therefore, it is important that if a broadcast message is lost during transmission, the transmitting node has to detect such a failure. Finally, the estimate (3.4) at each node can be determined using the received messages as $\hat{\bar{s}}_i[k]=\tilde{\mu}_i[k]/\tilde{\kappa}_i[k]$.

In the remainder of this section we study the energy reduction achieved with broadcast CP. To this end, we assume a flooding schedule, where at each iteration all nodes exchange messages with all their neighbors. This means that in conventional CP node i computes and transmits $|\mathcal{N}(i)|$ individual message pairs and receives $|\mathcal{N}(i)|$ message pairs at each time instant; in total $2|\mathcal{E}|$ message pairs are computed and transmitted per iteration. In contrast to broadcast CP each node pre-processes only one message pair and transmits this message pair to all its neighbors, this amounts to I message-pairs in total. Hence, each node receives messages from all its neighbors and post-processes each of it. As already described, also the transmitting nodes need to do the post-processing step. Therefore, we can observe that the computational effort almost doubles, but precious transmission power is reduced by the ratio between the number of conventional messages and the number of broadcast messages, i.e.,

$$\eta = \frac{2|\mathcal{E}|}{I} = \frac{1}{I}\sum_{i=1}^{I} |\mathcal{N}(i)| \,. \tag{3.16}$$

Here we used that $|\mathcal{E}| = \frac{1}{2}\sum_{i=1}^{I} |\mathcal{N}(i)|$. Thus, the energy saving factor equals the average number of neighbors. Because for connected graphs the minimum number of neighbors is lower-bounded by one (but typically it is much higher), we can conclude that $\eta \geq 1$. In Table 3.1 we provide some values of the expected energy saving for the different graph topologies introduced in Section 2.1.2 (see [36] for the derivation).

In practice reduced wireless transmission and slightly increased computational complexity will decrease the total energy consumption of each sensor. Additionally due to broadcasting the medium usage is also reduced, which will be considered in more detail in the next section.

3.4.2 Medium Access of CP

In this section we discuss the medium access of distributed averaging algorithms and propose a practical implementation for CP. Medium access is often not considered when distributed algorithms are discussed. In particular, WSN demand low-complexity sensor devices with simple modulation and transmission techniques.

When a message is exchanged between two nodes, the medium is used for a certain time and the messages arrive with a delay at the destination. For that purpose a protocol is required such that the messages do not interfere in time or space. Collisions can be avoided if orthogonal medium access schemes are used, e.g., each

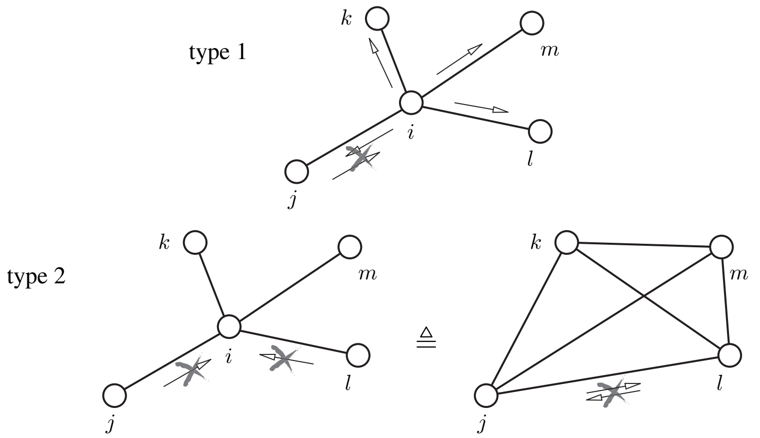


Figure 3.2: *Possible collision types involving node i.*

sensor transmits at a different frequency or in a different time slot. The latter requires synchronization between the nodes which is difficult to achieve ([64] considers time delays with AC). To sum up, orthogonal signaling requires a high level of (centralized) coordination which is not desired in WSN.

We therefore propose a simple medium access protocol for CP which is similar to ALOHA but avoids retransmissions. Our protocol requires only simple single carrier half-duplex transceivers with no multiuser detectors (that could distinguish between messages arriving at the same time). Hence, message collisions will occur, which we simply accept and assume that the affected messages are lost.

First of all we assume that each message exchange uses the medium within the communication radius r for a time interval of duration T_{m}. In the following we stick to graphs where neighboring nodes have distance at most r (which is a bit more general than random geometric graphs, since we make no assumptions regarding the node distribution). A collision will occur if one node starts the transmission at time t and a node which is at most two hops away starts the transmission in the interval $[t - T_{\mathrm{m}}, t + T_{\mathrm{m}}]$. We can distinguish between two types of collisions where messages are lost (see Fig. 3.2):

- If two neighboring nodes start their transmission within an interval of duration T_{m}, the half-duplex constraint implies that messages collide at both sensors and hence are lost (*type 1* collision).

- With a *type 2* collision, two neighboring nodes of a third node transmit within T_{m} and therefore the two messages collide at the receiver (third node) and cannot be resolved due to the lack of a multi-user detector.

In the following we will present two scheduling methods for the message transmission and derive the collision probabilities. The simpler method assumes that each node transmits according to a Poisson point process. Hence, the interval between two messages is exponentially distributed. On average each node should transmit once during the time-interval T (this is the time difference between two iterations of the equivalent synchronous processing of CP). Clearly, we have $T \gg T_{\mathrm{m}}$ and we obtain $\frac{1}{T - T_{\mathrm{m}}}$ as the parameter for the exponential distribution. The probability that a transmitted message collides with one of a different node is $p = \frac{2T_{\mathrm{m}}}{T}$. Since a node

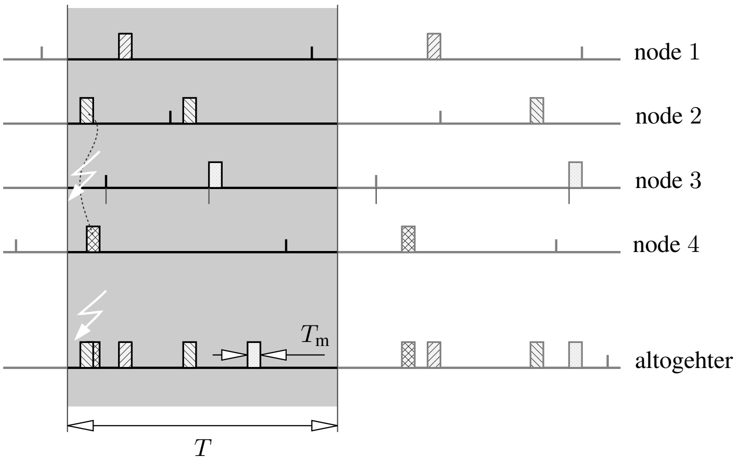


Figure 3.3: *Illustration of proposed ALOHA-like protocol: medium access in the vicinity of node* 1 *with neighbor set* $\mathcal{N}(1) = \{2, 3, 4\}$.

has d neighbors, the probability for at least one type 1 collision is given by

$$\mathrm{P}\{C_{\text{type 1}} > 0 | D = d\} = 1 - \left(1 - \frac{2T_\mathrm{m}}{T}\right)^d ,$$

where $C_{\text{type 1}}$ denotes the number of type 1 collisions. Since d Bernoulli trials (of a type 1 collision) yields a binomial distribution $\mathcal{B}(d, p)$, the expected number of type 1 collision equals

$$\mathrm{E}\{C_{\text{type 1}} | D = d\} = \frac{2T_\mathrm{m}}{T} d . \tag{3.17}$$

The derivation of the statistical properties of the type 2 collisions is little more involved. It is necessary to check all possible type 1 collisions between the the neighboring nodes. These collisions will not necessarily affect the neighbors, but it is impossible for the considered node to receive both messages at the same time. Therefore, we take all neighbors, construct a complete graph and pick one node and check for type 1 collisions. Then we remove the node and pick a different one and do the same procedure until one node is left and we are done. In total we checked for $\frac{d(d-1)}{2}$ type 1 collisions on the complete graph. Hence, we obtain for the probability that at least one type 2 collision occurs for a node with d neighbors

$$\mathrm{P}\{C_{\text{type 2}} > 0 | D = d\} \leq 1 - \left(1 - \frac{2T_\mathrm{m}}{T}\right)^{\frac{d(d-1)}{2}} ,$$

where $C_{\text{type 2}}$ denotes the number of type 2 collisions. If more than one type 2 collision occur, the events are not exclusive and therefore the expected number of collisions is only an upper bound. For the unrealistic case of independent type 2 collisions we obtain the overall distribution $\mathcal{B}\left(\frac{d(d-1)}{2}, p\right)$ which describes an upper bound for the number of type 2 collisions. Instead of the expected total number of collisions we are interested in the number of lost messages, i.e., k collisions affect $k+1$ messages for $k > 0$. With the relation $\mathrm{E}\{X + \mathbb{1}_{\{X \neq 0\}}\} = \mathrm{E}\{X\} + \mathrm{P}\{X > 0\}$ (where $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function, which is one when the condition is fulfilled and zero elsewhere) we obtain the bound

$$\mathrm{E}\{C_{\text{type 2}} | D = d\} \leq d(d-1)\frac{T_\mathrm{m}}{T} + \mathrm{P}\{C_{\text{type 2}} > 0 | D = d\} . \tag{3.18}$$

The total number of collisions is upper bounded as $C \le C_{\text{type 1}} + C_{\text{type 2}}$ because type 1 and type 2 collisions are not mutually exclusive and hence,

$$\mathrm{P}\{C > 0|D = d\} \le \mathrm{P}\{C_{\text{type 1}} > 0|D = d\} + \mathrm{P}\{C_{\text{type 2}} > 0|D = d\}\,,$$
$$\mathrm{E}\{C|D = d\} \le \mathrm{E}\{C_{\text{type 1}}|D = d\} + \mathrm{E}\{C_{\text{type 2}}|D = d\}\,.$$

A drawback of the protocol described in the preceding paragraphs is the fact that the nodes transmit at constant rate only on average, i.e., there is a non-zero probability that the gap between two messages of the same node becomes arbitrarily large. Therefore, we present a modification of this protocol where we assume that each sensor has an internal clock with tick T and necessarily transmits one (unsynchronized) message in each interval (k then is the clock counter). The timing within the interval is chosen randomly as $\Delta_i \sim \mathcal{U}\,(0, T - T_\mathrm{m})$. Hence, the message rate is the same as in the previous case, but the collision analysis is slightly different. Fig. 3.3 shows a schematic illustration of our transmission protocol. First of all the probability that two messages collide changes to $p = \frac{2T_\mathrm{m}}{T + T_\mathrm{m}}$ because the interval has to be extended to $[kT - T_\mathrm{m}, (k+1)T]$ to observe collisions also at the boundary. Hence, the number of Bernoulli trials becomes rational $n = d(1 + \frac{T_\mathrm{m}}{T})$. Since n cannot be rational for binomial distributions, we are going to average with the same ratio over binomial distribution of the two closest integers in the following results. The type 1 collision probability can be derived similarly as with the previous protocol with the result

$$P\{C'_{\text{type 1}} > 0|D = d\} = 1 - \left(1 - \frac{2T_\mathrm{m}}{T + T_\mathrm{m}}\right)^{d\left(1 + \frac{T_\mathrm{m}}{T}\right)}\,,$$

and the expected number of type 1 collisions is the same as in (3.17):

$$E\{C'_1|D = d\} = \frac{2T_\mathrm{m}}{T} d\,.$$

The distribution which describes the upper bound for type 2 collisions is again obtained by adding several binomial distributions and we have a distribution with $p = \frac{2T_\mathrm{m}}{T + T_\mathrm{m}}$ and $n = \frac{d(d-1)}{2}\left(1 + \frac{T_\mathrm{m}}{T}\right)$ (which is again not an integer and hence the averaging argument as described above has to be applied). The bound for the collision probability reads

$$P\{C'_{\text{type 2}} > 0|D = d\} = 1 - \left(1 - \frac{2T_\mathrm{m}}{T + T_\mathrm{m}}\right)^{\frac{d(d-1)}{2}\left(1 + \frac{T_\mathrm{m}}{T}\right)}\,,$$

and the bound on the expected number of collision is

$$E\{C'_{\text{type 2}}|D = d\} \le d(d-1)\frac{T_\mathrm{m}}{T} + P\{C'_{\text{type 2}} > 0|D = d\}\,,$$

which has the same form as (3.18), but since $P\{C'_{\text{type 2}} > 0|D = d\}$ is different, the values are not equal. Also the total conditional collision probability and the bound of the expected number of all collisions conditioned on the degree compute the same.

So far we have only results for the conditional statistics (given d). If there is knowledge available about the graph structure, i.e., the degree distribution is known, it is possible to derive the unconditional results as

$$\mathrm{P}\{C > 0\} \le \sum_d \mathrm{P}\{C > 0|D = d\}\ \mathrm{P}\{D = d\}\,,$$

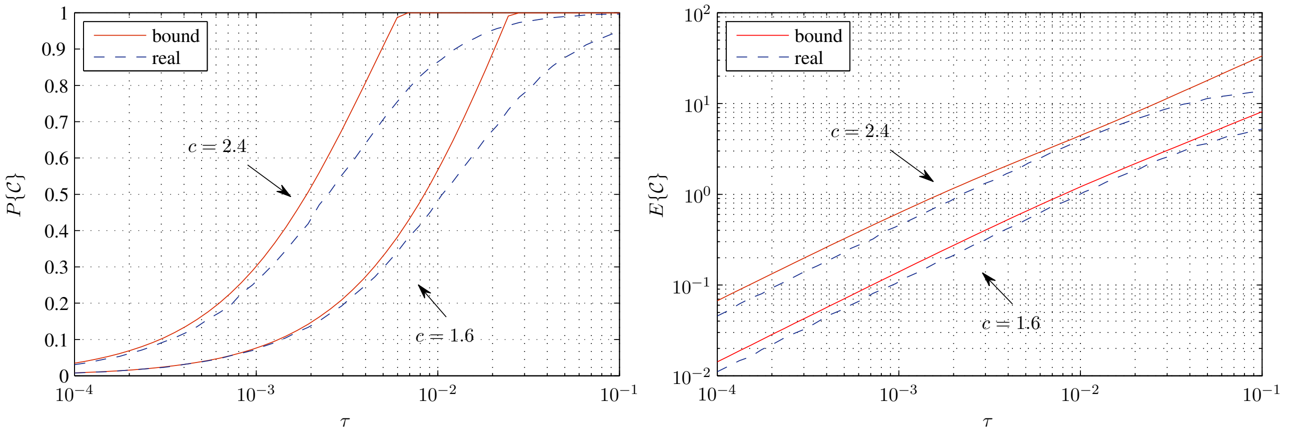


Figure 3.4: *Probability of collisions (left) and expected number of collisions (right) for different connectivities versus duty cycle* $\tau = \frac{T_m}{T-T_m}$ *for WSN with 100 nodes.*

$$\mathrm{E}\{C\} \leq \sum_{d} \left(\mathrm{E}\{C_1|D=d\} + \mathrm{E}\{C_2|D=d\}\right) \mathrm{P}\{D=d\} .$$

In the case of periodic random geometric graphs the degree has Poisson distribution (cf. 2.1.2). Considering bounded random geometric graphs, the gap between the theoretical and empirical results increases.

In Fig. 3.4 we have plotted the probability of collisions and also the number of collisions for our second proposed scheme applied on random geometric graphs with 100 nodes and different connectivity versus the duty cycle $\tau = \frac{T_\mathrm{m}}{T-T_\mathrm{m}}$. It is seen that the bound is quite close to the real implementation. In Section 3.6.1 and 3.6.2 we added our protocol with collisions in our numerical simulations and show the impact of collisions on the averaging performance.

3.5 Application: Field Reconstruction

Distributed averaging in WSN is an important tool to solve more sophisticated distributed inference problems. In particular, if there is an algorithm that involves a sum where each summand is provided by a different sensor node, it can be solved through distributed averaging. In this section we consider distributed reconstruction of static and even dynamic fields in WSN. For simplicity we assume only scalar spatial fields, but it is not difficult to extend our approach to vector fields. Our method has many potential practical applications, such as the estimation of a temperature or pollution field with low complexity sensors. Even though other distributed averaging algorithm can be used, we focus on CP to avoid synchronization issues. First of all we consider a subspace model to represent the spatial field efficiently.

3.5.1 Subspace Model

The observed field $f(\mathbf{r};t)$ is real-valued and time-varying and defined in the region $\mathcal{A}$ where the sensors are deployed. It is assumed that $f(\mathbf{r};t)$ lies in an L-dimensional spatial subspace $\mathrm{span}\{u_1(\mathbf{r}),\ldots,u_L(\mathbf{r})\}$, where $u_l(\mathbf{r})$, $l = 1,\ldots,L$, are linearly independent basis functions. We consider a finite-dimensional model, because of the smooth nature of many practical physical fields. Hence, the field can be written as

$$f(\mathbf{r};t) = \sum_{l=1}^{L} c_l(t)\, u_l(\mathbf{r})\,, \tag{3.19}$$

where $c_l(t)$ are the corresponding time-varying basis coefficients. The choice of the basis functions depend on the application, but L should be as small as possible to save energy (see Section 3.5.3). Other desirable properties of the basis are:

- the basis should be matched to the field under consideration in order to keep modeling errors as small as possible;
- for a suitable approximation of the LS estimator the set $\{u_1(\mathbf{r}), \ldots, u_L(\mathbf{r})\}$ should be orthonormal, i.e., $\int u_k(\mathbf{r})\, u_l(\mathbf{r})\, d\mathbf{r} = \delta_{kl}$ for all $k, l \in \{1, \ldots, L\}$,
- the basis functions should be robust to localization errors in the sense that $u_l(\mathbf{r} + \boldsymbol{\delta}) \approx u_l(\mathbf{r})$ for a small position error $\boldsymbol{\delta}$.

So far we considered already low-pass fields in Section 2.4.3. Here, we do not necessarily consider low-pass fields, but any field that can be constructed through a finite dimensional basis. In Section 2.4.3 we already showed how to extend a 1D field such that it can be applied for a 2D Euclidean space, i.e., through the spatial separability we have

$$u_l(r_1, r_2) = \tilde{u}_{k_1(l)}(r_1)\, \tilde{u}_{k_2(l)}(r_2)\,.$$

In the following we assume that the basis functions are known and hence the field can be fully determined by inferring the coefficients $c_l(t)$ in (3.19) from the sensor measurements.

3.5.2 Measurement Model

We consider a synchronous setting where each node measures the field at time nT and position $\mathbf{r}_i$. Additionally, we assume noise which includes measurement noise on the one hand and modeling errors on the other hand. This leads to the measurement equation at sensor node i:

$$f_i[n] = f(\mathbf{r}_i; nT) + w_i[n]\,. \tag{3.20}$$

Expressing the field in terms of its basis expansion in (3.19), stacking the field measurements into a vector $\mathbf{f}[n] = \big(f_1[n]\ f_2[n]\ \ldots\ f_I[n]\big)^T$, and using the length-$L$ vectors

$$\mathbf{u}_i = \big(u_1(\mathbf{r}_i)\ u_2(\mathbf{r}_i)\ \ldots\ u_L(\mathbf{r}_i)\big)^T,$$
$$\mathbf{c}[n] = \big(c_1(nT)\ c_2(nT)\ \ldots\ c_L(nT)\big)^T,$$

we rewrite the measurement equation in matrix-vector notation as

$$\mathbf{f}[n] = \mathbf{U}\mathbf{c}[n] + \mathbf{w}[n]\,, \tag{3.21}$$

where the $I \times L$ subspace matrix $\mathbf{U}$ is given by $\mathbf{U} = \big(\mathbf{u}_1 \ \mathbf{u}_2 \ \ldots \ \mathbf{u}_I\big)^T$ and $\mathbf{w}[n] = \big(w_1[n] \ w_2[n] \ \ldots \ w_I[n]\big)^T$ is the noise vector. Apart from the noise, the structure is similar to (2.8). The rows of $\mathbf{U}$ contain samples of the basis functions at the sensors positions. Hence, for a correct field reconstruction, the sampling criterion of the basis functions have to be fulfilled, e.g., for Fourier basis the Nyquist criteria. A Legendre basis is considered in Section 3.6.3 and also others in [45]. In the next step we show how to estimate the time-varying basis coefficients $\mathbf{c}[n]$ in a distributed fashion.

3.5.3 Distributed Least-Squares

The coefficients $\mathbf{c}[n]$ in (3.21) can be estimated by solving the LS problem

$$\hat{\mathbf{c}}[n] = \arg\min_{\mathbf{c}} \big\|\mathbf{f}[n] - \mathbf{U}\mathbf{c}[n]\big\|_2^2 ,$$

which yields [74]

$$\hat{\mathbf{c}}[n] = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{f}[n] , \tag{3.22}$$

In the following we use $\mathbf{U}^{\#} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$ which is the pseudo-inverse of $\mathbf{U}$. Inappropriate sampling will lead to an ill-conditioned problem. Hence, a sufficient number of sensors is required and the sensors need to be well spread over the sampling region. Clearly, (3.22) can be solved easily in a centralized manner, if the fusion center knows all sensor measurements and also all sensor positions (to determine $\mathbf{U}$ and subsequently $\mathbf{U}^{\#}$). Our goal however, is to find a distributed solution. To that end, we rewrite the solution of the LS problem in a component-wise manner:

$$\hat{\mathbf{c}}[n] = \sum_{i=1}^{I} \mathbf{u}_i^{\#} f_i[n] = \frac{1}{I}\sum_{i=1}^{I} \mathbf{v}_i[n] \, , \text{with } \mathbf{v}_i[n] = \mathbf{u}_i^{\#} f_i[n] \, I \, . \tag{3.23}$$

It is seen that if sensor i knows the number of sensors I and the ith column $\mathbf{u}_i^{\#}$ of $\mathbf{U}^{\#}$ in addition to the measurement $f_i[n]$, the problem amounts to distributed averaging of the L elements of the vector $\mathbf{v}_i[n]$. Hence, we can run L instances of any distributed averaging algorithm (e.g. CP or AC), with the lth instance operating on $s_i[n] = [\mathbf{v}_i[n]]_l$. For the case of (dynamic) CP, the κ-messages defined in (3.2) are the same for all instances and hence need to be computed and exchanged only once.

In the above argument, we assumed that each sensor knows the corresponding column of the pseudo-inverse, i.e., $\mathbf{u}_i^{\#} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{u}_i$, which requires on the one hand that $\mathbf{U}^T\mathbf{U}$ is globally known and on the other hand that the ith column of $\mathbf{U}$ is available at the ith sensor. The latter is fulfilled when each sensor knows its own position, because $\mathbf{u}_i$ are the basis functions sampled at position $\mathbf{r}_i$ (cf. (3.20)). Unfortunately, it is not so simple to derive the Gramian $\mathbf{U}^T\mathbf{U}$ at each node, because this requires that each sensor knows the positions of all sensors to be able to compute the elements of the Gramian via the inner products $[\mathbf{U}^T\mathbf{U}]_{k,l} = \sum_{i=1}^{I} u_k(\mathbf{r}_i)\, u_l(\mathbf{r}_i)$.

That each sensor has to know its own position is not always trivial but acceptable in WSN. However, that all sensors know the positions of all other sensors is possible in principle (even via distributed methods, e.g. [55]), but often undesired. Hence, we propose two approaches where only local knowledge is necessary to approximately obtain the Gramian. Both methods also implicitly compensate the factor I which appears in the definition of $\mathbf{v}_i[n]$ (cf. (3.23)).

Orthogonal Approximation

The first approach builds on orthogonal basis functions $\{u_l(\mathbf{r})\}$. If the basis functions are band-limited and the sensors are placed on a sufficiently dense uniform lattice such that the Nyquist criterion is satisfied, the orthogonality is inherited by the sampled basis functions, i.e., $\sum_{i=1}^{I} u_k(\mathbf{r}_i)\, u_l(\mathbf{r}_i) = I\delta_{kl}$. The idea now is to assume that such orthogonality relations hold approximately true for arbitrary dense sensor placements, i.e.,

$$\mathbf{U}^T\mathbf{U} \approx I\,\mathbf{I}\,. \tag{3.24}$$

The accuracy of this approximation improves with increasing number of sensors (see [45]). The approximation (3.24) implies $\mathbf{u}_i^{\#} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{u}_i \approx \frac{1}{I}\mathbf{u}_i$ and hence further $\mathbf{v}_i[n] = \mathbf{u}_i^{\#} f_i[n] I \approx \mathbf{u}_i f_i[n]$.

CP-based Computation of the Gramian

For WSN with few or irregularly placed sensors or for nonorthogonal bases, the approximation (3.24) may be too inaccurate. For such scenarios, we observe that the elements of the scaled Gramian $\mathbf{G} = \frac{1}{I}\mathbf{U}^T\mathbf{U}$ equal

$$[\mathbf{G}]_{k,l} = \frac{1}{I}\sum_{i=1}^{I} u_k(\mathbf{r}_i)\, u_l(\mathbf{r}_i)\,.$$

This is the arithmetic mean of $u_k(\mathbf{r}_i)\, u_l(\mathbf{r}_i)$ and hence can again be computed by applying distributed averaging to the "observations" $s_i = u_k(\mathbf{r}_i)\, u_l(\mathbf{r}_i)$ (again, this requires each sensor to know only its own position). Since $\mathbf{G}$ is symmetric, $\frac{L(L+1)}{2}$ instances of averaging are sufficient to obtain estimates $\hat{\mathbf{G}}_i^{(n)}$ of $\mathbf{G}$ at each sensor. Because $\mathbf{G}^{-1}\mathbf{u_i} = I\mathbf{u}_i^{\#}$, estimates of $I\mathbf{u}_i^{\#}$ can be computed at each sensor by multiplying $\mathbf{u}_i$ with the (possibly regularized) inverse of the averaging output $\hat{\mathbf{G}}_i^{(n)}$. Since the result already incorporates the scaling factor I, we have $\mathbf{v}_i[m] \approx \big(\hat{\mathbf{G}}_i^{(n)}\big)^{-1}\mathbf{u}_i\, f_i[m]$. We finally note that the CP-based estimation of $\mathbf{G}$ can either be performed as a preprocessing step during the initialization phase of the WSN or in an online manner, i.e., simultaneously with the estimation of the field coefficients. In the latter case the convergence results of distributed averaging no longer apply.

3.5.4 WLS Approach

If we consider basis functions like the Legendre polynomials we notice that they have a maximum or minimum at the boundary of $\mathcal{A}$. It follows directly from this property that sensors at the boundary initially obtain poorer estimates of the basis coefficients than the sensors in the center. Therefore, it is reasonable to weight the measurements from nodes at the boundary less than the others. Hence, we define for each node a weighting factor ω_i (which is preferable dependent on the distance to the boundary of the field) and replace the LS estimate 3.22 with the WLS estimate

$$\hat{\mathbf{c}}^* = (\mathbf{U}^T\boldsymbol{\Omega}\,\mathbf{U})^{-1}\mathbf{U}^T\boldsymbol{\Omega}\,\mathbf{f}[n]\,. \tag{3.25}$$

where $\boldsymbol{\Omega} = \mathrm{diag}\{\omega_1, \omega_2, ..., \omega_I\}$. The distributed LS method described in 3.5.3 can be modified accordingly.

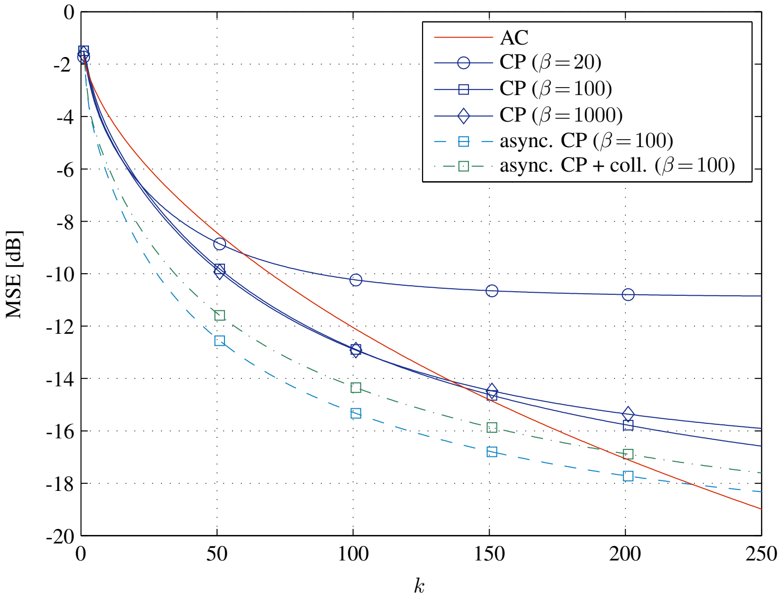


Figure 3.5: *Transient performance of AC and synchronous and asynchronous CP with different values of β and collisions for WSN with 100 nodes.*

3.6 Numerical Results

In this section we present numerical results which provide an intuition on how (static and dynamic) CP performs. In particular, we compare CP with AC in several scenarios and elaborate the potential applications for CP and AC, respectively. For the error measure we use the (weighted) MSE which we defined in Section 2.2.5. In static scenarios the time index n is replaced by the iteration index k and s_i and $\bar{s}$ is invariant to k. For all following simulations we consider random geometric graphs in an area $[0,1]\times[0,1]$ since those graphs are suitable to model WSN and inhere a geographic topology which allows a simple modeling of correlations between nodes. When AC is performed the weights are always constructed through the MH method, because it can be performed in a distributed manner.

The measurement vector s is obtained by sampling low-pass fields with spatial cut-off frequency $\theta_{\mathrm{c}}^{(\mathrm{s})}$ (cf. Section 2.4). For some scenarios we directly sample Gaussian noise. In the case of dynamic (time-varying) scenarios, we generate temporal low-pass signals with cut-off frequency $\theta_{\mathrm{c}}^{(\mathrm{t})}$.

3.6.1 Static Averaging

First of all we start with the initial convergence behavior of static CP and AC. We will only consider the transient phase, because we think that this phase has the most practical relevance for WSN, since flooding schemes are more efficient in terms of energy if more iterations are allowed. In Fig. 3.5 we show the MSE performance for AC and CP with different values of β in graphs with 100 nodes, connectivity of $c = 1.5$, and for 3000 Monte Carlo runs. Initially all sensors measure low-pass filtered white Gaussian noise with (spatial) cut-off frequency $\theta_{\mathrm{c}}^{(\mathrm{s})} = 0.08$. It is seen that when β is small CP saturates very fast with a large MSE. For $\beta = 100$ we achieve the best performance for $k < 250$. In the case when β is large, the MSE converges slower in the

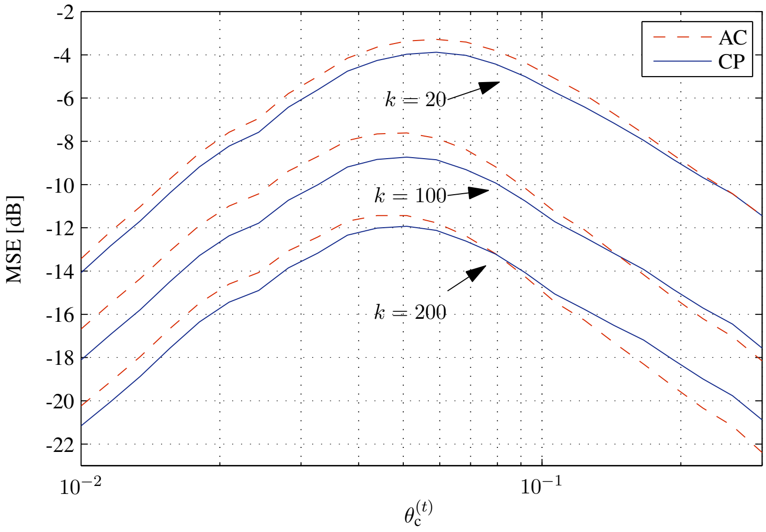


Figure 3.6: *Averaged MSE at different iterations k versus the spatial cut-off frequency of filtered Gaussian noise for WSN with 300 nodes.*

transient phase, but when k gets large CP will converge to a smaller MSE. This is consistent with Proposition 1. AC, in contrast, is slower until 150 iterations, but performs superior and the convergence speed is much faster for $k > 250$. Furthermore, asynchronous CP outperforms synchronous CP in Fig. 3.5 during the observed period. This behavior is due to the fact that information can be distributed to more than the direct neighbors in I asynchronous steps (which is equal in the power sense to one synchronous step). Finally, when collisions occur as described in Section 3.4.2 (the duty cycle was $\tau = 4 \cdot 10^{-3}$) the MSE is slightly worse than without collisions, but still better than the synchronous case.

In the next simulation we study the MSE at different iterations over the cut-off frequency $\theta_c^{(s)}$ of the initially measured filtered Gaussian noise. We did 3000 Monte Carlo runs with graphs of 300 nodes and connectivity $c = 1.6$. The result is presented in Fig. 3.6 and shows that CP (with $\beta = 100$) and AC perform worst when $\theta_c^{(s)} \approx 5 \cdot 10^{-2}$. When $\theta_c^{(s)}$ becomes smaller, the initial field tends to a constant field, i.e., the average is almost measured initially, and therefore the MSE decreases in Fig. 3.6 for both averaging methods. When $\theta_c^{(s)}$ gets larger the initial field is more like uncorrelated spatial noise, which has good transient performance because the local update equations lead to a good estimate of the average after a few iterations. Moreover we see that for the chosen parameter set CP by and large outperforms AC. For that purpose we are going to investigate the performance versus the connectivity c in the next simulation.

The graph setting for the results in Fig. 3.7 are the same as in the previous simulation. Additionally to random geometric graphs we have also computed the performance in tree graphs. It is seen that the performance improves for larger connectivities c; this gain increases linearly for AC and saturates for CP. The general behavior for both averaging methods remains almost the same for measured noise and low-pass fields (in our simulations with $\theta_c^{(s)} = 0.07$), only that there is a difference of approximately 15 dB between the MSE curves of the field and the noise. According to Fig. 3.7 CP outperforms AC in the case of low-pass fields and tree graphs (with preferable large β, in this setting it was chosen to be 1000) and sparse graphs. In contrast AC performs well when c is large. Hence, these results confirm the theory, i.e., CP has excellent performance with

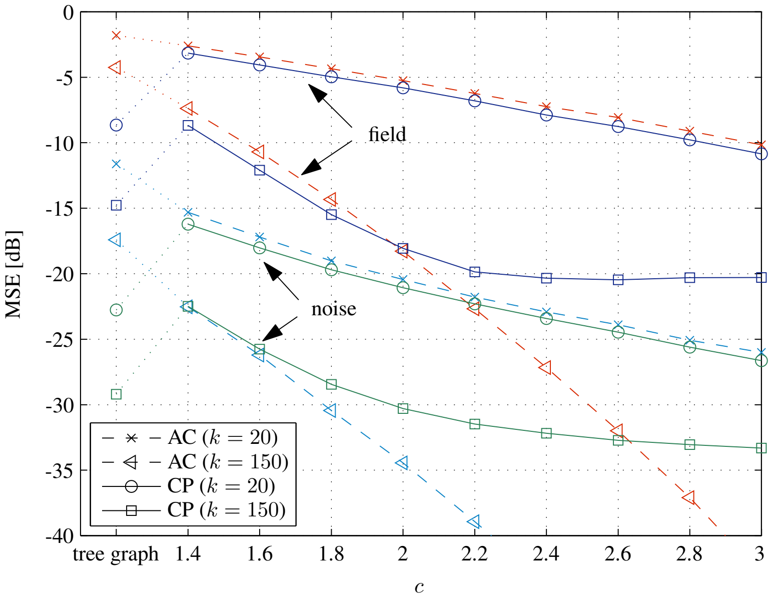


Figure 3.7: *Dependence of the MSE on the graph connectivity for an uncorrelated and a correlated field ($\theta_c^{(s)} = 0.07$) and for $k = 20$ and $k = 150$.*

tree graphs and large β [7] and AC with fully connected graphs.

3.6.2 Dynamic Averaging

In this section we are investigating the performance of dynamic averaging. First of all we present numerical results for the transfer function of CP and AC which were derived in Sections 3.2.4 and 3.3. The results were averaged over 20 graph realizations, where random geometric graphs with 100 nodes and $c = 1.5$ were considered. Fig. 3.8 shows the corresponding amplitude response and group delay. The average amplitude behavior of CP and AC is equal for $\theta = 0$ where it corresponds to the mean, but the group delay shows that AC gains the mean without delay (this is due to the property of AC that the average is always preserved), CP in contrast induces a delay of approximately 120 iterations. This delay can be mitigated through filtering as described in Section 3.2.5; in Fig. 3.8 we also provide the results for CP augmented with a linear predictor ($L = 300$, $h = 100$) for a low-pass process with $\theta_c^{(t)} = 5 \cdot 10^{-3}$. Although the delay compensation works fine, the predictor amplifies signals with frequencies outside of the interval $[-\theta_c^{(t)}, \theta_c^{(t)}]$, which is not desirable for noisy signals. To mitigate this effect, all expected artifacts have to be included in the source process for which the predictor is designed. The same problem with the noise is also relevant for AC, since the average amplitude behavior of AC has high-pass character and hence AC is much more sensitive to noise than CP. When the weights of AC are optimized according to Section 3.3 it is seen that the amplitude behavior and group delay have slightly improved, whereas the tremendous noise amplification at frequencies $-1/2$ and $1/2$, which cannot be seen in this plot, make this approach useless for practical applications.

The transfer behavior in Fig. 3.8 can also be interpreted in the sense that results dependent on the graph depth (the number on how many hops are required to get from one node to an other) are obtained. Therefore, the average amplitude response and group delay versus the graph depth for two different frequencies can be seen in

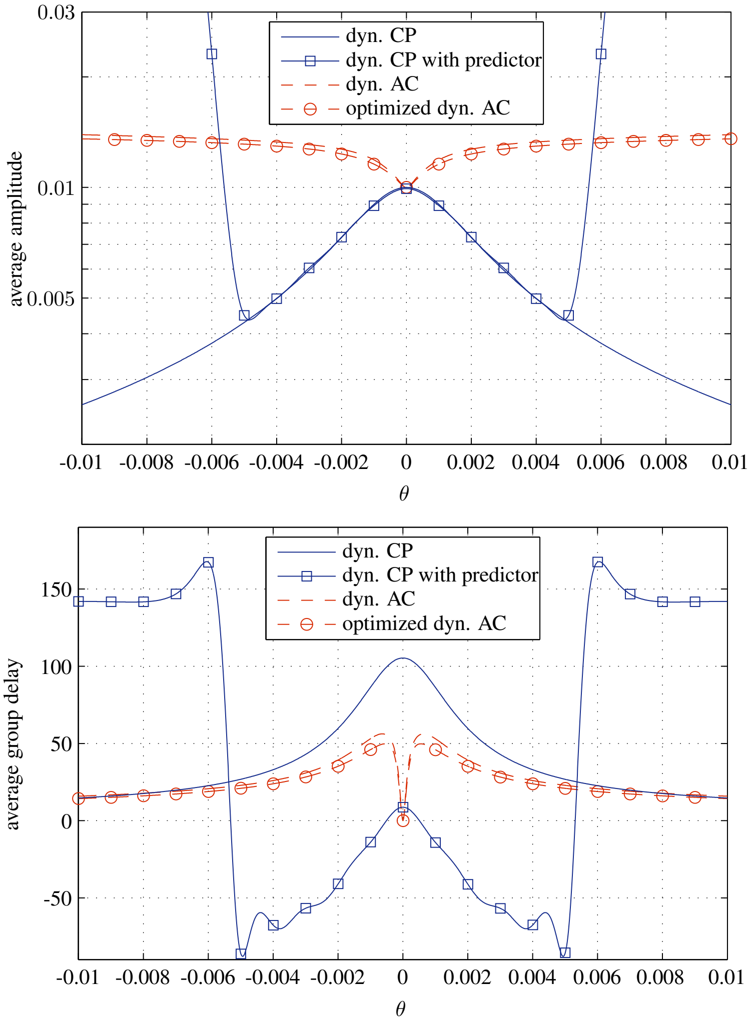


Figure 3.8: *Amplitude response (top) and group delay (bottom) of dynamic CP and AC for WSN with 100 nodes.*

Fig. 3.9 and Fig. 3.10. These values are obtained by collecting and then averaging the amplitude responses and group delays between nodes that have a certain hop distance. The average amplitude response in Fig. 3.9 shows that the influence of nodes which are farther away is less than the neighboring nodes, which is intuitive. The curves in Fig. 3.10 underline that slow variations require more time to pass through the network. Moreover, except for AC at low frequencies, the average group delay grows linearly with the depth. Unfortunately the scaling factor of this growth is larger than one (which would be optimum). The result has to be interpreted with caution, since the amount of nodes at a certain depth is not taken into account and hence the graph structure will have an additional impact on the behavior.

Finally the MSE behavior of dynamic averaging is shown in Fig. 3.11. For this simulations we generated a 2D spatial field through 16 coefficients of a Fourier basis, where each coefficient varies over time according a low-pass process with cut-off frequency $\theta_{\mathrm{c}}^{(\mathrm{t})}$, plus noise ($\mathrm{SNR} = 10\,\mathrm{dB}$). We plot the MSE of 1000 different realizations for random geometric graphs with 100 nodes and connectivity $c = 1.5$. For all CP variants we had $\beta = 100$. An important point is that all variants of CP involve a delay. For $\theta_{\mathrm{c}}^{(\mathrm{t})} = 10^{-4}$ a delay of approximately 400 occurs for (synchronous) dynamic CP, for asynchronous CP the delay is roughly the half. When the cut-

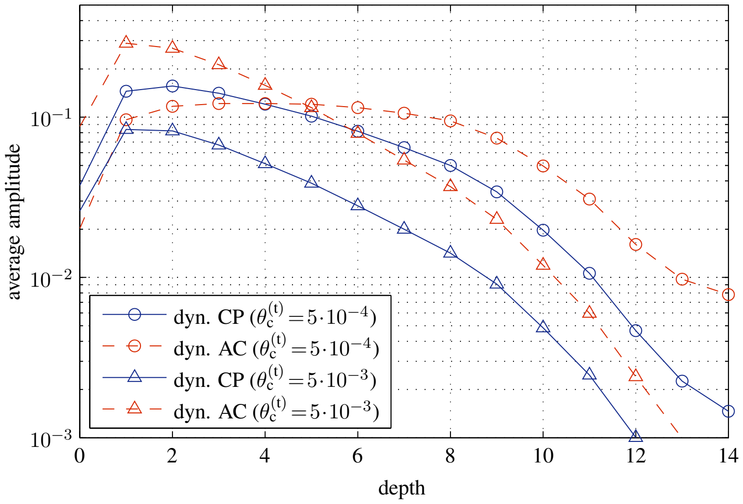


Figure 3.9: *Average amplitude response of dynamic CP and AC at two frequencies versus the hop distance between nodes for WSN with 100 nodes.*

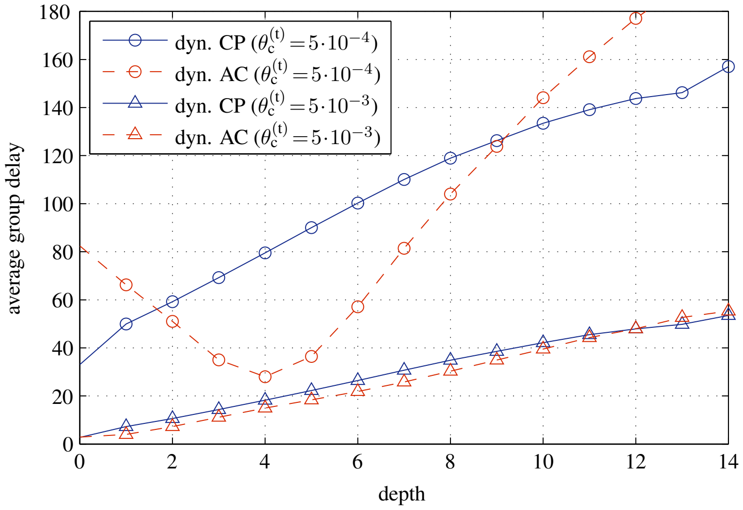


Figure 3.10: *Average group delay of dynamic CP and AC at two frequencies versus the hop distance between nodes for WSN with 100 nodes.*

off frequency increases the delay becomes smaller, e.g., for $\theta_c^{(t)} = 10^{-2}$ it is around 50 for synchronous and asynchronous CP. The delay can be decreased by making β smaller, but this increases the MSE, in particular the variance of the results of CP increase and therefore the MSE. Because of the low-pass character of CP it is also seen in Fig. 3.11 that CP cannot track the average for high frequencies (and the MSE tends to 0dB), but the variance of the results is still small, which implies that a consensus is reached. This is not the same for AC (here, the MSE is always equal to the variance), where the local results will always follow the observations. Especially for spatial low-pass fields this is far away from consensus, which results in a poor performance of AC for low SNR, which we also observed in simulations that are not presented in this thesis. For high SNR, however, AC outperforms CP at low frequencies. This SNR dependency of AC cannot be observed with CP where the MSE behavior is almost invariant of noise. Finally, the behavior of asynchronous CP (even with collisions)

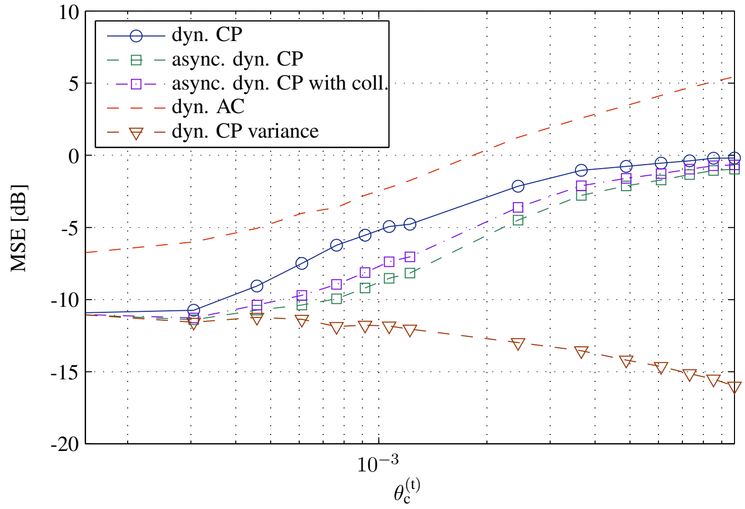


Figure 3.11: *MSE versus temporal cut-off frequency for dynamic averaging for a spatiotemporal low-pass field.*

outperforms synchronous CP not just in the static case, even in the dynamic case the MSE is significantly better and as already mentioned, the smaller delay is also favorable.

3.6.3 Distributed Field Reconstruction

So far we have only considered the tracking of the average (DC component of a field), but if it is necessary to track an entire field the method described in Section 3.5 has to be used (for simplicity we assume that the Gramian is known perfectly, see [33] for imperfect knowledge). We considered a setting similar to the previous simulation (i.e., $\beta = 100$), but we only average over 500 scenarios. In Fig. 3.12 we plot the MSE of the estimated field at each node. The SNR was chosen as 25 dB and models measurement noise. It is seen that the behavior is very similar to the tracking of the average in Fig.3.11. Again we have the behavior that AC is very sensitive to noise. For CP we also added the case where a field is constructed and estimated through a Legendre basis, that yields an MSE a bit worse to the Fourier basis. The reason for this are probably the minima and maxima of the Legendre polynomials at the boundaries, which are harder to average, because lots of information has to be passed through the graph. If a better performance of the field reconstruction is necessary, the number of iterations between each sampling period has to be increased (so far we had $M = 1$ in all dynamic simulations). In Fig. 3.12 it is seen that for $M = 10$ and $\beta = 500$ the performance is increased tremendously. It is important to note that increasing only β can have a negative effect.

3.7 Conclusions

Throughout this chapter we considered mainly dynamic distributed averaging including practical implementation issues. Although AC has a simple theoretical background and preferable convergence behavior, it does not have the advantages of CP for applications in WSN. We have shown that CP converges to the average up to an error, but still achieves better MSE performance for the initial phase than AC. In settings, however, where

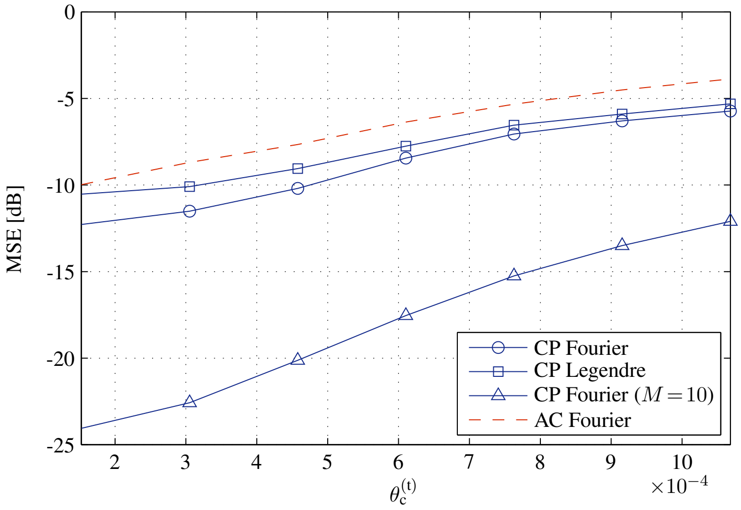


Figure 3.12: *MSE versus temporal cut-off frequency for distributed field estimation in WSN with 100 nodes.*

enough time and energy for a large number of processing iterations is available, AC performs better than CP. The superior behavior of CP is in particular observed in sparse networks, such as WSN. Moreover, our state space analysis provides better insights into the behavior of dynamic CP and AC. The state space model also allowed us to prove the stability of dynamic CP. Then we introduced a medium access protocol where simulations showed, that asynchronous CP performs better than synchronous CP even with collisions. Finally, we have shown how to use distributed averaging for field estimation.

An open question is the design of the parameter β of CP. Through our simulations we provided an intuition how to choose β but there exists no direct rule. A further improvement would be a proper design of edge weights for CP, similar to AC, but there have been only empirical approaches so far. Anyway, investigating CP is not trivial due to its nonlinear structure and we expect that for many open questions only heuristic solutions are feasible.

3.A Proofs

3.A.1 Diagonal Dominance of CP

Consider the quadratic form $\sum_{(i,j)\in\mathcal{E}} q_{ij}(\theta_i-\theta_j)^2 = \boldsymbol{\theta}^T\boldsymbol{\Gamma}\boldsymbol{\theta}$ with $q_{ij}>0$ for all $(i,j)\in\mathcal{E}$. For the case when $q_{ij}=1$ it is obvious that $\boldsymbol{\Gamma}=\mathbf{L}$ [7]. In the following we derive an expression for $\boldsymbol{\Gamma}$ assuming positive q_{ij}:

$$\begin{aligned}\sum_{(i,j)\in\mathcal{E}} q_{ij}(\theta_i-\theta_j)^2 &= \sum_{(i,j)\in\mathcal{E}} q_{ij}(\theta_i^2+\theta_j^2-2\theta_i\theta_j)\\ &= -2\sum_{(i,j)\in\mathcal{E}} q_{ij}\theta_i\theta_j + \sum_{(i,j)\in\mathcal{E}} q_{ij}\theta_i^2 + \sum_{(i,j)\in\mathcal{E}} q_{ij}\theta_j^2\,.\end{aligned}$$

Since $q_{ij}=0$ if $(i,j)\notin\mathcal{E}$ and $q_{ij}=q_{ji}$ we have

$$\sum_{(i,j)\in\mathcal{E}} q_{ij} = \sum_{i=1}^{I}\sum_{j=1}^{j<i} q_{ij} = \sum_{i=1}^{I}\sum_{j=i+1}^{j\le I} q_{ij}\,,$$

because the entire edge set is represented through the lower triangular or the upper triangular part of $\mathbf{Q}$ with $[\mathbf{Q}]_{ij} = q_{ij}$. Since we assume undirected graphs $\mathbf{Q}$ is symmetric. Finally we have

$$\begin{aligned}\sum_{(i,j)\in\mathcal{E}} q_{ij}(\theta_i-\theta_j)^2 &= -2\sum_{(i,j)\in\mathcal{E}} q_{ij}\theta_i\theta_j + \sum_{(i,j)\in\mathcal{E}} q_{ij}\theta_i^2 + \sum_{(i,j)\in\mathcal{E}} q_{ij}\theta_j^2 \\ &= -2\sum_{i=1}^{I}\sum_{j=1}^{j<i} q_{ij}\theta_i\theta_j + \sum_{i=1}^{I}\theta_i^2\sum_{j=1}^{j<i} q_{ij} + \sum_{j=1}^{I}\theta_j^2\sum_{i=j+1}^{i\le I} q_{ij} \\ &= -\sum_{\forall i,j} q_{ij}\theta_i\theta_j + \sum_{i=1}^{I}\theta_i^2\sum_{j=1}^{I} q_{ij} = \boldsymbol{\theta}^T(\mathrm{diag}\{\mathbf{Q1}\}-\mathbf{Q})\boldsymbol{\theta} = \boldsymbol{\theta}^T\boldsymbol{\Gamma}\boldsymbol{\theta}\,,\end{aligned}$$

and hence, $\boldsymbol{\Gamma} = \mathrm{diag}\{\mathbf{Q1}\} - \mathbf{Q}$.

The average is obtained with CP by maximizing the following exponential expression

$$\arg\max_{\boldsymbol{\theta}} \exp\left\{-||\boldsymbol{\theta}-\mathbf{s}||^2 - \beta\boldsymbol{\theta}^T\boldsymbol{\Gamma}\boldsymbol{\theta}\right\} = \arg\max_{\boldsymbol{\theta}} \exp\left\{-\boldsymbol{\theta}^T(\mathbf{I}+\beta\boldsymbol{\Gamma})\boldsymbol{\theta} + 2\mathbf{s}^T\boldsymbol{\theta}\right\}. \tag{3.26}$$

Hence, $(\mathbf{I}+\beta\boldsymbol{\Gamma})$ is the information matrix of GaBP and according to [11] it has to be diagonal dominant to ensure convergence of GaBP. Since $\mathbf{I}+\beta\boldsymbol{\Gamma} = \mathbf{I}+\beta\,\mathrm{diag}\{\mathbf{Q1}\} - \beta\mathbf{Q}$ the diagonal elements are defined as $1+\beta\sum_{\forall j} q_{ij}$ and the off diagonal elements are simply $-\beta q_{ij}$. Therefore, the criterion for diagonal dominance $\sum_j |-\beta q_{ij}| < |1+\beta\sum_{\forall j} q_{ij}|$ is always fulfilled for positive q_{ij}.

3.A.2 Proof of Proposition 1

The optimization problem in (3.26) can be rewritten as $\arg\min_{\boldsymbol{\theta}} \boldsymbol{\theta}^T(\mathbf{I}+\beta\boldsymbol{\Gamma})\boldsymbol{\theta} - 2\mathbf{s}^T\boldsymbol{\theta}$. Setting the first derivative to zero let us obtain

$$(\mathbf{I}+\beta\boldsymbol{\Gamma})\hat{\hat{\mathbf{s}}} = \mathbf{s}\,. \tag{3.27}$$

Following the derivation in [7] we use the eigendecomposition $\boldsymbol{\Gamma} = \mathbf{UDU}^T$ where the columns of $\mathbf{U}$ contain the eigenvectors of $\boldsymbol{\Gamma}$ and $\mathbf{D}$ is a diagonal matrix with the corresponding eigenvalues λ_i. Therefore, we can rewrite the solution of (3.27) as

$$\hat{\hat{\mathbf{s}}} = \mathbf{U}(\mathbf{I}+\beta\mathbf{D})^{-1}\mathbf{U}^T\mathbf{s} = \sum_i \mathbf{u}_i\mathbf{u}_i^T\frac{1}{1+\beta\lambda_i}\mathbf{s}\,.$$

Since we have $\boldsymbol{\Gamma}\mathbf{1} = (\mathrm{diag}\{\mathbf{Q1}\}-\mathbf{Q})\mathbf{1} = \mathbf{Q1}-\mathbf{Q1} = \mathbf{0}$ the matrix $\boldsymbol{\Gamma}$ has one zero eigenvalue with corresponding normalized eigenvector $\frac{1}{\sqrt{I}}\mathbf{1}$ which yields

$$\hat{\hat{\mathbf{s}}} = \frac{1}{I}\mathbf{1}\mathbf{1}^T\mathbf{s} + \sum_{i=2}^{I}\mathbf{u}_i\mathbf{u}_i^T\frac{1}{1+\beta\lambda_i}\mathbf{s}\,.$$

The first term provides the average of the input vector. The second term, however, is the induced error, i.e.,

$$\epsilon^2 = \frac{||\hat{\hat{\mathbf{s}}}-\mathbf{s}||_2^2}{||\mathbf{s}||_2^2} = \frac{\left|\left|\sum_{i=2}^{I}\mathbf{u}_i\mathbf{u}_i^T\frac{1}{1+\beta\lambda_i}\mathbf{s}\right|\right|_2^2}{||\mathbf{s}||_2^2}$$

that tends to zero if $\beta \to \infty$. The squared error can be bounded using the fact the the vectors $\mathbf{u}_i$ are orthonormal as

$$\epsilon^2 \leq \left\| \sum_{i=2}^{I} \mathbf{u}_i \mathbf{u}_i^T \frac{1}{1+\beta\lambda_i} \right\|_2^2 = \sum_{i=2}^{I} \left| \frac{1}{1+\beta\lambda_i} \right|^2 \|\mathbf{u}_i \mathbf{u}_i^T\|_2^2 = \sum_{i=2}^{I} \left| \frac{1}{1+\beta\lambda_i} \right|^2 = \sum_{i=2}^{I} \frac{1}{(1+\beta\lambda_i)^2} \,,$$

which concludes the proof. □

3.A.3 Bounds on the κ Messages

The expected value of the κ-messages that are defined in (3.2) is given by,

$$\mathrm{E}\{\kappa_{i\to j}\} = \mathrm{E}\left\{ \frac{1 + \sum_{u \in \mathcal{N}(i)\backslash j} \kappa_{u\to i}}{1 + \frac{1}{\beta q_{ij}} \left(1 + \sum_{u \in \mathcal{N}(i)\backslash j} \kappa_{u\to i}\right)} \right\} \leq \frac{1 + \mathrm{E}\left\{\sum_{u \in \mathcal{N}(i)\backslash j} \kappa_{u\to i}\right\}}{1 + \frac{1}{\beta q_{ij}} \left(1 + \mathrm{E}\left\{\sum_{u \in \mathcal{N}(i)\backslash j} \kappa_{u\to i}\right\}\right)} \,,$$

where the inequality follows from concavity. We can rewrite the term $\mathrm{E}\left\{\sum_{u \in \mathcal{N}(i)\backslash j} \kappa_{u\to i}\right\} = (\mathrm{E}\{d\}-1)\,\mathrm{E}\{\kappa\}$ where d denotes the degree. Since we want to have the expected value not conditioned on any specific i and j we also have $\mathrm{E}\{\kappa_{u\to i}\} = \mathrm{E}\{\kappa\}$. Therefore, we obtain an upper-bound for the κ-messages which is equal to the converged κ-messages in a regular graph where each node has $\mathrm{E}\{d\}$ neighbors (which is not necessarily an integer), see [7] for more details.

If the degree of node i is given we can use the previous result and obtain

$$\mathrm{E}\{\kappa_{i\to j} | d_i = n\} \leq \frac{1 + (n-1)\,\mathrm{E}\{\kappa\}}{1 + \frac{1}{\beta q_{ij}} \left(1 + (n-1)\,\mathrm{E}\{\kappa\}\right)} \,.$$

4

Weight Design Methods for Average Consensus

A powerful tool to enhance the performance of average consensus (AC) is to tune the weights which correspond the edges of the communication graph. Motivated by this fact, we introduce in this chapter new weight design schemes that lead to a more favorable performance of AC.

First of all, we provide sufficient and necessary convergence conditions for constant and Metropolis-Hastings weights. Although such conditions are mentioned in literature, our approach allows new insights when using Metropolis-Hastings weights in WSN.

The next part provides a weight sequence that yields optimum per-step convergence of AC. In particular, second-order statistics are used to optimize the weights such that the MSE sequence is minimized. The main observation is that time-varying weights yield a performance benefit. Therefore, we introduce weight morphing, a heuristic approach with time-varying weights. This methods uses weights that have excellent performance for uncorrelated observations in the transient phase, and as soon as the inner states become more correlated, the initial weights are morphed into the asymptotic optimum weights.

Finally, we show how to apply an advective flow to AC. Our method is based on the work of [35] but uses a different discretization scheme that allows faster flows and more advanced advection schemes, e.g., an advective flow which is inspired by the behavior of a fluid in a blender. Additionally, we show that our novel approach is also applicable to general WSN.

4.1 Convergence Conditions for Constant and Metropolis-Hastings Weights

Any weight design for AC requires that the convergence conditions are fulfilled (cf. Section 2.2.2). In the case of undirected graphs the first condition $\mathbf{W1} = \mathbf{1}$ ensures that the matrix is doubly stochastic and can be easily satisfied by setting the "self weight" to $w_{ii} = 1 - \sum_{j\neq i} w_{ij}$. The second condition requires that $\rho\{\mathbf{W} - \mathbf{J}\} < 1$ holds, which cannot be enforced in a similar trivial manner. In this context we present necessary and sufficient conditions for the stability of AC with constant weights (CW) and Metropolis-Hastings (MH) weights in this section. Since we only deal with undirected graphs in the context of AC, we solely consider undirected graphs in this section as well. For constant weights [5] states necessary and sufficient conditions for the convergence of AC with general (directed and undirected) graphs. The authors show that for

$$\mathbf{W}^{\text{CW}} = \mathbf{I} - \alpha\mathbf{L}\,,$$

the parameter α has to be bounded as $0 < \alpha < \frac{1}{\max_i d_i}$. Switching to the MH weight design, we introduce the following generalized version of MH weights (GMH), using a non-negative regularization parameter $\epsilon \geq 0$:

$$w_{ij}^{\text{MH}}(\epsilon) = \begin{cases} \frac{1}{\max\{d_i,d_j\}+\epsilon}\,, & \text{for } (i,j) \in \mathcal{E}, \\ 0\,, & \text{for } (i,j) \notin \mathcal{E}, \text{ and } i \neq j, \\ 1 - \sum_{j\neq i} w_{ij}^{\text{MH}}(\epsilon)\,, & \text{for } i = j. \end{cases}$$

The conventional MH weight designs are obtained with $\epsilon = 0$ [40] and $\epsilon = 1$ [41]. The latter was proposed to guarantee convergence for any graph topology (see next section). Decreasing ϵ also decreases the "self-loop" weights $w_{ii}^{\text{MH}}(\epsilon)$ and implies that the nodes tend less to preserve their own current state.

4.1.1 Sufficient Conditions for Convergence

In the following we use weighted undirected graphs with (strictly) positive weights to show stability. As mentioned in Section 2.1.1, $\mathbf{A}_\mathbf{w}$ is the adjacency matrix with the weights contained in the vector $\mathbf{w}$ assigned to the edges. We have $[\mathbf{A}_\mathbf{w}]_{ij} = w_{ij}$, where $w_{ij} = [\mathbf{w}]_l$ with l indicates the edge $(i,j) \in \mathcal{E}$. The weighted Laplacian is defined as

$$\mathbf{L}_\mathbf{w} = \mathbf{B}\,\text{diag}\{\mathbf{w}\}\mathbf{B}^T = \mathbf{B}_\mathbf{w}\mathbf{B}_\mathbf{w}^T = \text{diag}\{\mathbf{A}_\mathbf{w}\mathbf{1}\} - \mathbf{A}_\mathbf{w}\,,$$

where $\mathbf{B}$ is the incidence matrix and $\mathbf{B}_\mathbf{w} = \mathbf{B}\,\text{diag}\{\sqrt{\mathbf{w}}\}$ the incidence matrix of the weighted graph. The assignment of the sign in $\mathbf{B}$ plays no role in the following, it is only important that we have one $+1$ and one -1 in each column. The degree of node i in the weighted graph is defined as $d_\mathbf{w}(i) = \sum_{j\in\mathcal{N}(i)} w_{ij}$ and is also equal to the ith row/column sum of $\mathbf{A}_\mathbf{w}$, similar to ordinary graphs. Using the Laplacian of the weighted graph we can compute the AC weight matrix as

$$\mathbf{W} = \mathbf{I} - \mathbf{L}_\mathbf{w}\,.$$

For the stability of AC we have to show that $\mathbf{W}$ is doubly stochastic and that $\rho\{\mathbf{W} - \frac{1}{I}\mathbf{1}\mathbf{1}^T\} < 1$. It follows directly from the definition of $\mathbf{L}_\mathbf{w}$ that $\mathbf{W}$ is doubly stochastic, which also implies that there exists an eigenvalue with magnitude one. Since we cancel this eigenvalue in the second condition, it is necessary that there exist no other eigenvalue at plus one. Hence, we have to ensure that the multiplicity of the 1 eigenvalue of $\mathbf{W}$ equals one. In other words, the multiplicity of the zero eigenvalue of the Laplacian has to be one (since $\lambda(\mathbf{L_w}) = 1 - \lambda(\mathbf{W})$). We have

$$\mathbf{L_w}\mathbf{v} = 0\,\mathbf{v}\,,$$

where $\mathbf{v}$ denotes the corresponding eigenvector for eigenvalue 0. We can multiply this equation with $\mathbf{v}^T$ from the left and obtain:

$$\mathbf{v}^T\mathbf{L_w}\mathbf{v} = \sum_{(i,j)\in\mathcal{E}} w_{ij}(v_i - v_j)^2 = 0\,.$$

For positive weights, this quadratic form equals zero if and only if $v_i = v_j$ for any two connected nodes. For a connected graph, the unique normalized vector meeting this requirement is $\mathbf{v} = \mathbf{1}\sqrt{I}$. Moreover it follows that $\mathbf{L_w}$ is positive semidefinite, since $\mathbf{v}^T\mathbf{L_w}\mathbf{v}$ cannot become negative.

In the next step we have to show that $\lambda_i(\mathbf{W}) > -1$, which is equal to $\rho\{\mathbf{L_w}\} < 2$. The Gerschgorin circle theorem [77] implies that the eigenvalues $\lambda(\mathbf{L_w})$ of $\mathbf{L_w}$ satisfy

$$\left|\lambda(\mathbf{L_w}) - d_{\mathbf{w}}(i)\right| \leq d_{\mathbf{w}}(i)\,,$$

and hence we have the bound $\lambda(\mathbf{L_w}) \leq 2d_{\mathbf{w}}(i)$. Therefore, $d_{\mathbf{w}}(i) < 1$ for all $i \in \mathcal{V}$ is a sufficient condition for convergence.

For GMH weights the weighted degree computes $d_{\mathbf{w}}(i) = \sum_{j\in\mathcal{N}(i)} \frac{1}{\max\{d_i,d_j\}+\epsilon}$ with $\epsilon \geq 0$. We have $\max\{d_i, d_j\} \geq d_i$ which in turn implies

$$d_{\mathbf{w}}(i) \leq \sum_{j\in\mathcal{N}(i)} \frac{1}{d_i + \epsilon} = \frac{d_i}{d_i + \epsilon}\,.$$

Thus, $\epsilon > 0$ is a sufficient condition for convergence because it ensures $d_{\mathbf{w}}(i) < 1$. It is seen that $\epsilon = 1$ as in conventional MH [41] is actually a very conservative choice.

In the case of constant weights we have as already mentioned $\mathbf{L_w} = \alpha\mathbf{L}$. Hence, it follows that $d_{\mathbf{w}}(i) = \alpha d_i$ and therefore $\alpha < 1/d_i$ is a sufficient condition for convergence.

4.1.2 Necessary Conditions for Convergence

We next provide necessary conditions for AC convergence and we identify the corresponding critical graph topologies.

Theorem 2. *The conditions $\epsilon > 0$ (AC/GMH) and $\alpha < 1/d_{\max}$ (AC/CW) are necessary for convergence. AC/GMH with $\epsilon = 0$ and AC/CW with $\alpha = 1/d_{\max}$ converge unless the graph is regular and bipartite.*

Proof: Lemma 1 below establishes

$$\rho\{\mathbf{L_w}\} \leq \rho\{|\mathbf{L_w}|\},$$

with equality if and only if the graph is bipartite. Furthermore, for AC/GMH with $\epsilon \geq 0$ and for AC/CW with $\alpha \leq 1/d_{\max}$, Lemma 2 below states that

$$\rho\{|\mathbf{L_w}|\} \leq 2,$$

with equality if and only if the graph is regular and $\epsilon = 0$ (AC/GMH) or $\alpha = 1/d_{\max}$ (AC/CW). Hence, for regular bipartite graphs and $\epsilon = 0$ (AC/GMH) or $\alpha = 1/d_{\max}$ (AC/CW), we have $\rho\{\mathbf{L_w}\} = 2$, which implies $\rho\{\mathbf{W}-\frac{1}{I}\mathbf{1}\mathbf{1}^{\mathrm{T}}\} = 1$ und hence divergence of AC. If the graph is neither regular nor bipartite, AC converges even with $\epsilon = 0$ (AC/GMH) and $\alpha = 1/d_{\max}$ (AC/CW) since here $\rho\{\mathbf{L_w}\} < 2$. $\square$

The next Lemma is a specialization of [78, Lemma 2.1] to weighted Laplacians.

Lemma 1: *For any weighted connected graph with strictly positive weights, the weighted Laplacian* $\mathbf{L_w}$ *satisfies*

$$\rho\{\mathbf{L_w}\} \leq \rho\{|\mathbf{L_w}|\}.$$

Equality holds if and only if the graph is bipartite.

Proof: According to [77, Theorem 8.1.18], $\rho\{\mathbf{L_w}\} \leq \rho\{|\mathbf{L_w}|\}$. Furthermore, [77, Theorem 8.4.5] states that equality holds if $|\mathbf{L_w}|$ is irreducible (in our case this is the case since the underlying graph is assumed to be connected) and there exists a diagonal matrix $\mathbf{\Lambda} = \mathrm{diag}\{e^{\iota\varphi_1}, \ldots, e^{\iota\varphi_I}\}$ such that

$$\mathbf{L_w} = e^{\iota\theta}\mathbf{\Lambda}|\mathbf{L_w}|\mathbf{\Lambda}^{-1}. \tag{4.1}$$

Here, $e^{\iota\theta}$ is the phase of the largest eigenvalue, which in our case equals plus one since $\mathbf{L_w}$ is positive semidefinite. Denoting the elements of $\mathbf{L_w}$ by $\tilde{l}_{ij}$, (4.1) can be rewritten as

$$\tilde{l}_{ij} = e^{\iota(\varphi_i - \varphi_j)}|\tilde{l}_{ij}|. \tag{4.2}$$

Due to $\mathbf{W} = \mathbf{I} - \mathbf{L_w}$, we furthermore have

$$\tilde{l}_{ij} = \begin{cases} -w_{ij}, & (i,j) \in \mathcal{E}, \\ \sum_{k\in\mathcal{N}(i)} w_{ik}, & i = j, \\ 0, & \text{else.} \end{cases}$$

Condition (4.2) is trivially satisfied for $i = j$ and for $\tilde{l}_{ij} = 0$. It remains to study the case $(i,j) \in \mathcal{E}$; here, (4.2) is equivalent to $w_{ij} = -e^{\iota(\varphi_i - \varphi_j)}|w_{ij}|$, which in turn necessitates $\varphi_i - \varphi_j = \pi \bmod 2\pi$, $(i,j) \in \mathcal{E}$. Suppose we pick an arbitrary node i_0 with associated phase φ_{i_0}. Then, $\varphi_j = \varphi_{i_0} + \pi \bmod 2\pi$ for all neighboring nodes $j \in \mathcal{N}(i_0)$. Furthermore, the neighbors $i \in \mathcal{N}(j)$ for $j \in \mathcal{N}(i_0)$, i.e., all nodes that are two hops away from i_0, must have $\varphi_i = \varphi_{i_0} \bmod 2\pi$. Continuing this argument iteratively, it follows that $\varphi_j = \varphi_{i_0} \bmod 2\pi$ if there is an even number of edges between i_0 and j and $\varphi_j = \varphi_{i_0} + \pi \bmod 2\pi$ if there is an odd number of edges between i_0 and j. This implies that there are two groups of nodes, i.e., one with $\varphi_j = \varphi_{i_0} + \pi \bmod 2\pi$ and one with $\varphi_j = \varphi_{i_0} \bmod 2\pi$, where none of the nodes in a group are neighbors. Thus, the underlying graph has to be bipartite. $\square$

The following Lemma establishes an upper bound on the spectral radius of the signless Laplacian.

Lemma 2: *For a connected graph with GMH weights $\epsilon \geq 0$ or CW with $\alpha \leq 1/d_{\max}$, the spectral radius of the the signless Laplacian is bounded as*

$$\rho\{|\mathbf{L}_\mathbf{w}|\} \leq 2.$$

Equality is obtained if and only if the graph is regular and $\epsilon = 0$ (GMH) or $\alpha = 1/d_{\max}$ (CW).

Proof: According to [77, Theorem 8.1.22 and Theorem 8.4.4], we have $\rho\{|\mathbf{L}_\mathbf{w}|\} \leq \max_i \sum_{j=1}^{I} |\tilde{l}_{ij}|$ with equality if and only if all row sums of $|\mathbf{L}_\mathbf{w}| = \operatorname{diag}\{\mathbf{A}_\mathbf{w}\mathbf{1}\} + \mathbf{A}_\mathbf{w}$ are equal. These row sums are given by $\sum_{j=1}^{I} |\tilde{l}_{ij}| = 2d_\mathbf{w}(i) = 2\sum_{j=1}^{I} w_{ij}$.

For AC/CW with $\alpha \leq 1/d_{\max}$, $d_\mathbf{w}(i) = \alpha d_i \leq 1$ and hence $\rho\{|\mathbf{L}_\mathbf{w}|\} \leq 2$. Equality holds if and only if (i) all degrees d_i are identical, i.e., $d_i = d$, and thus the graph is d-regular, and (ii) $\alpha = 1/d$.

For AC/GMH with $\epsilon \geq 0$, $\max\{d_i, d_j\} \geq d_i$ and hence $w_{ij}^{\mathrm{MH}}(\epsilon) \leq \frac{1}{d_i+\epsilon}$, which in turn implies $d_\mathbf{w}(i) \leq \frac{d_i}{d_i+\epsilon} \leq 1$. Again, $\rho\{|\mathbf{L}_\mathbf{w}|\} \leq 2$ with equality if and only if (i) $\epsilon = 0$ and (ii) the graph is regular (i.e., all degrees d_i are identical). $\square$

4.1.3 Discussion

According to the results above, $\epsilon > 0$ and $\alpha < 1/d_{\max}$ are necessary and sufficient conditions for convergence of AC/GMH and AC/CW, respectively, on arbitrary connected graphs. However, on graphs that are neither regular nor bipartite, convergence is obtained even with $\epsilon = 0$ and $\alpha = 1/d_{\max}$. Furthermore, as will be seen in the simulation results, GMH with $\epsilon = 1$ is not necessarily optimal for regular bipartite graphs. More specifically, for the case of 4-regular graphs the optimum ϵ is roughly inversely proportional to the number of nodes.

In contrast to applications like grid computing, it is rather unlikely in the context of WSN that the communication graph is regular and bipartite. If the WSN is modeled via random geometric graphs, the probability for a regular topology can be shown to decrease exponentially with increasing network size. Thus, the probability for a regular topology is non-vanishing only for very small random geometric graphs. If in addition we require bipartiteness, geometric constraints imply that the worst case are 4-regular graphs on the torus that are bipartite. In a finite two-dimensional region, 4-regular bipartite graphs are not feasible. In higher dimensions, regular bipartite graphs with larger degrees are possible but very unlikely to occur with random geometric graph models.

In summary, in small WSN the regularization parameter ϵ of AC/GMH should be chosen larger than in huge WSN. For the latter, AC/GMH with $\epsilon = 0$ will usually be superior.

4.1.4 Numerical results

In general, the convergence rate of AC/GMH depends on ϵ. To study this dependence, Fig. 4.1 shows the mean-square error (MSE) in dB versus ϵ after 30 AC iterations for different graphs and independent, identically distributed (i.i.d.) initial values s_i. In a 4-regular bipartite graph with 16 nodes, the optimum regularization

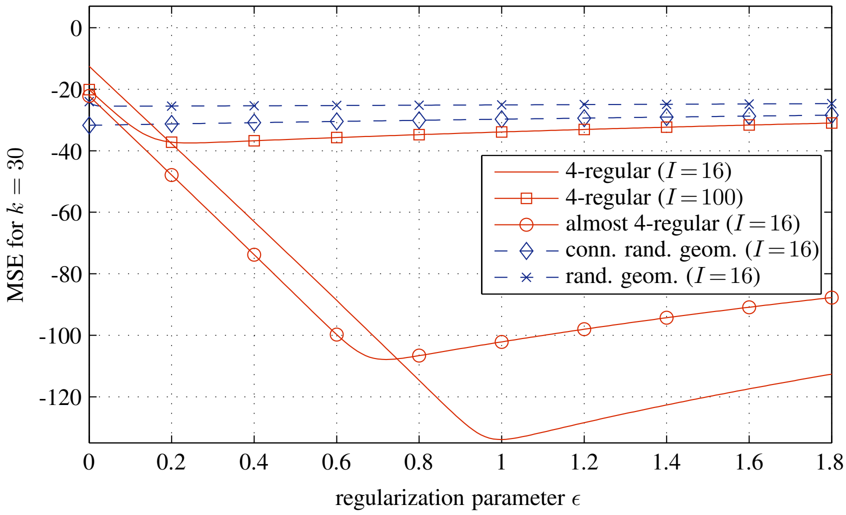


Figure 4.1: *MSE (in dB) after* 30 *iterations versus GMH parameter* ϵ *for different graph topologies; initial measurements were i.i.d.*

turns out to be $\epsilon \approx 1$. In the case of the regular bipartite graph with 100 nodes, however, the minimum MSE is obtained with $\epsilon \approx 0.25$. These results agree with the analytical findings regarding optimal CW in the asymptotic regime [19]. For an "almost regular graph," obtained by deleting one edge from the 16-node 4-regular bipartite graph, the optimum MSE is achieved with $\epsilon \approx 0.7$. In the case of connected random geometric graphs with $c = 1.3$, (almost) regular topologies are highly unlikely; here, the optimum is at $\epsilon = 0$, i.e., a non-zero regularization term only slows down convergence. For possibly unconnected random geometric graphs (resulting, e.g., from a smaller communication radius), the MSE depends weakly on the GMH parameter ϵ, with a poorly pronounced optimum at 0; this can be attributed to the fact that here smaller unconnected components (mostly consisting of 2 nodes) arise in the graph. We also note that in all our simulations choosing $\epsilon > 1$ never lead to performance improvements.

We next compare the performance of the two extreme cases of GMH, i.e., $\epsilon = 0$ and $\epsilon = 1$. We consider random geometric graphs with $I = 100$ nodes in a square region $\mathcal{A}$ which is periodic. The initial values s_i were again i.i.d. Fig. 4.2 shows the median MSE advantage in AC/GMH obtained with $\epsilon = 0$ relative to $\epsilon = 1$ as a function of network connectivity c (which is proportional to the average number of neighbors). Results are shown for $k = 20$ and $k = 70$ iterations and for static and dynamic networks. In static networks, the topology and hence the GMH weights do not change over time. In the dynamic case, 20 nodes randomly changed their position during each time step, resulting in time-varying GMH weights (further details on this node mobility model and its impact on the averaging performance is provided in Chapter 5). It is seen that in this scenario, $\epsilon = 0$ is superior to $\epsilon = 1$ in all operating conditions except for poorly connected static networks, where $\epsilon = 1$ is slightly superior after 70 iterations. In the static case, the MSE advantage increases with increasing connectivity c whereas in the dynamic case the MSE advantage is almost independent of c. For strongly connected graphs ($c = 2.8$), $\epsilon = 0$ performs better than $\epsilon = 1$ after 70 iterations by about 18 dB (static networks) and 11 dB (dynamic networks). This can be attributed to the fact that for well-connected graphs it is highly unlikely to have (almost) regular or bipartite topologies. We performed the same experiments also with (spatially)

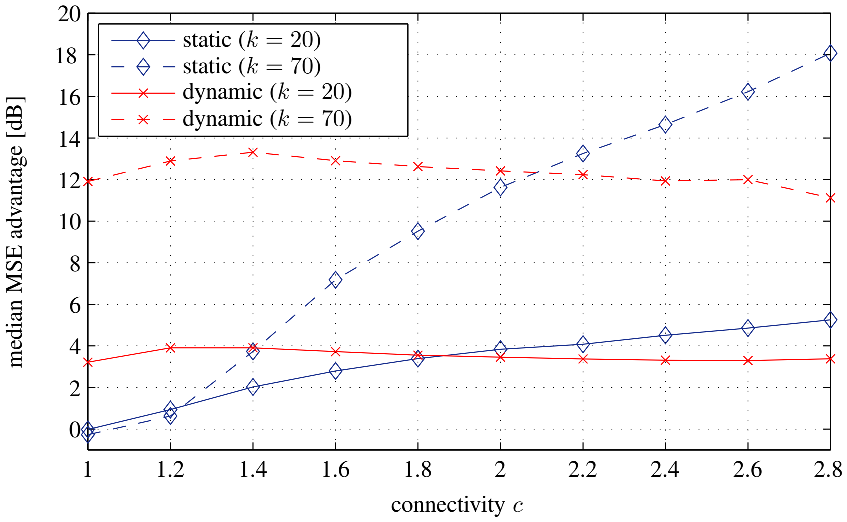


Figure 4.2: *Median MSE advantage of AC/GMH with* $\epsilon = 0$ *relative to* $\epsilon = 1$ *versus connectivity* c *after* 20 *and* 70 *iterations in static and dynamic random geometric graphs with* $I = 100$ *nodes.*

correlated initial values. In this case we observed that $\epsilon = 0$ is superior even at low connectivity ($c \approx 1$).

4.2 Optimum Symmetric Weights

The main contribution of this section is a step-wise MSE-optimal weight design method. For that purpose we first show how to compute the MSE for a given second-order statistics of the initial measurements (i.e. the measurement correlation and therefore inner-state correlation). Then we derive a closed-form solution for the weights that minimize the MSE for the upcoming AC iteration. Because we allow for a different weight matrix $\mathbf{W}[k]$ to be used in each AC update (cf. (2.3)), the optimization method can be applied every iteration. The corresponding inner-state correlation can be computed easily due to the linear structure of AC. We assume symmetric doubly stochastic weight matrices, i.e., $\mathbf{W}^T[k] = \mathbf{W}[k]$ and $\mathbf{W}[k]\mathbf{1} = \mathbf{1}$. Hence, the spatial mean is preserved during the iterations, i.e., $\mathbf{1}^T\mathbf{x}[k] = I\bar{s}$ and therefore $\mathbf{J}\mathbf{x}[k] = \mathbf{J}\mathbf{x}[0] = \bar{s}\mathbf{1}$ where $\mathbf{J} = \frac{1}{I}\mathbf{1}\mathbf{1}^T$. In contrast to the previous section, we allow the weights to be negative in the following.

4.2.1 MSE Derivation

A reasonable metric for the difference between the actual mean and the AC estimate $\mathbf{x}[k]$ is given by the MSE (cf. Section 2.2.5)

$$\epsilon^2[k] \triangleq \mathrm{E}\left\{ \left\| \mathbf{x}[k] - \bar{s}\mathbf{1} \right\|^2 \right\}.$$

We omit constant factors (e.g., normalization), since it is not necessary for the subsequent optimization. The MSE will be seen below to have the advantage of leading to a framework amenable to a closed-form solution. Using $\mathbf{x}[k+1] = \mathbf{W}[k]\mathbf{x}[k]$ and $\bar{s} = \frac{1}{I}\mathbf{1}^T\mathbf{x}[0]$, the MSE at time $k+1$ can be developed as

$$\epsilon^2[k+1] \triangleq \mathrm{E}\left\{ \left\| \mathbf{W}[k]\mathbf{x}[k] - \mathbf{J}\mathbf{x}[0] \right\|^2 \right\}$$

$$\begin{aligned}
&= \mathrm{E}\Big\{\big\|\mathbf{W}[k]\mathbf{x}[k] - \mathbf{J}\mathbf{x}[k]\big\|^2\Big\} \\
&= \mathrm{E}\Big\{\big\|\big(\mathbf{W}[k] - \mathbf{J}\big)\mathbf{x}[k]\big\|^2\Big\} \\
&= \mathrm{E}\Big\{\operatorname{tr}\Big\{\big(\mathbf{W}[k]-\mathbf{J}\big)\mathbf{x}[k]\mathbf{x}^T[k]\big(\mathbf{W}[k]-\mathbf{J}\big)^T\Big\}\Big\} \\
&= \operatorname{tr}\Big\{\big(\mathbf{W}[k]-\mathbf{J}\big)\mathbf{R}_{\mathbf{x}}[k]\big(\mathbf{W}[k]-\mathbf{J}\big)^T\Big\}\,,
\end{aligned} \tag{4.3}$$

where in the second step we used the mean-preservation property and $\operatorname{tr}\{\cdot\}$ denotes the matrix trace; furthermore, in the last step we defined $\mathbf{R}_{\mathbf{x}}[k] = \mathrm{E}\{\mathbf{x}[k]\mathbf{x}^T[k]\}$. The expression (4.3) shows that the MSE only depends on the correlation matrix and the weight matrix (and, via the weight matrix, implicitly on the graph topology). If the states are i.i.d., i.e., $\mathbf{R}_{\mathbf{x}}[k] = \mathbf{I}$, the MSE simplifies to

$$\epsilon^2[k+1] = \|\mathbf{W}[k] - \mathbf{J}\|_{\mathrm{F}}^2 = \sum_i \lambda_i^2[k]\,,$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm and λ_i is the ith eigenvalue of $\mathbf{W}[k] - \mathbf{J}$. This special case was already considered in [44].

4.2.2 MSE Minimization

Our goal is to choose the weight matrix $\mathbf{W}[k]$ to minimize the MSE in (4.3), thereby achieving the best step-by-step performance in the MSE sense. Repeating this optimization for every iteration step k yields a sequence of weight matrices that optimize the overall performance. The optimization problem of interest can be written as

$$\begin{aligned}
\mathbf{W}[k] &= \arg\min_{\mathbf{W}} \epsilon^2[k+1] \\
&\text{subject to } \mathbf{W}\mathbf{1} = \mathbf{1},\ \mathbf{W}^T = \mathbf{W}\,,
\end{aligned} \tag{4.4}$$

where the matrix $\mathbf{W}[k]$ has to possess the same zero pattern as the adjacency matrix, except for the diagonal elements. This constrained optimization problem is convex and can be solved with standard numerical optimization tools (e.g. [76]) as is done in [44] for the uncorrelated case. We next derive a closed-form solution for the general case.

Replacing the weight matrix with its decomposition in (2.4), i.e.,

$$\mathbf{W} = \mathbf{I} - \mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T\,, \tag{4.5}$$

the side constraints are satisfied for any choice of $\mathbf{w}$ and thus the constrained optimization problem (4.4) with respect to $\mathbf{W}$ is thereby converted into an unconstrained optimization problem with respect to $\mathbf{w}$.

We next rewrite the cost function $\epsilon^2[k]$ as

$$\begin{aligned}
\epsilon^2[k+1] &= \operatorname{tr}\Big\{\big(\mathbf{W}[k]-\mathbf{J}\big)\mathbf{R}_{\mathbf{x}}[k]\big(\mathbf{W}[k]-\mathbf{J}\big)^T\Big\} \\
&= \operatorname{tr}\Big\{\mathbf{W}[k]\mathbf{R}_{\mathbf{x}}[k]\mathbf{W}^T[k] - \mathbf{J}\mathbf{R}_{\mathbf{x}}[k]\mathbf{W}^T[k] \\
&\qquad - \mathbf{W}[k]\mathbf{R}_{\mathbf{x}}[k]\mathbf{J}^T + \mathbf{J}\mathbf{R}_{\mathbf{x}}[k]\mathbf{J}^T\Big\}
\end{aligned}$$

$$= \operatorname{tr}\left\{\mathbf{W}[k]\mathbf{R}_{\mathbf{x}}[k]\mathbf{W}^T[k]\right\} - c^2\,, \tag{4.6}$$

where

$$c^2 = \operatorname{tr}\left\{\mathbf{J}\mathbf{R}_{\mathbf{x}}[k]\mathbf{J}^T\right\} = \mathrm{E}\left\{\operatorname{tr}\left\{\mathbf{J}\mathbf{x}[k]\mathbf{x}^T[k]\mathbf{J}^T\right\}\right\} = \mathrm{E}\{\bar{s}^2\}\, I\,,$$

is independent of $\mathbf{W}$. Inserting (4.5) into (4.6) we obtain

$$\begin{aligned}\epsilon^2[k{+}1] &= \operatorname{tr}\left\{\mathbf{I}\mathbf{R}_{\mathbf{x}}[k]\mathbf{I}\right\} - 2\operatorname{tr}\left\{\mathbf{R}_{\mathbf{x}}[k]\mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T\right\}\\ &\quad + \operatorname{tr}\left\{\mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T\mathbf{R}_{\mathbf{x}}[k]\mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T\right\} - c^2\\ &= \operatorname{tr}\left\{\operatorname{diag}\{\mathbf{w}\}\,\mathbf{R}_{\mathbf{y}}\operatorname{diag}\{\mathbf{w}\}\mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}}\right\}\\ &\quad - 2\operatorname{tr}\{\operatorname{diag}\{\mathbf{w}\}\mathbf{R}_{\mathbf{y}}\} + \operatorname{tr}\{\mathbf{R}_{\mathbf{x}}[k]\} - c^2,\end{aligned}$$

where we defined $\mathbf{R}_{\mathbf{y}} \triangleq \mathbf{B}^T\mathbf{R}_{\mathbf{x}}[k]\mathbf{B}$ and $\mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}} \triangleq \mathbf{B}^T\mathbf{B}$ (the iteration index k of $\mathbf{R}_{\mathbf{y}}$ is omitted for simplicity). The detailed structure of $\mathbf{R}_{\mathbf{y}}$ and $\mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}}$ is discussed later, but we point out already here that both are positive definite. This last expression can be minimized by setting its first-order derivative to zero, i.e.,

$$\frac{d\epsilon^2[k{+}1]}{d\mathbf{w}} = 0\,,$$

Writing out the MSE in terms of the elements of $\operatorname{diag}\{\mathbf{w}\}$, $\mathbf{R}_{\mathbf{y}}$, and $\mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}}$ yields

$$\epsilon^2[k{+}1] = \sum_{i,j} w_i[\mathbf{R}_{\mathbf{y}}]_{ij}w_j[\mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}}]_{ji} - 2\sum_i(w_i[\mathbf{R}_{\mathbf{y}}]_{ii} + [\mathbf{R}_{\mathbf{x}}[k]]_{ii}) - c^2,$$

which, after differentiation, results in the linear equations

$$\frac{d\epsilon^2[k{+}1]}{dw_i} = 2\sum_j[\mathbf{R}_{\mathbf{y}}]_{ij}[\mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}}]_{ij}w_j - 2[\mathbf{R}_{\mathbf{y}}]_{ii} = 0\,.$$

Equivalently, we have

$$(\mathbf{R}_{\mathbf{y}} \circ \mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}})\mathbf{w} = \operatorname{diag}\{\mathbf{R}_{\mathbf{y}}\}\,,$$

with the shorthand notation $\mathbf{r}_{\mathbf{y}} = \operatorname{diag}\{\mathbf{R}_{\mathbf{y}}\}$ and $\tilde{\mathbf{R}}_{\mathbf{y}} = \mathbf{R}_{\mathbf{y}} \circ \mathbf{R}_{\mathbf{y}}^{\text{i.i.d.}}$ we obtain the linear system of equations

$$\tilde{\mathbf{R}}_{\mathbf{y}}\mathbf{w} = \mathbf{r}_{\mathbf{y}}\,. \tag{4.7}$$

The symbol $\circ$ denotes the direct (element-wise) product, also known as Hadamard product. The solution of (4.7) is given by

$$\mathbf{w}^\star = \tilde{\mathbf{R}}_{\mathbf{y}}^{-1}\mathbf{r}_{\mathbf{y}}$$

and can be computed e.g. using the Cholesky factorization since $\tilde{\mathbf{R}}_{\mathbf{y}}$ is positive definite. This follows from the fact that the Hadamard product of two positive definite matrices is positive definite as well. The optimum weight matrix is then obtained by inserting $\mathbf{w}^\star$ into (4.5).

We note that the AC state update $\mathbf{x}[k{+}1] = \mathbf{W}[k]\mathbf{x}[k]$ implies $\mathbf{R}_{\mathbf{x}}[k{+}1] = \mathbf{W}[k]\mathbf{R}_{\mathbf{x}}[k]\mathbf{W}^T[k]$ and hence by induction

$$\mathbf{R}_{\mathbf{x}}[k{+}1] = \mathbf{W}[k]\ldots\mathbf{W}[0]\mathbf{R}_{\mathbf{x}}[0]\mathbf{W}^T[0]\ldots\mathbf{W}^T[k].$$

Thus, successive optimization of the weight matrices requires only the initial correlation $\mathbf{R_x}[0] = \mathbf{R_s}$ of the measurements to be known.

Since the states $x_i[k]$ will become more and more equal as k increases, their correlation matrix tends towards a scalar multiple of the rank-one matrix $\mathbf{J}$ and (4.7) becomes numerically unstable. To mitigate numerical problems, the weight optimization scheme is modified by replacing $\mathbf{R_x}[k]$ with $\mathbf{R_x}[k] - \mathrm{tr}\{\mathbf{R_x}[k]\mathbf{J}\}\mathbf{J}$. This replacement causes only an additive shift in the MSE cost function (4.6) and hence does not change the solution for the optimum weights.

In general, $\mathbf{W}^*[k]$ is not guaranteed to be a stable solution for AC, i.e., freezing the weight matrix $\mathbf{W}[k'] = \mathbf{W}^*[k]$, $k' > k$, may lead to unstable behavior. While it is not possible to compute stable MSE-optimal weights in closed form, they can in principle be obtained by solving the following constrained convex optimization problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\tilde{\mathbf{R}}_\mathbf{y}\mathbf{w} - \mathbf{r_y}\|^2 \qquad (4.8)$$
$$\text{subject to} \quad \rho\big(\mathbf{W} - \mathbf{J}\big) < 1\,.$$

In the next paragraphs we will discuss the structure of $\mathbf{R_y}$ and $\mathbf{R}_\mathbf{y}^{\text{i.i.d.}}$, respectively. First of all, let us define $y_l = x_i - x_j$ where l indicates the edge $(i,j) \in \mathcal{E}$ (we are going to omit the time index k in the following). Hence, y_l represents the difference of the inner states of two adjacent nodes connected by the edge l. Therefore, if we stack the y values for all edges into a vector $\mathbf{y}$ we can write $\mathbf{y} = \mathbf{B}^T\mathbf{x}$. It is seen, that the sign of the elements of $\mathbf{y}$ depend on the choice of the signs of the incidence matrix elements (we can either choose $y_l = x_i - x_j$ or $y_l = x_j - x_i$).

The correlation matrix of $\mathbf{y}$ equals,

$$\mathbf{R_y} = \mathrm{E}\{\mathbf{y}\mathbf{y}^T\} = \mathrm{E}\{\mathbf{B}^T\mathbf{x}\mathbf{x}^T\mathbf{B}\} = \mathbf{B}^T\mathbf{R_x}\mathbf{B}\,,$$

which justifies the usage of $\mathbf{R_y}$ some paragraphs above. In the case when we have an i.i.d. inner state vector with unit power, $\mathbf{R}_\mathbf{x}^{\text{i.i.d.}} = \mathbf{I}$ and therefore $\mathbf{R}_\mathbf{y}^{\text{i.i.d.}} = \mathbf{B}^T\mathbf{B}$.

In the following we are going to investigate the ingredients of (4.7) in more detail. First of all, we have

$$\big[\mathbf{R_y} \circ \mathbf{R}_\mathbf{y}^{\text{i.i.d.}}\big]_{lp} = \begin{cases} 2(r_x(i,i) + r_x(j,j) - 2r_x(i,j))\,, & l = p\,, \\ r_x(i,j) + r_x(j,k) - r(j,j) - r(i,k)\,, & l \neq p \text{ and } i,k \in \mathcal{N}(j) \text{, and} \\ 0\,, & \text{else}\,, \end{cases}$$

with $r_x(i,i) = \mathrm{E}\{x_i x_j\}$. Moreover, l denotes the edge (i,j) and p is the edge between node k and j. It is seen, that the dependency on the choice of the sign in the incidence matrix exists no more. This is also true for the right hand side of (4.7), i.e.,

$$[\mathbf{r_y}]_l = \mathrm{E}\{y_l y_l\} = r_x(i,i) + r_x(j,j) - 2r_x(i,j)\,,$$

where nodes i and j are linked by edge l. Hence, the optimum weights do not depend on the choice of the signs in the incidence matrix, which is also necessary because the directivity of the edges should play no role in AC.

It is possible to investigate the MSE of $\mathbf{y}$ instead of that of $\mathbf{x}$ and derive the optimum weights, but we obtain an underdetermined system where (4.7) is contained in the solution set.

4.2.3 Uncorrelated Measurements

In the following we consider the special case of uncorrelated measurements, i.e., $\mathbf{R}_{\mathbf{x}}[0] = \mathbf{R}_{\mathbf{s}} = \gamma\mathbf{I}$. Here, it can be shown that (4.7) simplifies to

$$(\mathbf{A}(\mathcal{L}) + 4\mathbf{I})\mathbf{w} = 2\mathbf{1}\,, \tag{4.9}$$

where $\mathbf{A}(\mathcal{L})$ denotes the adjacency matrix of the line graph $\mathcal{L}$ (also called adjoint graph) of $\mathbf{A}$ (see [58] for more details). It is seen that by minimizing the MSE (4.6), the weights are invariant to the variance γ of the measurements, which is intuitive. In practice uncorrelated measurements can only occur in the initial state, but the resulting weights can be also used as static weights for AC.

In our simulations we observed that for the uncorrelated case we mostly obtained positive weights. For some graphs this can be proved by requiring diagonal dominance in (4.9). In particular, [79] implies that a sufficient condition for the solution of (4.9) to be positive is that the sum of the non-diagonal elements in each row of $(\mathbf{A}(\mathcal{L}) + 4\mathbf{I})$ is smaller than the diagonal element (which here equals 4). This is guaranteed for graphs where the sum of the degrees of any two neighboring nodes is less than 5 and hence the maximum degree in the associated line graph is less than 4. We found empirically that this sufficient condition is rather conservative, i.e., the majority of graphs that does not satisfy this condition still leads to positive weights. So far, we are aware of only few types of graph for which $\mathbf{w}$ has non-positive elements: sufficiently large star graphs with additional edges between leaf nodes or a graph which consists of two star graphs where the center nodes are connected).

A positive $\mathbf{w}$ ensures that $\mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T$ is positive-semidefinite. Hence, for positive weights the eigenvalues of $\mathbf{W} - \mathbf{J}$ are all smaller than one (cf. (4.5)). Initial results and simulations suggest that the smallest eigenvalue of $\mathbf{W} - \mathbf{J}$ is always larger than -1 (thereby rendering $\mathbf{W}$ stable). However, we do not have a general proof for this conjecture so far. As an example, the smallest eigenvalue of the star graph can be shown to equal $\frac{2-I}{2+I} > -1$ (this is the eigenvalue closest to -1 that we observed for graphs that result in positive weights).

4.2.4 Numerical Solver

In the following we show how to design a numerical solver for our optimization problem that yields a stable weight vector and achieves the best possible per-step performance. Our optimization problem reads (cf. (4.8))

$$\begin{aligned} &\text{minimize} && \left\|\tilde{\mathbf{R}}_{\mathbf{y}}\mathbf{w} - \mathbf{r}_{\mathbf{y}}\right\|_2 \\ &\text{subject to} && \rho\left\{\mathbf{I} - \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T - \tfrac{1}{I}\mathbf{1}\mathbf{1}^T\right\} < 1\,. \end{aligned}$$

This problem can be rewritten in the following equivalent form,

$$\begin{aligned} &\text{minimize} && f_0(\mathbf{w}) = \mathbf{w}^T\tilde{\mathbf{R}}_{\mathbf{y}}^2\mathbf{w} - 2\mathbf{r}_{\mathbf{y}}^T\tilde{\mathbf{R}}_{\mathbf{y}}\mathbf{w} \\ &\text{subject to} && \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T + \tfrac{1}{I}\mathbf{1}\mathbf{1}^T \succ 0 \\ & && 2\mathbf{I} - \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T \succ 0\,, \end{aligned} \tag{4.10}$$

where $\mathbf{M} \succ 0$ means that $\mathbf{M}$ has to be positive definite. It is seen that this problem is quadratic and the feasible region is defined by generalized inequalities. To solve such a problem, we can use a barrier method.

Since our side constraints require positive definiteness we use a generalized logarithm, in particular we use $\psi\{\mathbf{X}\} = \log\det\{\mathbf{X}\}$ which is a generalized logarithm for the cone of the positive semidefinite matrices (we refer to [80] for more details). Hence, we can reformulate (4.10) as an unconstrained problem,

$$\text{minimize} \;\; f(\mathbf{w}) = tf_0(\mathbf{w}) + \phi(\mathbf{w})\,,$$

where $\phi(\mathbf{w}) = -\psi\left\{\mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T + \frac{1}{I}\mathbf{1}\mathbf{1}^T\right\} - \psi\left\{2\mathbf{I} - \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T\right\}$ and t describes the similarity of the problem to the constrained form. If $t \rightarrow \infty$ the unconstrained problem is equal to the constrained version, but numerically hard to solve. Therefore, we start with a moderate value of t and solve the corresponding problem and use the optimal value of $\mathbf{w}$ for the next problem instance with increased t.

To obtain a similar structure as in [80] and to increase the readability we substitute $\mathbf{C}_1(\mathbf{w}) = \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T + \frac{1}{I}\mathbf{1}\mathbf{1}^T$ and $\mathbf{C}_2(\mathbf{w}) = 2\mathbf{I} - \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T$ and obtain

$$\text{minimize} \;\; tf_0 + \sum_{m=1}^{2} -\psi\left\{\mathbf{C}_m(\mathbf{w})\right\}\,.$$

For a given t we can solve this problem by the Newton method [80]. To that end, we next compute the gradient and Hessian for our problem.

Gradient

The computation of the gradient of the quadratic function $f_0(\mathbf{w})$ is trivial and therefore not discussed in the following. For the generalized logarithm the computation is a bit more complex. First of all we have

$$\frac{\partial \log\det\{\mathbf{C}_m(\mathbf{w})\}}{\partial w_i} = \mathrm{tr}\left\{\mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial \mathbf{C}_m(\mathbf{w})}{\partial w_i}\right\}\,. \tag{4.11}$$

The inner derivatives equal

$$\begin{aligned}
\frac{\partial \mathbf{C}_1(\mathbf{w})}{\partial w_i} &= \frac{\partial\left(\mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T + \frac{1}{I}\mathbf{1}\mathbf{1}^T\right)}{\partial w_i} = \mathbf{B}\,\mathrm{diag}\{\mathbf{e}_i\}\mathbf{B}^T\,,\\
\frac{\partial \mathbf{C}_2(\mathbf{w})}{\partial w_i} &= \frac{\partial\left(2\mathbf{I} - \mathbf{B}\,\mathrm{diag}\{\mathbf{w}\}\mathbf{B}^T\right)}{\partial w_i} = -\mathbf{B}\,\mathrm{diag}\{\mathbf{e}_i\}\mathbf{B}^T\,.
\end{aligned}$$

Inserting the inner derivatives into (4.11) we obtain

$$\begin{aligned}
\frac{\partial \log\det\{\mathbf{C}_1(\mathbf{w})\}}{\partial w_i} &= \mathrm{tr}\left\{\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\,\mathrm{diag}\{\mathbf{e}_i\}\mathbf{B}^T\right\}\,,\\
\frac{\partial \log\det\{\mathbf{C}_2(\mathbf{w})\}}{\partial w_i} &= -\mathrm{tr}\left\{\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\,\mathrm{diag}\{\mathbf{e}_i\}\mathbf{B}^T\right\}\,.
\end{aligned}$$

Finally, we can simplify these expressions by developing the trace, i.e.,

$$\begin{aligned}
\mathrm{tr}\left\{\mathbf{C}_m^{-1}(\mathbf{w})\mathbf{B}\,\mathrm{diag}\{\mathbf{e}_i\}\mathbf{B}^T\right\} &= \sum_{j,k}\left[\mathbf{C}_m^{-1}(\mathbf{w})\right]_{kj}[\mathbf{B}]_{ji}[\mathbf{B}]_{ik}\\
&= \sum_{j,k}[\mathbf{B}]_{ki}\left[\mathbf{C}_m^{-1}(\mathbf{w})\right]_{kj}[\mathbf{B}]_{ji}\\
&= [\mathbf{B}\mathbf{C}_m^{-1}(\mathbf{w})\mathbf{B}]_{ii}\,.
\end{aligned}$$

Hence, we obtain for the individual gradients

$$\begin{aligned}
\nabla f_0(\mathbf{w})t &= \left(2\tilde{\mathbf{R}}_{\mathbf{y}}^2\mathbf{w} - 2\mathbf{r}_{\mathbf{y}}^T\tilde{\mathbf{R}}_{\mathbf{y}}\right)t\,, \\
\nabla\psi\left\{\mathbf{C}_1(\mathbf{w})\right\} &= \operatorname{diag}\left\{\mathbf{B}^T\left(\mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T + \tfrac{1}{I}\mathbf{1}\mathbf{1}^T\right)^{-1}\mathbf{B}\right\}\,, \\
\nabla\psi\left\{\mathbf{C}_2(\mathbf{w})\right\} &= -\operatorname{diag}\left\{\mathbf{B}^T\left(2\mathbf{I} - \mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T\right)^{-1}\mathbf{B}\right\}\,,
\end{aligned}$$

which further yields

$$\begin{aligned}
\nabla f(\mathbf{w}) = \left(2\tilde{\mathbf{R}}_{\mathbf{y}}^2\mathbf{w} - 2\mathbf{r}_{\mathbf{y}}^T\tilde{\mathbf{R}}_{\mathbf{y}}\right)t &- \operatorname{diag}\left\{\mathbf{B}^T\left(\mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T + \tfrac{1}{I}\mathbf{1}\mathbf{1}^T\right)^{-1}\mathbf{B}\right\} \\
&+ \operatorname{diag}\left\{\mathbf{B}^T\left(2\mathbf{I} - \mathbf{B}\operatorname{diag}\{\mathbf{w}\}\mathbf{B}^T\right)^{-1}\mathbf{B}\right\}\,,
\end{aligned}$$

for the gradient of our optimization problem.

Hessian

The Hessian is computed in a similar manner. Again, the quadratic term is trivial to develop. For the other terms, we compute the partial derivatives as

$$\begin{aligned}
\frac{\partial^2 \log\det\{\mathbf{C}_m(\mathbf{w})\}}{\partial w_i \partial w_j} = \frac{\partial \operatorname{tr}\left\{\mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial \mathbf{C}_m(\mathbf{w})}{\partial w_i}\right\}}{\partial w_j} &= \operatorname{tr}\left\{\frac{\partial \mathbf{C}_m^{-1}(\mathbf{w})}{\partial w_j}\frac{\partial \mathbf{C}_m(\mathbf{w})}{\partial w_i} + \mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial^2 \mathbf{C}_m(\mathbf{w})}{\partial w_i \partial w_j}\right\} \\
&= \operatorname{tr}\left\{-\mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial \mathbf{C}_m(\mathbf{w})}{\partial w_j}\mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial \mathbf{C}_m(\mathbf{w})}{\partial w_i} + \mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial^2 \mathbf{C}_m(\mathbf{w})}{\partial w_i \partial w_j}\right\} \\
&= -\operatorname{tr}\left\{\mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial \mathbf{C}_m(\mathbf{w})}{\partial w_j}\mathbf{C}_m^{-1}(\mathbf{w})\frac{\partial \mathbf{C}_m(\mathbf{w})}{\partial w_i}\right\}\,.
\end{aligned}$$

We can next plug in the results from the gradient and develop the trace to simplify the Hessian

$$\begin{aligned}
-\operatorname{tr}&\left\{\mathbf{C}_m^{-1}(\mathbf{w})\mathbf{B}\operatorname{diag}\{\mathbf{e}_i\}\mathbf{B}^T\mathbf{C}_m^{-1}(\mathbf{w})\mathbf{B}\operatorname{diag}\{\mathbf{e}_j\}\mathbf{B}^T\right\} \\
&= -\sum_{k,l,n,o}\left[\mathbf{C}_m^{-1}(\mathbf{w})\right]_{kl}[\mathbf{B}]_{li}[\mathbf{B}]_{ni}\left[\mathbf{C}_m^{-1}(\mathbf{w})\right]_{no}[\mathbf{B}]_{oj}[\mathbf{B}]_{kj} \\
&= -\sum_{k,l}[\mathbf{B}]_{kj}\left[\mathbf{C}_m^{-1}(\mathbf{w})\right]_{kl}[\mathbf{B}]_{li}\sum_{n,o}[\mathbf{B}]_{ni}\left[\mathbf{C}_m^{-1}(\mathbf{w})\right]_{no}[\mathbf{B}]_{oj} \\
&= -\left[\mathbf{B}^T\mathbf{C}_m^{-1}(\mathbf{w})\mathbf{B}\right]_{ij}^2\,.
\end{aligned}$$

Therefore, the individual results of the Hessian are

$$\begin{aligned}
\nabla^2 f_0(\mathbf{w})t &= 2\tilde{\mathbf{R}}_{\mathbf{y}}^2 t\,, \\
\nabla^2\psi\left\{\mathbf{C}_1(\mathbf{w})\right\} &= -\left(\mathbf{B}^T\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\right)\circ\left(\mathbf{B}^T\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\right)\,, \\
\nabla^2\psi\left\{\mathbf{C}_2(\mathbf{w})\right\} &= -\left(\mathbf{B}^T\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\right)\circ\left(\mathbf{B}^T\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\right)\,.
\end{aligned}$$

Putting the components together we obtain for the Hessian of our target function

$$\nabla^2 f(\mathbf{w}) = 2\tilde{\mathbf{R}}_{\mathbf{y}}^2 t + \left(\mathbf{B}^T\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\right)\circ\left(\mathbf{B}^T\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\right) + \left(\mathbf{B}^T\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\right)\circ\left(\mathbf{B}^T\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\right)\,.$$

Outer Loop

As already mentioned we need to solve several optimization problems with different values of t. As a starting value we choose $t_0 = 20$. After each optimization, the new value of t is obtained by multiplying the current value with a factor $\mu > 1$. This factor should not be too large. A choice could be $\mu = 12$. When t is sufficiently large (e.g., $4 \cdot 10^5$) we stop the iterations. The choice $t > 4 \cdot 10^5$ would lead a duality gap of $5 \cdot 10^{-6}$ in which the optimal solution lies (see [80] for more details).

Newton Method

Each individual optimization problem is solved with the Newton method. For this method we need to compute the Newton step for each temporary solution:

$$\begin{aligned}\Delta_{\text{nt}} &= -\left(\nabla^2 f(\mathbf{w})\right)^{-1} \nabla f(\mathbf{w}) \\ &= -\Big(2\tilde{\mathbf{R}}_{\mathbf{y}}^2 t + \left(\mathbf{B}^T\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\right) \circ \left(\mathbf{B}^T\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\right) + \left(\mathbf{B}^T\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\right) \circ \left(\mathbf{B}^T\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\right)\Big)^{-1} \\ &\quad \Big(\Big(2\tilde{\mathbf{R}}_{\mathbf{y}}^2\mathbf{w} - 2\mathbf{r}_{\mathbf{y}}^T\tilde{\mathbf{R}}_{\mathbf{y}}\Big) t - \operatorname{diag}\left\{\mathbf{B}^T\mathbf{C}_1^{-1}(\mathbf{w})\mathbf{B}\right\} + \operatorname{diag}\left\{\mathbf{B}^T\mathbf{C}_2^{-1}(\mathbf{w})\mathbf{B}\right\}\Big)\ .\end{aligned}$$

This Newton step points into a direction in which we are searching for a minimum (this minimum coincides with the minimum of the individual optimization problem only if the Newton step points into its direction). If we have found the minimum, we have a temporary solution and compute the next Newton step. A widely used method to find the minimum is the backtracking line search (cf. [80, Section 9.2]). It does not provide us with the exact result and hence requires more Newton steps, but reduces the complexity compared to the exact analytical search. Parameters for the backtracking methods can be for instance $\alpha = 0.3$ and $\beta = 0.8$. After each Newton step, we compute the Newton decrement $\frac{\lambda^2}{2}$ which represents the accuracy of the temporary result in relation to the true result. If the Newton decrement lies for instance below 10^{-5} the Newton method can be stopped.

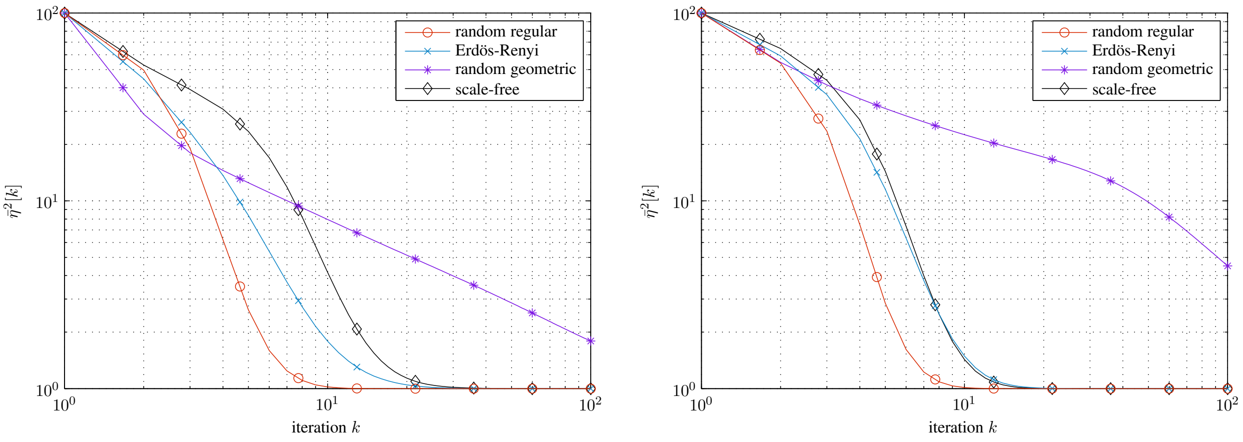
4.2.5 Numerical Results

The MSE-optimal convergence of AC is provided in Section 4.5, where we compare it with different weight design methods. In this section we investigate the correlation of the inner-state vector, which is a major ingredient for the optimization to obtain the optimal per-step MSE weights.

With AC, the states $x_i[k]$ become more and more equal as the iterations progress. This implies that the correlation between the state variables increases. Fast convergence is indicated by a rapid buildup of strong correlation. We investigate this aspect using uncorrelated measurements, i.e., $\mathbf{R}_{\mathbf{s}} = \mathbf{I}$. As a correlation measure we use the eigenvalue spread of the correlation matrix $\mathbf{R}_{\mathbf{x}}[k]$,

$$\eta^2[k] \triangleq \frac{\left(\sum_{i=1}^{I} \lambda_i[k]\right)^2}{\sum_{i=1}^{I} \lambda_i^2[k]}\,,$$

with $\lambda_i[k]$ denoting the ith eigenvalue of $\mathbf{R}_x[k]$. The metric $\eta^2[k]$ equals I in case all states are uncorrelated and 1 in case of full correlation. To suppress outliers, we averaged $\eta^2[k]$ over 100 scenarios, resulting in $\bar{\eta}^2[k]$.



(a) MH weights (b) CVX weights

Figure 4.3: *State correlation $\bar{\eta}^2[k]$ versus the number of AC iterations for different graph types: (a) MH weights, (b) CVX weights.*

Fig. 4.3(a) shows $\bar{\eta}^2[k]$ versus the number of AC iterations. For this setting we used MH weights for random regular, Erdös-Renyi (ER) [81], random geometric, and scale free graphs. All graphs have 100 nodes and approximately 300 edges. It is seen that random regular graphs are first to achieve almost full correlation. This can be explained by the regular structure and the small diameter of such graphs. ER graphs and scale-free graphs behave very similarly because both have small diameters, but are less homogenous than random regular graphs. The correlation decrease for random geometric graphs scales as $\frac{1}{k}$, i.e., has slowest convergence to full correlation. This is because these graphs have large diameters and hence require many hops to traverse the network.

A slightly different behavior is obtained for CVX weights as shown in Fig. 4.3(b). For all graphs, $\bar{\eta}^2[k]$ converges to full correlation faster than with MH weights, but the initial decrease is slower than for MH weights. Otherwise the observations made for MH weights apply to CVX weights as well.

Hence, it is reasonable to adopt the weights that are suitable for the corresponding correlation. Since the proposed method in this section is hard to implement in practice, we present a heuristic method in the following.

4.3 Weight Morphing

Simulations verify that different types of weights perform differently depending on the correlation of the inner-state vector. Fig. 4.4 illustrates the convergence behavior of static AC with $\mathbf{W}^{\mathrm{CVX}}$ and $\mathbf{W}^{\mathrm{MH}}$ (and with other weight designs described below) for the case of a WSN (random geometric graphs) with $I = 100$ nodes. The measurements here were obtained by independently sampling from a Gaussian distribution $\mathcal{N}\{0, 1\}$, i.e., the inner-state vector is uncorrelated initially. It is seen that the CVX weights provide an excellent asymptotic convergence rate, but the MH weights are clearly superior in the transient phase (i.e., for $k < 40$). Therefore, we want to exploit the strength of each weight design by choosing the corresponding weight design adaptively. We call this method "weight morphing" since we morph (nonlinearly) one weight design into the other; more

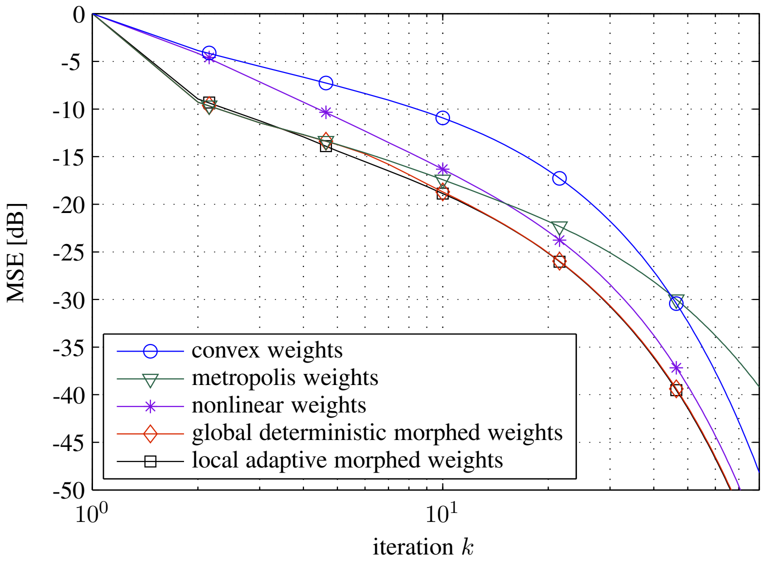


Figure 4.4: *Illustration of AC behavior for different weight designs in terms of mean-square error versus iteration index k.*

details are presented throughout this section.

Before starting with our approach, we want to refer to [21] where the authors proposed already a method that should improve the transient behavior of CVX weights by using a nonlinear method to compute time-variant weights. The AC update equation in [21] reads

$$\mathbf{x}[k+1] = \mathbf{W}^{\mathrm{NL}}[k]\,\mathbf{x}[k]\,, \tag{4.12}$$

where the weight matrix $\mathbf{W}^{\mathrm{NL}}[k]$ depends on the current states according to

$$w_{ij}^{\mathrm{NL}}[k] = \begin{cases} w_{ij}^{\mathrm{CVX}} f\big(x_i[k] - x_j[k]\big)\,, & \text{for } i \neq j, \\ 1 - \sum_{j\neq i} w_{ij}^{\mathrm{NL}}[k]\,, & \text{for } i = j. \end{cases}$$

The authors of [21] chose the nonlinear function as $f(u) = \frac{\tanh(\theta_1 u)\theta_2}{u}$ with $\theta_1\theta_2 \leq 1$.

The behavior of this NL weight design is also shown in Fig. 4.4. It is seen that in the transient performance with NL appropriately chosen θ_1 and θ_2 outperforms CVX . However, NL is still inferior to MH in the initial phase; furthermore, the parameters θ_1 and θ_2 have to be manually adapted to the specific scenario and there is no rule of thumb how to do that.

4.3.1 General Method

Motivated by the observation that MH performs best in the transient phase while CVX is superior in the asymptotic phase, we aim at combining the advantages of MH and CVX to obtain a weight design that performs uniformly best for all k. To this end, we propose to morph MH into CVX via local convex combinations in which the coefficients depend nonlinearly on the current states. This results in a time-varying nonlinear update as

$$\mathbf{x}[k+1] = \mathbf{W}^{\mathrm{M}}[k]\,\mathbf{x}[k]\,, \tag{4.13}$$

which is similar to (4.12). The morphed weights are given by

$$w_{ij}^{\mathrm{M}}[k] = \begin{cases} (1-\alpha_{ij}[k])w_{ij}^{\mathrm{MH}} + \alpha_{ij}[k]w_{ij}^{\mathrm{CVX}}\,, & \text{for } i \neq j\,, \\ 1 - \sum_{j\neq i} w_{ij}^{\mathrm{M}}[k]\,, & \text{for } i = j. \end{cases} \tag{4.14}$$

where $0 \leq \alpha_{ij}[k] \leq 1$ denotes the morphing coefficients. Note that the coefficients $\alpha_{ij}[k]$ are time-varying and potentially different for each edge. MH and CVX are obtained as special cases of (4.14) with $\alpha_{ij}[k] = 0$ and $\alpha_{ij}[k] = 1$, respectively. It is straightforward to check that $\mathbf{W}^{\mathrm{M}}\mathbf{1} = \mathbf{1}$, i.e., (4.14) preserves the true mean and satisfies the property that the mean is a fixed point. The condition $\rho\{\mathbf{W}^{\mathrm{M}}[k] - \frac{1}{I}\mathbf{1}\mathbf{1}^T\} < 1$ may be violated in the transition phase between MH and CVX. However, if our scheme should start to diverge, the weight adaptation described below would switch back to MH and thereby bring the algorithm back on track.

4.3.2 Global Deterministic Morphing

A very simple and intuitive choice for the coefficients $\alpha_{ij}[k]$ amounts basically to using MH initially (i.e., for $k < k_0$, with k_0 a precomputed number of iterations), and afterwards switch to CVX. This means $\alpha_{ij}[k] = 0$ for $k < k_0$ and $\alpha_{ij}[k] = 1$ for $k \geq k_0$. In practice, it is advantageous to use a gradual transition rather than an abrupt change, i.e., $\alpha_{ij}[k] = g[k]$, where $g[k]$ is a sigmoid-type function with $0 \leq g[0]$ and $\lim_{k\to\infty} g[k] = 1$. Here, the coefficients for all edges are identical and (4.14) simplifies to $\mathbf{W}^{\mathrm{M}}[k] = (1-g[k])\mathbf{W}^{\mathrm{MH}} + g[k]\mathbf{W}^{\mathrm{CVX}}$. Since the weights do not depend on the current state, the AC update is time-varying but remains linear. Due to the triangle equality $\rho\{(1-a)\mathbf{A} + a\mathbf{B}\} \leq |1-a|\rho\{\mathbf{A}\} + |a|\rho\{\mathbf{B}\}$, it follows directly that

$$\rho\left\{\mathbf{W}^{\mathrm{M}}[k] - \frac{1}{I}\mathbf{1}\mathbf{1}^T\right\} < 1.$$

The performance of the globally morphed weights with $g[k] = \frac{1}{1+\exp(-2(k-6))}$ is also shown in Fig. 4.4. Clearly, this scheme outperforms MH, CVX, and even nonlinear AC. While it is desirable that the transition between MH and CVX is optimized for the specific scenario at hand, the performance of the algorithm turns out to be rather insensitive to the precise shape of $g[k]$.

4.3.3 Local Adaptive Morphing

We next propose an alternative to the global morphing described above that chooses the coefficients $\alpha_{ij}[k]$ locally in an adaptive manner, i.e., depending on the current states. The basic idea is that if the states $x_i[k]$, $i = 1, \ldots, I$, differ "strongly," we are still in the transient phase and should rather use MH; in contrast, if all states are "close" to each other, we are in the asymptotic phase and should use CVX. To prevent additional communication between nodes, we propose to assess the current discrepancy of the states via the following two metrics that depend only on the states of two neighboring nodes:

$$m_{ij}^{(1)}[k] = \frac{(x_i[k] - x_j[k])^2}{2}\,, \tag{4.15}$$

$$m_{ij}^{(2)}[k] = \frac{(x_i[k] - x_j[k])^2}{x_i^2[k] + x_j^2[k]}\,. \tag{4.16}$$

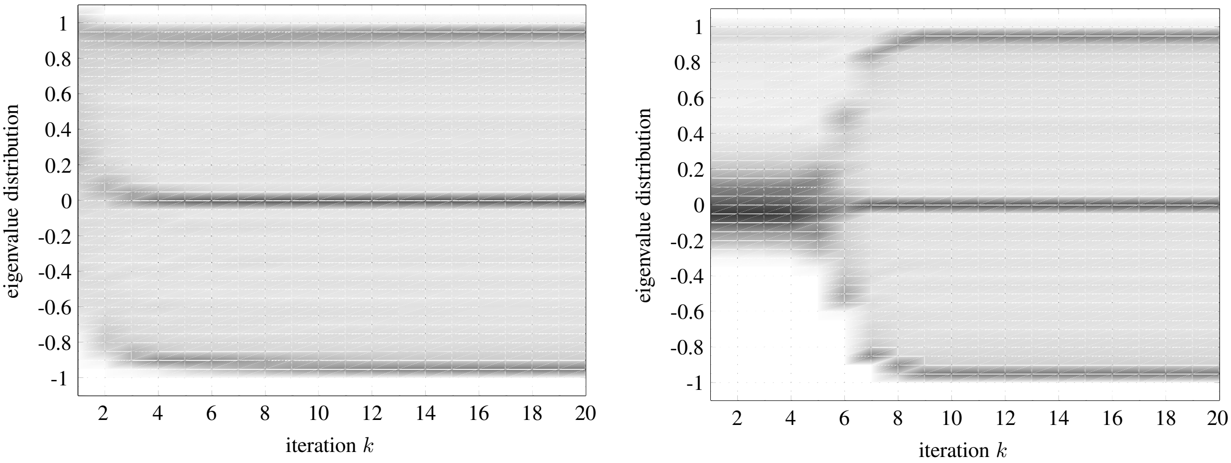


Figure 4.5: *Distribution of the eigenvalues of the weight matrix* $\mathbf{W}[k]$ *versus iteration index* k*: nonlinear AC [21] (left) and global deterministic morphing (right). In the right-hand side, the eigenvalue distribution for* $k=0$ *and* $k=20$ *corresponds to MH and CVX, respectively.*

The morphing coefficients are then chosen in a local and adaptive fashion as

$$\alpha_{ij}[k] = f(m_{ij}^{(l)}[k]), \quad l = 1, 2. \tag{4.17}$$

where $f(\cdot)$ is a sigmoid-like monotonically decreasing function satisfying $f(0)=1$. Since the morphing coefficients and thus the AC weights depend on the current states, the overall AC update here becomes nonlinear. The metric $m_{ij}^{(1)}[k]$ equals the sample variance of the pair $\{x_i[k], x_j[k]\}$ and has the advantage of being translation invariant, i.e., an additive offset of both states leaves $m_{ij}^{(1)}[k]$ unchanged. In contrast, $m_{ij}^{(2)}[k]$ is scale invariant but not translation invariant, i.e., a multiplicative scaling of both states does not change $m_{ij}^{(2)}[k]$.

An illustration of the performance of AC with locally morphed weights based on the metric $m_{ij}^{(1)}[k]$ is given in Fig. 4.4. It is seen that this scheme performs best among all weight design even though the transition between MH and CVX is completely adaptive.

4.3.4 Interpretation

Fig. 4.4 reveals that our proposed method using morphed AC weights combines the good transient performance of MH and the optimal asymptotic performance of CVX. This excellent performance is further supported by Fig. 4.5, which shows the average eigenvalue distribution of the weight matrix for the case of nonlinear AC [21] and our globally morphed weight matrix. It is seen that—in contrast to nonlinear AC—our method initially has an eigenvalue distribution that is strongly concentrated about zero. This implies that the corresponding components of the state vector (i.e., which are orthogonal to $\mathbf{1}$) are quickly attenuated. The transition to CVX weights then minimizes the spectral radius, which in turn attenuates the remaining components as fast as possible.

4.3.5 Dynamic Case

In this section we show how to apply our weight morphing scheme to dynamic AC introduced in [29]. Here, each sensor obtains a time-varying measurement $s_i[n]$, $i=1,\ldots,I$ and in the AC update equation the first order

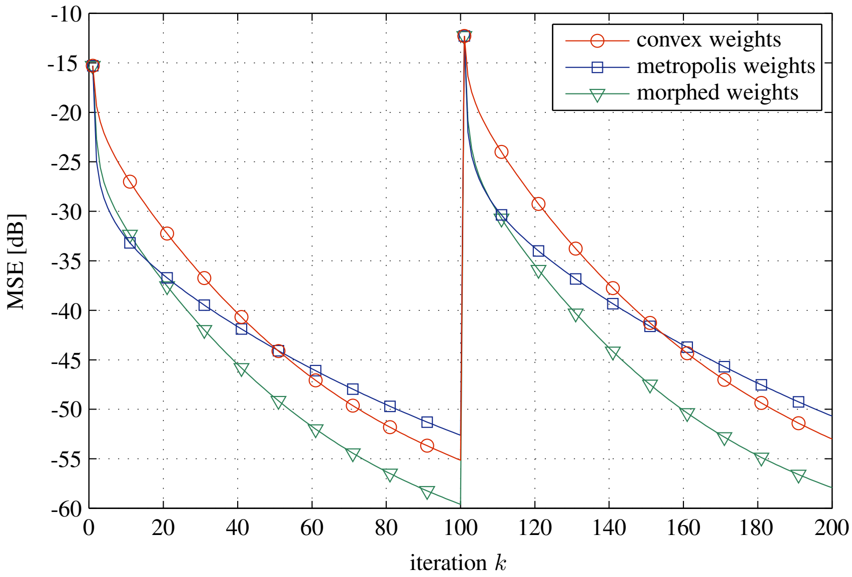


Figure 4.6: *MSE versus time for dynamic AC with different weight designs and an abrupt change in the measurements.*

difference $(\Delta\mathbf{s})[n] = \mathbf{s}[n] - \mathbf{s}[n-1]$ has to be incorporated, see Section 2.2.4 for more details.

We can apply our weight morphing method to dynamic AC by replacing $\mathbf{W}$ in (2.5) with the time-varying morphed weights in (4.14):

$$\mathbf{x}[n+1] = \mathbf{W}^{\mathrm{M}}[n]\,\mathbf{x}[n] + (\Delta\mathbf{s})[n]\,. \tag{4.18}$$

For the morphing coefficients, we use local adaptation according to (4.17), since the global deterministic morphing is not meaningful in a dynamic setting. Hence, the resulting AC algorithm tends to prefer MH weights for fast varying signal portions and low SNR while for slowly varying signal parts and high SNR the CVX weights will be used.

4.3.6 Numerical Results

In our simulations, we consider random geometric graphs with $I = 100$ sensors randomly deployed within the unit square. The connectivity (cf. Section 2.1.2) is chosen as $c = 2$. The performance of the various dynamic AC algorithms is assessed by the normalized MSE introduced in Section 2.2.5.

We first consider a pseudo-static scenario in which the measurements are constant for some time and then change abruptly to different values that again remain constant afterwards. Fig. 4.6 shows the MSE achieved in this scenario with MH, CVX, and local adaptive weight morphing (based on $m_{ij}^{(1)}[k]$) averaged over 3000 scenarios. In particular, we chose

$$\alpha_{ij}[k] = \frac{1}{1 + \exp\left(20\log\left(m_{ij}^{(1)}[k]\right)\right)}.$$

It is seen that our proposed scheme outperforms MH and CVX and is able to cope both with the transient and the quasi-stationary parts of the measurements. More specifically, while CVX weights are used after $k = 15$,

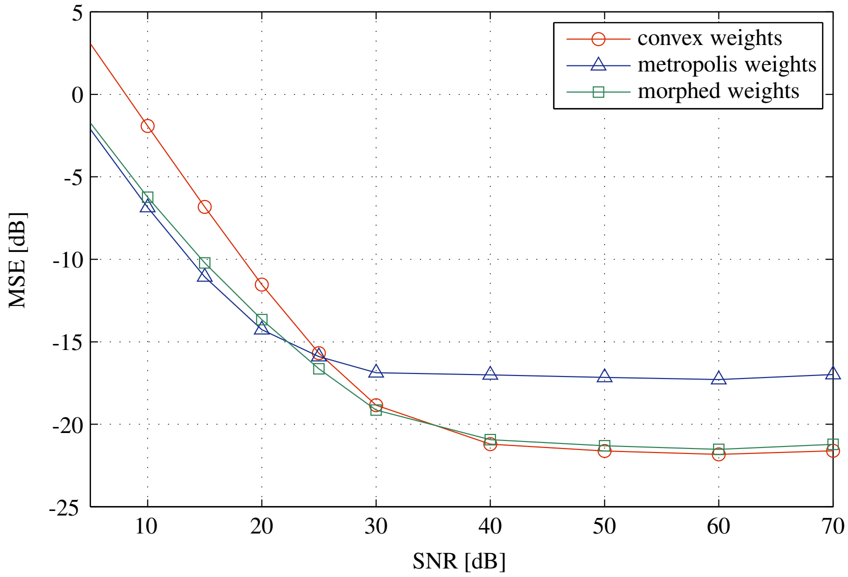


Figure 4.7: *Influence of the SNR on the MSE achieved with dynamic AC using different weight designs.*

the adaptation underlying the morphing coefficients recognizes the abrupt change at $k=100$ and switches back from CVX to MH.

We next examine a scenario in which the sensor measurements are noisy samples from a spatio-temporal field which is constructed via a spatial Fourier series in which the coefficients are low-pass processes with normalized bandwidth $\theta_{\mathrm{c}}^{(\mathrm{t})} = 3 \cdot 10^{-4}$. Fig. 4.7 displays the (time-averaged) MSE versus the SNR (averaged over 3000 scenarios) for MH, CVX, and local adaptive weight morphing (based on $m_{ij}^{(2)}[n]$) with

$$\alpha_{ij}[n] = \frac{1}{1 + e^{46}\left(m_{ij}^{(2)}[n]\right)^{86.8}}$$

Again our proposed scheme overall performs best by combining in a non-supervised manner the advantages of MH at low SNR and of CVX at high SNR. Hence, in the context of distributed averaging in unknown noise levels, our weight morphing scheme picks the appropriate weights in an automated manner.

A comparison with the optimum time-varying weights of the previous section is provided in Section 4.5.

4.4 Blending with Advection-Diffusion

In contrast to the two previous sections, which considered symmetric weights that vary over time, we next introduce asymmetric constant weights motivated by a physical flow in this section. Our contribution is based on [35] where the authors use an analogy with fluid dynamics to improve the convergence speed of AC. In particular, they discretize the advection-diffusion equation and apply a simple velocity field (i.e., flow). However, their design is limited to networks with nodes uniformly distributed on a torus (i.e., without boundaries). Unfortunately, the practical relevance of such a setup is rather limited.

Our approach uses a different discretization method for advection-diffusion processes that leads to AC designs with improved performance and allow us to take into account boundaries. Moreover for the case where sensors are arranged on a uniform grid, we propose new velocity fields and provide a stability analysis for the resulting AC algorithm (following [47]). Finally, we show how to apply the idea of advection on arbitrary network topologies and we present an advection optimization approach for random geometric graphs that causes a blending of the inner states.

4.4.1 Basic Theory

We start with a brief explanation of the advection-diffusion process [68], which is a combination of a diffusion process and an advection process, i.e., a bulk flow induced by an external velocity field. We consider a quantity $x(\mathbf{r}, t)$, with $\mathbf{r}$ and t denoting space and time, respectively. The general advection-diffusion equation reads

$$\frac{\partial x}{\partial t} = \nabla \cdot (D \nabla x) - \nabla \cdot (\mathbf{v} x) + S\,, \tag{4.19}$$

where we omitted arguments for simplicity (the Nabla operator ∇ corresponds the spatial partial derivatives, thus we can also write $\nabla_{\mathbf{r}}$). The diffusion coefficient is denoted as $D(\mathbf{r}, t)$, $\mathbf{v}(\mathbf{r}, t)$ is the velocity vector, and $S(\mathbf{r}, t)$ describes the sources and sinks. All of these parameters can vary over time and space. The first term on the right-hand side of (4.19) is the diffusion part and the second term characterizes the advection.

In the following we consider a reduced parameter set that simplifies the advection-diffusion equation. First, we assume that the diffusion is constant over time and space, i.e., $D(\mathbf{r}, t) = D$ which yields $\nabla_{\mathbf{r}} \cdot (D \nabla_{\mathbf{r}}\, x(\mathbf{r}, t)) = D \nabla_{\mathbf{r}}^2\, x(\mathbf{r}, t)$. Second, we set sources and sinks to zero ($S(\mathbf{r}, t) = 0$). Finally it is assumed that the flow created through the velocity field cannot be compressed, which means that there are no points or areas where the velocity field will start or end.

These simplifications lead to the following advection-diffusion equation:

$$\frac{\partial x}{\partial t} = D \nabla^2 x - \mathbf{v} \cdot \nabla x\,. \tag{4.20}$$

So far we have ignored boundary conditions, which amounts to a toroidal geometry (cf. [35]). A more realistic assumption is to define the region of interest as the unit square $[0, 1] \times [0, 1]$. This requires the specification of boundary conditions for the advection-diffusion process. For the diffusion we impose a reflection at the boundary ($\mathbf{x}(\mathbf{r}, t)$ is preserved in the unit square), which implies that the first derivative of the normal component has to be zero. For the velocity field, the normal component is assumed to be zero at the entire boundary (nothing is transported outside the unit square).

4.4.2 Discretization of the Advection-Diffusion Equation

We next describe the discretization of the advection-diffusion equation (4.20) on a square lattice, using a different approach than [35]. Our discretization method is based on the control-volume formulation, which is well tailored for advection-diffusion processes. The discretized model can be also obtained by applying a modified difference method [46], i.e., a modified second-order centered spatial differencing and first-order forward temporal differencing (Euler) scheme, which is also called FTCS (forward time, centered space). In [35], in contrast, they are using FTBS (forward time, backward space). The rest of this section is dedicated to the discretization using the control-volume formulation where we follow [68].

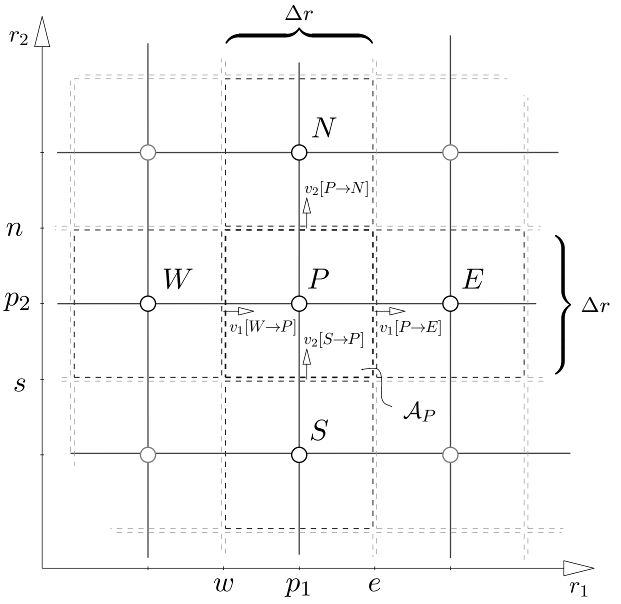


Figure 4.8: *Illustration of a control-volume in 2D space.*

The control-volume formulation divides the observed area $\mathcal{A}$ into disjoint areas. Through integrating over these small areas we obtain an approximation of the partial differential equation. For simplicity, these small areas are assumed to have the same size and shape. Hence, it is only necessary to consider one of these in more detail, which we denote by $\mathcal{A}_P$. This area is a square around the point P with side length Δr. The surrounding areas are denoted with the corresponding geographic direction (e.g. E stands fore east). The geometry is illustrated in Fig. 4.8. The position of the point P is given by the vector $(p_1, p_2)^T$ and the distance to the neighboring points S, W, N, and E is Δr. For instance, point E lies at the coordinates $(p_1 + \Delta r, p_2)^T$. The boundary of the area $\mathcal{A}_P$ is given through w, e, n, and s, which can be expressed via p_1, p_2 and Δr as $w = p_1 - \frac{\Delta r}{2}$, $e = p_1 + \frac{\Delta r}{2}$, $s = p_2 - \frac{\Delta r}{2}$, and $n = p_2 + \frac{\Delta r}{2}$. Therefore, we have $\mathcal{A}_P = [w, e] \times [s, n]$ and $|\mathcal{A}_P| = (\Delta r)^2$. Similar to the spatial partitioning, the temporal coordinate is divided into successive intervals of duration Δt.

In the next step we integrate equation (4.20) over the control-volume $[w, e] \times [s, n] \times [t, t + \Delta t]$:

$$\int_t^{t+\Delta t} \int_w^e \int_s^n \frac{\partial x}{\partial t} \mathrm{d}r_2 \mathrm{d}r_1 \mathrm{d}t = \int_t^{t+\Delta t} \int_w^e \int_s^n D\nabla^2 x - \mathbf{v} \cdot \nabla x \, \mathrm{d}r_2 \mathrm{d}r_1 \mathrm{d}t \,. \tag{4.21}$$

In the following, we are going to consider the left hand side and the right hand side of (4.21) separately. Starting with the left hand side, we obtain

$$\begin{aligned} \int_t^{t+\Delta t} \int_w^e \int_s^n \frac{\partial x}{\partial t} \mathrm{d}r_2 \mathrm{d}r_1 \mathrm{d}t &= \int_w^e \int_s^n \int_t^{t+\Delta t} \frac{\partial x}{\partial t} \mathrm{d}t \, \mathrm{d}r_2 \mathrm{d}r_1 && (4.22) \\ &= \int_w^e \int_s^n \left(x(t + \Delta t, r_1, r_2) - x(t, r_1, r_2) \right) \mathrm{d}r_2 \mathrm{d}r_1 \\ &\approx (\Delta r)^2 \left(x[k + 1, P] - x[k, P] \right) \,, && (4.23) \end{aligned}$$

where (4.22) follows from Fubini's theorem. The approximation in (4.23) is exact, if the quantity x is constant in $\mathcal{A}_P$. Additionally we have $x[k, P] = x(k\Delta t, p_1, p_2)$ and $k\Delta t = t$.

We next consider the right side of (4.23). We separate the integral due to its linearity,

$$\begin{aligned}&\int_t^{t+\Delta t}\int_w^e\int_s^n \big(D\nabla^2 x-\mathbf{v}\cdot\nabla x\big)\,\mathrm{d}r_2\mathrm{d}r_1\mathrm{d}t\\ &\quad=\int_t^{t+\Delta t}\int_w^e\int_s^n D\nabla^2 x\,\mathrm{d}r_2\mathrm{d}r_1\mathrm{d}t-\int_t^{t+\Delta t}\int_w^e\int_s^n \mathbf{v}\cdot\nabla x\,\mathrm{d}r_2\mathrm{d}r_1\mathrm{d}t\,, \qquad (4.24)\end{aligned}$$

and evaluate the summands individually. For simplicity we omit the time dependency of x in the following. First, the Laplace operator for the 2D Euclidean space is given by

$$\nabla^2 x(r_1,r_2)=\frac{\partial^2 x(r_1,r_2)}{\partial r_1^2}+\frac{\partial^2 x(r_1,r_2)}{\partial r_2^2}\,,$$

where the mixed terms are zero due to the orthogonality. We apply the approximation

$$\left(\frac{\partial^2 x(r_1,r_2)}{\partial r_1^2}+\frac{\partial^2 x(r_1,r_2)}{\partial r_2^2}\right)_{w<r_1<e,\ s<r_2<n}\approx\frac{\mathrm{d}^2 x(r_1,p_2)}{\mathrm{d}r_1^2}+\frac{\mathrm{d}^2 x(p_1,r_2)}{\mathrm{d}r_2^2}\,,$$

which is exact, if the second derivative in r_1 is constant over r_2 inside $\mathcal{A}_P$ and conversely. We thus obtain

$$\begin{aligned}&\int_w^e\int_s^n D\nabla^2 x(r_1,r_2)\,\mathrm{d}r_2\mathrm{d}r_1\\ &\quad=D\left(\int_w^e\frac{\mathrm{d}^2 x(r_1,p_2)}{\mathrm{d}r_1^2}\mathrm{d}r_1\int_s^n\mathrm{d}r_2+\int_s^n\frac{\mathrm{d}^2 x(p_1,r_2)}{\mathrm{d}r_2^2}\mathrm{d}r_2\int_w^e\mathrm{d}r_1\right) \qquad (4.25)\\ &\quad=\Delta r D\left(\left(\frac{\mathrm{d}x(r_1,p_2)}{\mathrm{d}r_1}\right)_{r_2=e}-\left(\frac{\mathrm{d}x(r_1,p_2)}{\mathrm{d}r_1}\right)_{r_2=w}+\left(\frac{\mathrm{d}x(p_1,r_2)}{\mathrm{d}r_2}\right)_{r_1=n}-\left(\frac{\mathrm{d}x(p_1,r_2)}{\mathrm{d}r_2}\right)_{r_1=s}\right). \qquad (4.26)\end{aligned}$$

The four derivatives in (4.26) are approximated as

$$\begin{aligned}\left(\frac{\mathrm{d}x(r_1,p_2)}{\mathrm{d}r_1}\right)_{r_1=e}&\approx\frac{x[E]-x[P]}{\Delta r}\,, &\qquad \left(\frac{\mathrm{d}x(r_1,p_2)}{\mathrm{d}r_1}\right)_{r_1=w}&\approx\frac{x[P]-x[W]}{\Delta r}\,,\\ \left(\frac{\mathrm{d}x(p_1,r_2)}{\mathrm{d}r_2}\right)_{r_2=n}&\approx\frac{x[N]-x[P]}{\Delta r}\,, &\qquad \left(\frac{\mathrm{d}x(p_1,r_2)}{\mathrm{d}r_2}\right)_{r_2=s}&\approx\frac{x[P]-x[S]}{\Delta r}\,,\end{aligned}$$

where equality would follow if x is a linear function. To sum up, we obtain for first term in (4.24)

$$\int_w^e\int_s^n D\nabla^2 x(r_1,r_2)\,\mathrm{d}r_2\mathrm{d}r_1\approx D\left(x[W]+x[E]+x[N]+x[S]-4x[P]\right)\,.$$

The integration over time is considered later. We continue with evaluating the second term in (4.24), where we again omit the time index. We start with the following approximation:

$$\begin{aligned}&\int_w^e\int_s^n \mathbf{v}\cdot\nabla x\mathrm{d}r_1\mathrm{d}r_2\\ &\quad\approx\int_w^e v_1(r_1,p_2)\frac{\mathrm{d}x(r_1,p_2)}{\mathrm{d}r_1}\mathrm{d}r_1\int_s^n\mathrm{d}r_2+\int_s^n v_2(p_1,r_2)\frac{\mathrm{d}x(p_1,r_2)}{\mathrm{d}r_2}\mathrm{d}r_2\int_w^e\mathrm{d}r_1 \qquad (4.27)\\ &\quad=\Delta r\Big(\big(v_1(r_1,p_2)x(r_1,p_2)\big)_{r_1=e}-\big(v_1(r_1,p_2)x(r_1,p_2)\big)_{r_1=w}\\ &\qquad\qquad+\big(v_2(p_1,r_2)x(p_1,r_2)\big)_{r_2=n}-\big(v_2(p_1,r_2)x(p_1,r_2)\big)_{r_2=s}\Big)\,. \qquad (4.28)\end{aligned}$$

The approximation is accurate if v_1 is constant regarding r_2 and v_2 is constant regarding r_1 in $\mathcal{A}_P$, and additionally if the derivative of x with respect to r_1 is constant within $[w,e]$ and if the same holds for the derivative with respect to to r_2 equivalently. In the next step we interpolate the terms of (4.28) by averaging the field on the boundary, i.e.,

$$\begin{aligned}
\big(x(r_1,p_2)\big)_{r_1=e} &\approx \tfrac{1}{2}(x[E]+x[P])\,, & \big(x(r_1,p_2)\big)_{r_1=w} &\approx \tfrac{1}{2}(x[P]+x[W])\,,\\
\big(x(p_1,r_2)\big)_{r_2=n} &\approx \tfrac{1}{2}(x[N]+x[P])\,, & \big(x(p_1,r_2)\big)_{r_2=s} &\approx \tfrac{1}{2}(x[P]+x[S])\,,
\end{aligned}$$

which are exact if x constant inside $\mathcal{A}_P$. Hence, (4.28) yields

$$\begin{aligned}
&\frac{\Delta r}{2}\big(v_1(e,p_2)(x[E]+x[P]) - v_1(w,p_2)(x[P]+x[W])\\
&\qquad\qquad + v_2(p_1,n)(x[N]+x[P]) - v_2(p_1,s)(x[P]+x[S])\big)\\
=&\frac{\Delta r}{2}\big(v_1[P\to E](x[E]+x[P]) - v_1[W\to P](x[P]+x[W])\\
&\qquad\qquad + v_2[P\to N](x[N]+x[P]) - v_2[S\to P](x[P]+x[S])\big)\,, \qquad (4.29)
\end{aligned}$$

where we changed to a more intuitive notation for the velocities (e.g., $v_1(e,p_2) = v_1[P\to E]$, cf. Fig. 4.8). In the next step we include the incompressible flow condition, i.e., $\nabla\cdot\mathbf{v}=0$, that yields

$$\begin{aligned}
&\int_w^e\int_s^n \frac{\mathrm{d}v_1(r_1,r_2)}{\mathrm{d}r_1} + \frac{\mathrm{d}v_2(r_1,r_2)}{\mathrm{d}r_2} dr_1\mathrm{d}r_2\\
&\quad\approx \int_w^e \frac{\mathrm{d}v_1(r_1,p_2)}{\mathrm{d}r_1}\mathrm{d}r_1 \int_s^n dr_2 + \int_s^n \frac{\mathrm{d}v_2(p_1,r_2)}{\mathrm{d}r_2}\mathrm{d}r_2\int_w^e \mathrm{d}r_1\\
&\quad= \Delta r(v_1[P\to E] - v_1[W\to P] + v_2[P\to N] - v_2[S\to P]) \overset{!}{=} 0\,, \qquad (4.30)
\end{aligned}$$

where the approximation is accurate, if the first derivative of v_1 with respect to r_1 is invariant of a change of r_2 within $[s,n]$ and if the same applies for the derivative of v_2. Finally, we obtain

$$\int_w^e\int_s^n \mathbf{v}\cdot\nabla x\mathrm{d}r_1\mathrm{d}r_2 \approx \frac{\Delta r}{2}\big(v_1[P\to E]x[E] - v_1[W\to P]x[W] + v_2[P\to N]x[N] - v_2[S\to P]x[S]\big)\,.$$

In the next step we consider the time-dependence of the previously derived expressions. To this end we are using an explicit time discretization, i.e., the value of the field inside the control volume changes immediately before the next sampling point. Therefore, we assume that the field inside the period $[k\Delta t,(k+1)\Delta t]$ is constant and represented by its sampling value at index k. Different concepts for time discretization are the implicit method and Crank-Nicolson scheme [68, Section 4.3-2]. Hence, we obtain a discrete approximation of the advection-diffusion equation, i.e.,

$$\begin{aligned}
\frac{x[k+1,P]-x[k,P]}{\Delta t} =& \frac{D}{(\Delta r)^2}\left(x[k,E]+x[k,W]+x[k,N]+x[k,S]-4x[k,P]\right)\\
&-\frac{1}{2\Delta r}\left(v_1[P\to E]x[k,E] - v_1[W\to P]x[k,W] + v_2[P\to N]x[k,N] - v_2[S\to P]x[k,S]\right)\,,
\end{aligned}$$

where the design of the advection field has to fulfill (4.30).

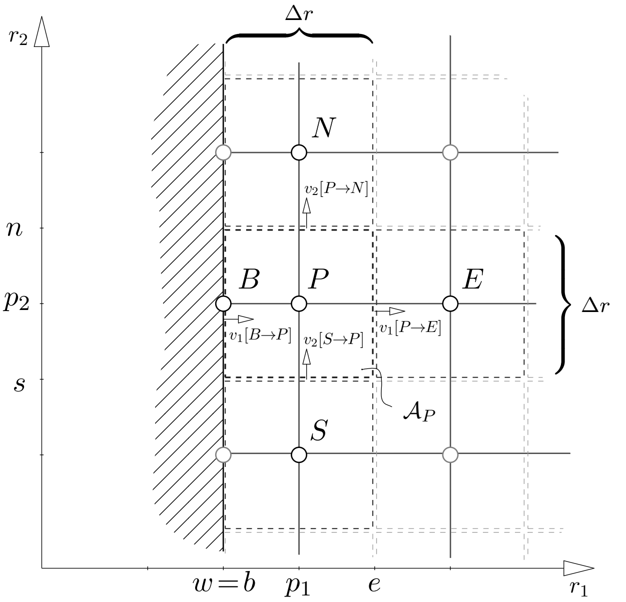


Figure 4.9: *Illustration of a control-volume lying at the boundary in 2D space.*

In the following, we impose boundary conditions for a control volume. We assume that instead of point W, point P is adjacent to the boundary point B as illustrated in Fig. 4.9. To ensure the boundary conditions being fulfilled, we have to substitute in (4.26) the following expression:

$$\left(\frac{\mathrm{d}x(r_1,p_2)}{\mathrm{d}r_1}\right)_{r_1=b}=0\,.$$

This equation leads to the property (due to the approximations above) that the value at point B is equal to point P. Therefore, the value at the boundary point $x[k,B]$ will not show up in the diffusion part. For the advection the velocity normal to the boundary is zero, because no energy is allowed to leave the system. Hence, for our example in Fig. 4.9 we have $v_1[B\to P]=0$. Putting all things together we obtain for point P a discretized version of the advection-diffusion equation as

$$\begin{aligned}\frac{x[k+1,P]-x[k,P]}{\Delta t}=&\ \frac{D}{(\Delta r)^2}\,(x[k,E]+x[k,N]+x[k,S]-3x[k,P])\\ &-\frac{1}{2\Delta r}\,(v_1[P\!\to\! E]x[k,E]+v_2[P\!\to\! N]x[k,N]-v_2[S\!\to\! P]x[k,S])\ .\end{aligned}$$

If the boundary is in a different direction, the conditions apply analogously.

Finally, we put all the local advection-diffusion equations together, such that we have the global system represented. We assume that the points are distributed on a regular square grid, where we have $L=\sqrt{I}$ points in each direction. Then we stack the field values into a vector $\mathbf{x}$ according to the rule: the first element is the field point with $r_1=0$ and $r_2=0$ (the first point in the first row) followed by its neighbor on the east side. This is done until the end of the row (after L points), then we switch to the first element in the second row (which is the element on the north of the first element of the vector $\mathbf{x}$).

Therefore, for any point P that does not lie at the boundary (no matter if periodic or non-periodic structure) we have

$$\mathbf{x} = (\ldots \underbrace{x[S] \ldots x[W]}_{L \text{ elements}} \; x[P] \; \underbrace{x[E] \ldots x[N]}_{L \text{ elements}} \ldots)^T$$

where we omitted the time index. We can then write all local relations as the following linear system:

$$\mathbf{x}[k+1] = \mathbf{x}[k] - \alpha\mathbf{D}\mathbf{x}[k] - \zeta\mathbf{C}\mathbf{x}[k] = (\mathbf{I} - \alpha\mathbf{D} - \zeta\mathbf{C})\mathbf{x}[k]\,, \tag{4.31}$$

which approximates the advection-diffusion equation in (2.7). In (4.31), the matrix $\mathbf{D}$ represents the diffusion and the matrix $\mathbf{C}$ (which contains the sampled velocity field) summarizes the advection. Moreover we have $\alpha = \frac{D\Delta t}{\Delta r^2}$ and $\zeta = \frac{v_{\text{max}}\Delta t}{2\Delta r}$, with v_{max} the maximum velocity at the sampling positions and hence we have $\max|[\mathbf{C}]_{ij}| = 1$. The ratio ζ/α is called the Peclet number [68]. According to the distribution of the points in the vector $\mathbf{x}$ the matrix $\mathbf{D}$ has to have the following structure

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_{\text{row}} & \mathbf{I} & 0 & \cdots & 0 & \boxed{\mathbf{I}} \\ \mathbf{I} & \mathbf{D}_{\text{row}} & \mathbf{I} & \ddots & & 0 \\ 0 & \mathbf{I} & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & & 0 \\ 0 & & & & \mathbf{D}_{\text{row}} & \mathbf{I} \\ \boxed{\mathbf{I}} & 0 & \cdots & 0 & \mathbf{I} & \mathbf{D}_{\text{row}} \end{pmatrix},$$

where the matrix $\mathbf{D}_{\text{row}}$, which represents the relation of the points of each row in the grid, is defined as

$$\mathbf{D}_{\text{row}} = \begin{pmatrix} -4 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -4 & 1 & \ddots & & 0 \\ 0 & 1 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & 1 & 0 \\ 0 & & & 1 & -4 & 1 \\ 1 & 0 & \cdots & 0 & 1 & -4 \end{pmatrix}.$$

This definition of the $\mathbf{D}$ matrix is only valid if the points are connected periodically, i.e., the area $\mathcal{A}$ is the surface of a torus. If we have boundaries, all the marked identity matrices in $\mathbf{D}$ have to be substituted with zero matrices and the $\mathbf{D}_{\text{row}}$ matrix has to be modified to

$$\mathbf{D}_{\text{row}} = \begin{pmatrix} -3 & 1 & 0 & \cdots & 0 & 0 \\ 1 & -4 & 1 & \ddots & & 0 \\ 0 & 1 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & 1 & 0 \\ 0 & & & 1 & -4 & 1 \\ 0 & 0 & \cdots & 0 & 1 & -3 \end{pmatrix},$$

and the first and the last $\mathbf{D}_{\text{row}}$ matrix in $\mathbf{D}$ has to be computed as $\mathbf{D}_{\text{row}} + \mathbf{I}$. It is seen that $\mathbf{D} = \mathbf{A} - \text{diag}\{\mathbf{A1}\}$, where $\mathbf{A}$ is the adjacency matrix of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponding to the square grid. the sampling points. The structure of $\mathbf{D}$ implies that the row and column sums are always zero.

The advection matrix $\mathbf{C}$, however, has zero entries on the main diagonal, which is also a difference to [35]. Its zero pattern is equal to $\mathbf{A}$ and the non-zero elements are defined as,

$$[\mathbf{C}]_{ij} = \begin{cases} \frac{v_1[j \to i]}{v_{\max}}, & |i-j| < L \wedge (i,j) \in \mathcal{E}\,, \\ \frac{v_2[j \to i]}{v_{\max}}, & |i-j| \geq L \wedge (i,j) \in \mathcal{E}\,, \\ 0 & \text{else}\,. \end{cases}$$

Since the velocity which points from one node to another is the negative of the velocity pointing into the other direction (e.g. $v_1[j \to i] = -v_1[i \to j]$), $\mathbf{C}$ is skew-symmetric, i.e., $\mathbf{C} = -\mathbf{C}^T$; this property does not hold for the discretization in [35].

If the field is constant and there is only diffusion, there is no change in the field over time. In the discretization this amounts to $\mathbf{x}[k+1] = \mathbf{x}[k]$ and requires $\mathbf{D1} = \mathbf{0}$, which is fulfilled by the discretization. If there is advection, the same condition has to hold for $\mathbf{C}$, but this follows also from the incompressible flow condition $\nabla \cdot \mathbf{v} = 0$ which leads to (4.30). This means that the outgoing velocities at each node sum up to zero since $v_1[W \to P]$ and $v_2[S \to P]$ denote incoming velocities and therefore the sign has to be changed. Hence, it follows that the row-sums and column-sums of $\mathbf{C}$ are zero, i.e., $\mathbf{C1} = \mathbf{0}$ and $\mathbf{1}^T\mathbf{C} = \mathbf{0}^T$.

4.4.3 Relation Between AC and Advection-Diffusion

We can interpret the discrete advection-diffusion process (4.31) as an AC algorithm by assuming that the sampling positions are sensor's positions and the field samples are the AC states. Specifically, comparing (2.3) and (4.31) formally yields $\widetilde{\mathbf{W}} = \mathbf{I} - \alpha\mathbf{D} - \zeta\mathbf{C}$. The update equation (4.31) requires only local computation, i.e., each sampling point which is equal to one sensor node uses the values of the direct neighboring sampling points (four in total, because there are two in each direction). Hence, the updates are processed on a graph where the points are located on a regular grid. A comparison of the graph Laplacian and the discretized Laplacian operator (∇^2) of the advection-diffusion equation reveals that they are equal, i.e., $\mathbf{L} = \mathbf{D}$. Additionally, if $\alpha = \epsilon$ and $\zeta = 0$ the advection-diffusion process is equivalent to AC with constant weights. In general $\alpha\mathbf{D}$ is a weighted Laplacian $\mathbf{L}_{\mathbf{w}}$ of the underlying graph.

For the case $\zeta \neq 0$, there is an additional component $\zeta\mathbf{Cx}[k]$ in the update equation. Because of $\mathbf{C1} = \mathbf{0}$ and $\mathbf{1}^T\mathbf{C} = \mathbf{0}$ this term does not influence the average of the state vector and the fact that any constant vector is a fixed point of AC. The term $\zeta\mathbf{C}$ can act like a blender (cf. Section 4.4.4) on the data and influences the convergence. However, if ζ is too large, the iterations become unstable. We provide a convergence analysis in Section 4.4.5.

4.4.4 Design of the Velocity Field

In [35] the velocity field was designed according to the condition $\mathbf{v}(r_1, r_2, t) = (v_1(r_2, t)\ v_2(r_1, t))^T$, i.e., the velocity field in direction r_1 is invariant with respect to a change in r_1 and the same applies in r_2. This

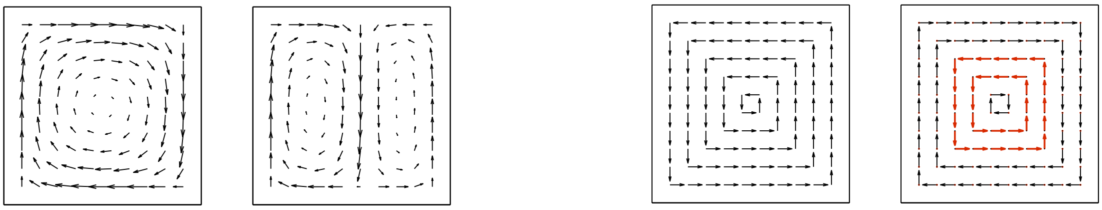


Figure 4.10: Examples of different rotational velocity fields.

assumption simplifies the derivation of theoretical results but restricts the setting to toroidal topologies. For practical WSN with boundaries, this is not feasible and hence we need different velocity fields. A natural choice are rotational vector fields. Inspired by physics it is possible to generate vector fields that fulfill $\nabla \times \mathbf{v} = c$ (where $c \in \mathbb{R}$ is arbitrary) under the constraint that $\nabla \cdot \mathbf{v} = 0$. Some examples are shown in Fig. 4.10, where it is seen that the absolute value of the velocity decreases when approaching the center. Since the design of such rotational fields is involved, we use a modification which we call regular rotational vector field. Here, each velocity vector has the same length and each vector points in one of the two main directions. Realizations of regular rotational fields can be found in Fig. 4.10. It is seen that the vectors in the center have the same length as those at the boundary and therefore entail stronger advection. It is also possible to revert the direction of some of the cycles, which intuitively provides an improved blending. Performance results are provided in Section 4.4.8.

4.4.5 Convergence Analysis

In the following we provide sufficient conditions for the convergence of AC with advection for regular rotation fields. To that end, we follow [47], where the so-called von Neumann analysis [82] is applied to test the stability of our discretization scheme. In Appendix 4.A we go into more details of this type of analysis for a more general case and hence we keep the discussion short in this section. In Section 4.4.7 we provide a convergence analysis which is suitable for any graph topology and yields the same result as in Appendix 4.A, but the proof is much simpler.

With the von Neumann analysis it can be checked whether any averaging error $\mathbf{x}[k] - \bar{s}\mathbf{1}$ decays to zero. Since the error can be decomposed into Fourier modes with non-zero spatial frequency, it suffices to consider those modes separately. The Fourier modes are given by

$$f[p_1, p_2, k] = \xi^k \exp\{\imath(p_1\theta_1 + p_2\theta_2)\},$$

where $\theta_m = k_m \Delta x$, $m = 1, 2$, with k_m denoting the wave number (hence, $-\pi \le \theta_m \le \pi$). Inserting these modes into the local discretized update equation (cf. [47], which is similar to Section 4.4.2), we obtain

$$\xi = 1 - 2\imath(\zeta_1 \sin(\theta_1) + \zeta_2 \sin(\theta_2)) + 2\alpha(\cos(\theta_1) + \cos(\theta_2) - 2).$$

The values ζ_1 and ζ_2 represent the velocities in the two spatial directions. To ensure convergence we need to guarantee that $|\xi|^2 < 1$ for all θ_1 and θ_2, which means that all spatial frequency modes decay as the iterations progress. The results of [47] require that $\alpha < 1/4$ and $\zeta_1^2 + \zeta_2^2 < \alpha/2$, which are sufficient and necessary conditions for von Neumann stability for the case that α and ζ_m are spatially constant and that the sampling points lie on a torus.

In our case, we assume regular rotational velocity fields (but still on a torus). For this setting, we arrive at the same result under the assumption that one of the two ζ is zero since only two of the possible four velocities around each node are zero and these two velocities have to be equal and are represented by the nonzero ζ. Hence, we obtain $\alpha < 1/4$ and $\zeta^2 \leq \alpha/2$ as sufficient conditions for convergence (the first one is also necessary). We will see in Section 4.4.8 that the real stability bound described by the second condition is a bit off from these sufficient conditions, which can be explained by the fact that the cycles on which spatial waves are propagated and amplified through advection have limited size and therefore in particular small spatial frequencies are attenuated by the topology.

4.4.6 Blending in WSN with Arbitrary Topology

So far we considered rotational fields on regular grid graphs, whose practical importance is limited. Hence, we next attempt to generalize the idea of blending the data via rotational advection fields to arbitrary WSNs in order to improve the AC performance. To this end we start with a weighted Laplacian matrix $\mathbf{L}_\mathbf{w}$ (which corresponds to the diffusion part) where the weights can be designed through any method that leads to AC convergence. Then we add an optimized advection matrix $\mathbf{C}$ to improve performance. We omit in the following the factor ζ of the previous sections by assuming that the elements of $\mathbf{C}$ are directly the velocities and we have no normalization, i.e., $[\mathbf{C}]_{ij} = v_{ij}$. In the following, we show how to analytically construct the matrix $\mathbf{C}$ for any given graph and velocity vector and then we present an approach to design appropriate velocity fields.

The velocity vector $\mathbf{v}$ has $|\mathcal{E}|$ elements, where each element defines the velocity for one edge (the sign defines the direction). To construct the advection matrix $\mathbf{C}$ we need the incidence matrix of the underlying graph $\mathcal{G}$ as defined in Section 3.2.1. The incidence matrix $\tilde{\mathbf{B}}$ comes with a permutation matrix $\mathbf{P}$ such that the adjacency matrix of the graph can be written as $\mathbf{A} = \tilde{\mathbf{B}}\mathbf{P}\tilde{\mathbf{B}}^T$ and the graph Laplacian reads $\mathbf{L} = \tilde{\mathbf{B}}\tilde{\mathbf{B}}^T - \tilde{\mathbf{B}}\mathbf{P}\tilde{\mathbf{B}}^T$. This representation is not unique and therefore we restrict to the case where the permutation matrix is chosen as $\mathbf{P} = \mathbf{I} \otimes (\mathbf{E} - \mathbf{I})$, where $\mathbf{E}$ denotes the all one matrix. The structure of $\mathbf{P}$ allows us to define $\mathbf{F} = \mathbf{I} \otimes [1\ -1]^T$ with which we can construct proper advection matrices $\mathbf{C}$ according to

$$\mathbf{C} = \tilde{\mathbf{B}}\mathbf{P}\,\mathrm{diag}\{\mathbf{F}\mathbf{v}\}\tilde{\mathbf{B}}^T\,. \tag{4.32}$$

In general, the incompressible flow condition has to be ensured via the design of the velocity vector $\mathbf{v}$. The construction in (4.32) shows that $\mathbf{C}$ depends linearly on $\mathbf{v}$.

The velocity vector is obtained by solving the optimization problem

$$\begin{aligned} &\text{minimize } f(\mathbf{v}) \\ &\text{subject to } \mathbf{C}\mathbf{1} = \mathbf{0}, \\ &\qquad \left\| \mathbf{I} - \mathbf{L}_\mathbf{w} - \mathbf{C} - \frac{1}{I}\mathbf{1}\mathbf{1}^T \right\|_2 < 1, \end{aligned} \tag{4.33}$$

where $f(\mathbf{v})$ is a suitable convex function (see below), chosen to increase the advection of the resulting algorithm. The first constraint ensures an incompressible flow and the second constraint enforces convergence.

The matrix $\mathbf{L_w}$ has to be prescribed (e.g., using the standard designs mentioned in Section 2.2.2) and has to fulfill all AC convergence conditions. The stability constraint (4.33) can be replaced with the more stringent condition $0 \leq w_{ij} \pm v_{ij} \leq 1$ for all $(i,j) \in \mathcal{E}$, which we derive in the next section. In this context w_{ij} denotes the symmetric weights of AC and the final weights are $w_{ij} \pm v_{ij}$, accordingly. The resulting optimization problem contains only local conditions. We therefore expect that for specific target functions this problem can be solved in a distributed manner.

To obtain a suitable objective function $f(\mathbf{v})$ we again use the idea of rotation fields. Essentially, we impose an initial target velocity vector that mimics a rotational field and use as convex objective function the Euclidean distance to this target velocity field, i.e., $f(\mathbf{v}) = \|\mathbf{v} - \mathbf{v}_{\text{init}}\|_2$. We generate the target velocity vector $\mathbf{v}_{\text{init}}$ using the direction of each edge with respect to a circular reference field. If the direction of an edge l linking node i and j (almost) points into the tangential direction of a centered circle (i.e., the center of the circle lies in the center of the area where the nodes are distributed) then we assign a large velocity. An edge that is (almost) collinear to the radial direction gets a small velocity. This can be achieved by considering the metric $\mathbf{r}_i^T \mathbf{T} \mathbf{r}_j$ with

$$\mathbf{T} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

and the positions $\mathbf{r}_i$ and $\mathbf{r}_j$ of the two nodes are relative to the center of the circle. A proper normalization (i.e., the expected distance between two nodes $\mathrm{E}\{\|\mathbf{r}_i - \mathbf{r}_j\|\}$ times the expected average distance of two nodes to the center $\mathrm{E}\{\|\mathbf{r}_i + \mathbf{r}_j\|\}/2$) finally yields

$$[\mathbf{v}_{\text{init}}]_l = \frac{2\mathbf{r}_i^T \mathbf{T} \mathbf{r}_j}{\mathrm{E}\{\|\mathbf{r}_i - \mathbf{r}_j\|\}\mathrm{E}\{\|\mathbf{r}_i + \mathbf{r}_j\|\}},$$

for the initial velocity field.

4.4.7 Convergence Analysis for Arbitrary Topologies

Before starting with the convergence analysis, we recapitulate the AC update equation with blending (represented by the blending matrix $\mathbf{C}$ that contains the velocities v_{ij}):

$$\mathbf{x}[k] = \widetilde{\mathbf{W}}\mathbf{x}[k-1] = (\mathbf{W} - \mathbf{C})\,\mathbf{x}[k-1]\,.$$

Here, $\mathbf{W}$ denotes the conventional symmetric AC weight matrix and $\widetilde{\mathbf{W}}$ is the weight matrix that includes the blending process. The elements of $\widetilde{\mathbf{W}}$ are defined as $\tilde{w}_{ij} = w_{ij} - v_{ij}$. As discussed in [19] a necessary condition for asymptotic convergence (for any input) is given by

$$\rho\Big\{\tilde{\mathbf{W}} - \frac{1}{I}\mathbf{1}\mathbf{1}^T\Big\} < 1\,.$$

A more stringent condition is the per-step stability, i.e.,

$$\Big\|\widetilde{\mathbf{W}} - \frac{1}{I}\mathbf{1}\mathbf{1}^T\Big\|_2 < 1\,.$$

This condition has the advantage that it can be used as a side constraint in a convex optimization problem (cf. (4.33)). In the symmetric case, of course, both conditions are equivalent.

Unfortunately, these conditions are rather hard to check in a distributed fashion. The following theorem provides local conditions, but it is restricted to non-negative matrices $\widetilde{\mathbf{W}}$.

Theorem 3. *Average consensus with skew-symmetric blending matrix* $\mathbf{C}$ *converges asymptotically if*

(a) the underlying graph is connected,

(b) the consensus weights w_{ij} *are positive,*

(c) the condition $\sum_{i \neq j} w_{ij} < 1$ *holds, and*

(d) the advection coefficients are smaller than the diffusion, i.e., $v_{ij} \leq w_{ij}$.

*Proof:*The proof uses methods which were already introduced in Section 4.1.1. First of all, we want to show that $\widetilde{\mathbf{W}}$ has only one eigenvalue equal to 1. Therefore, $\mathbf{L_w} - \mathbf{C}$ has to be positive semidefinite with only one zero eigenvalue. Hence, $\mathbf{v}^T(\mathbf{L_w} - \mathbf{C})\mathbf{v} \geq 0$ has to be fulfilled. Since $\mathbf{v}^T\mathbf{C}\mathbf{v} = 0$ because $\mathbf{C}$ is skew-symmetric, we end up with the same condition as for AC with positive weights, i.e., if and only if (a) the underlying graph is connected and (b) all weights are positive, we obtain only one zero eigenvalue. The second part of the proof investigates the conditions such that $\rho\{\mathbf{L_w} - \mathbf{C}\} < 2$ which is equivalent to $\lambda_i(\widetilde{\mathbf{W}}) > -1$. Using the Gerschgorin circle theorem we have

$$\left|\lambda(\mathbf{L_w}) - d_{\mathbf{w}}(i)\right| \leq \sum_{j \neq i} |w_{ij} + v_{ij}|\,,$$

where $d_{\mathbf{w}}(i) = \sum_{j \neq i} w_{ij}$. If we assume (d) $v_{ij} \leq w_{ij}$ then we obtain

$$\left|\lambda(\mathbf{L_w}) - d_{\mathbf{w}}(i)\right| \leq d_{\mathbf{w}}(i)\,,$$

and therefore (c) $d_{\mathbf{w}}(i) < 1$ ensures $\rho\{\mathbf{L_w} - \mathbf{C}\} < 2$.

□

We obtain the same result with the von Neumann analysis in Appendix 4.A. For grid graphs, however, the sufficient results of Section 4.4.5 based on the von Neumann analysis are not restricted to a non-negative $\widetilde{\mathbf{W}}$ and therefore allows more freedom for choosing the blending matrix $\mathbf{C}$.

4.4.8 Numerical Results

The spectral radius $\rho\{\mathbf{W} - \frac{1}{I}\mathbf{1}\mathbf{1}^T\}$ is an indicator for the convergence speed of AC (cf. [19]). Therefore, we show the impact of the diffusion (equivalent to a constant weight design of $\mathbf{W}$) and advection component on the spectral radius in Fig. 4.11 for different advective flows in uniform grid WSN with 625 nodes. Besides Anosov flow [35] we used rotational velocity fields where 12 rotation rings are grouped into four groups of three, each group rotating in the opposite direction (see the rightmost image in Fig. 4.10). It is seen that for FTBS discretization the regular rotational flow yields almost the same minimum spectral radius as the periodically changing Anosov flow, but for our discretization proposed in Section 4.4.2 the minimum spectral radius is slightly smaller. For the non-toroidal geometry with boundaries, the minimum spectral radius hardly changes. Moreover, we can conclude that the proposed sampling performs best with maximum diffusion ($\alpha \approx 1/4$) and

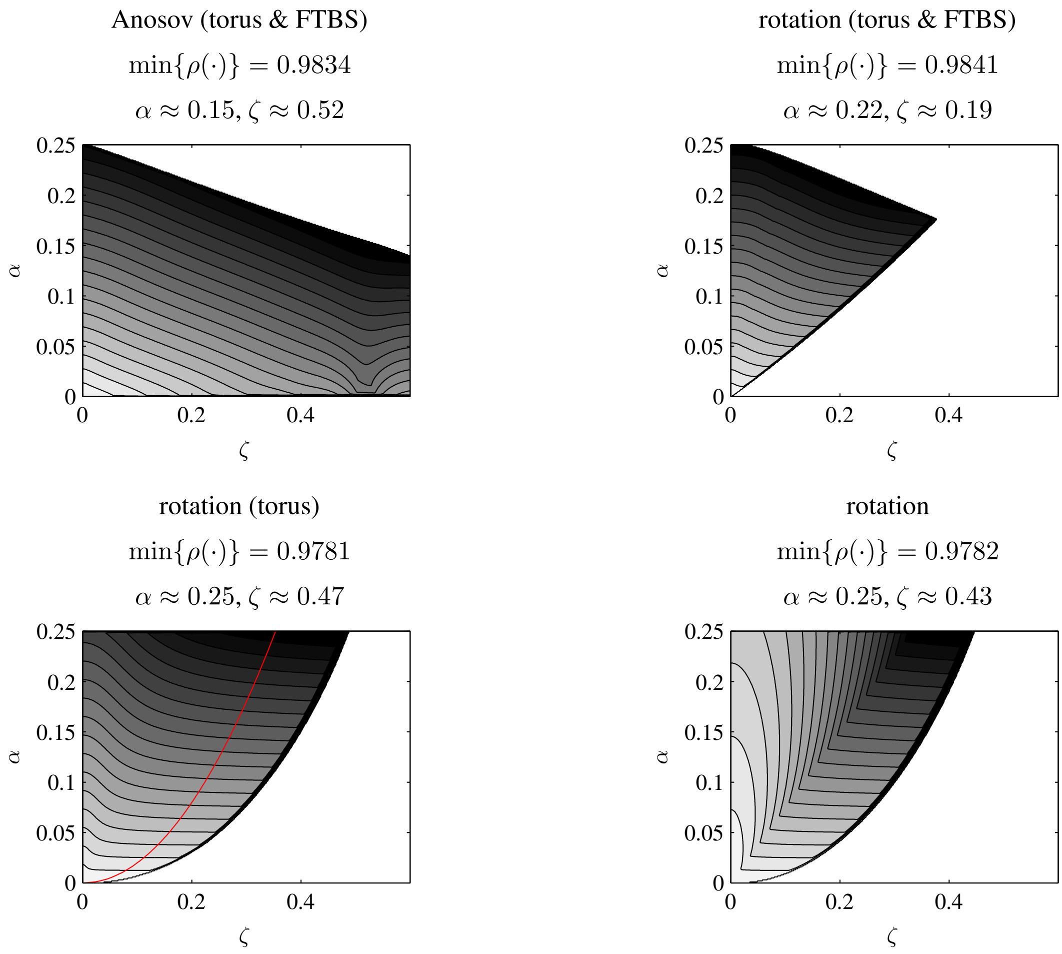
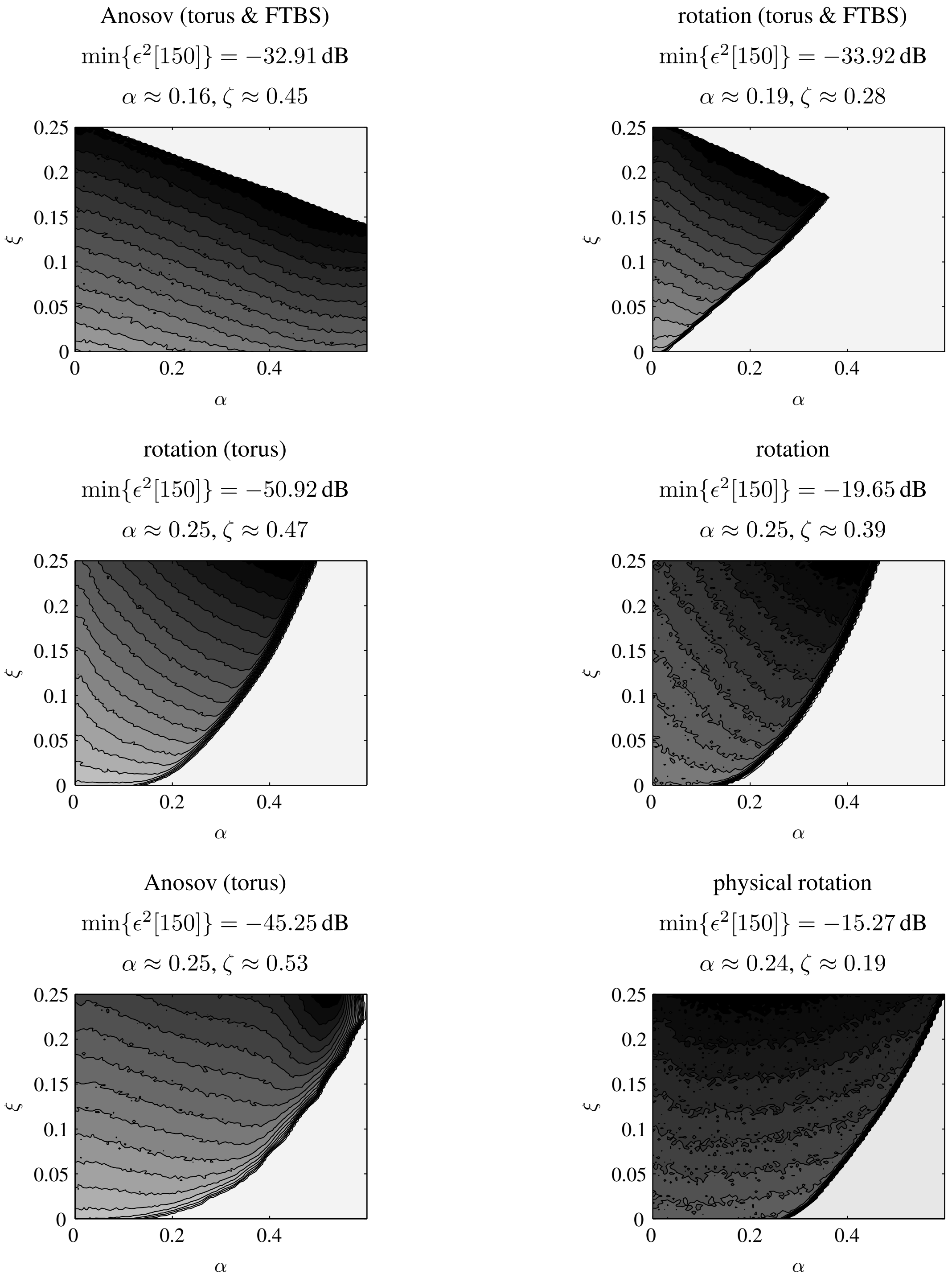


Figure 4.11: *Spectral radius for different advection and diffusion parameters on uniform grid WSN with* 625 *nodes (the additional line in the third plot indicates the theoretical sufficient condition for convergence).*

advection. Numerical MSE results shown in Fig. 4.12 corroborate these findings and demonstrate even stronger performance improvements with our proposed method. Our simulation considers a setting with 625 nodes, an initial low-pass field with $\theta_c^{(s)} = 0.01$, and 500 Monte-Carlo experiments. The MSE is shown after 150 iterations for various advection models and different parameters. The first four settings are identical to Fig. 4.11. The last two show Anosov flow with our novel discretization and a physical rotational field as shown in Fig. 4.10 on the left. In the plots it is seen that the MSE optimum points almost coincide with the points with minimal spectral radius in Fig. 4.11, whereas the deviation of the optima for the FTBS discretization is the largest. This implies that good advection and diffusion parameters can be empirically determined by investigating the spectral radius before starting extensive numerical investigations. Moreover, when we compare the MSE of the optimum Anosov setting as proposed in [35] and our rotation field on a torus, it is seen that our method has a gain of around 23 dB. But also Anosov flow with our discretization yields better performance than for FTBS discretization. Here, because of our discretization we have again the advantage that the optimum point requires almost maximum diffusion. Finally, when the WSN does not lie on a torus (i.e., the boundaries are not connected), the performance is much worse, but still better than for the non-advection case. We also observed that for an increased number of nodes, the difference to the WSN on a torus becomes smaller, because the


Anosov (torus & FTBS)

$\min\{\epsilon^2[150]\} = -32.91\,\mathrm{dB}$

$\alpha \approx 0.16, \zeta \approx 0.45$

rotation (torus & FTBS)

$\min\{\epsilon^2[150]\} = -33.92\,\mathrm{dB}$

$\alpha \approx 0.19, \zeta \approx 0.28$

rotation (torus)

$\min\{\epsilon^2[150]\} = -50.92\,\mathrm{dB}$

$\alpha \approx 0.25, \zeta \approx 0.47$

rotation

$\min\{\epsilon^2[150]\} = -19.65\,\mathrm{dB}$

$\alpha \approx 0.25, \zeta \approx 0.39$

Anosov (torus)

$\min\{\epsilon^2[150]\} = -45.25\,\mathrm{dB}$

$\alpha \approx 0.25, \zeta \approx 0.53$

physical rotation

$\min\{\epsilon^2[150]\} = -15.27\,\mathrm{dB}$

$\alpha \approx 0.24, \zeta \approx 0.19$

Figure 4.12: *MSE for different advection and diffusion parameters on uniform grid WSN with* 625 *nodes.*

impact of the link across the boundaries decreases. In Fig. 4.12 it is also seen that the performance of regular rotation as described above is significantly better than the physical motivated field.

The MSE performance of our proposed method with random geometric graphs is presented in the next section, where we compare the convergence with other weight methods.

4.5 Comparison and Conclusion

In this section we compare our different weight designs and conclude our findings. The MSE as defined in Section 2.2.5 is used as performance metric. We always consider random geometric graphs with 100 nodes, since the graph is sufficiently large while the simulation complexity is still moderate. We also think that such a number of nodes could be realistic in practice. Having a much smaller number of sensors would suggest to use flooding schemes or routing protocols instead of iterative algorithms, since their complexity and precision becomes very attractive then.

For the MSE optimum weights (cf. Section 4.2), we present four different cases:

- The *optimum symmetric* weights denote the per-step MSE optimum weights, which do not necessary fulfill the convergence property of the spectral radius.
- Including the spectral radius condition yields the *optimum stable* weights, which ensure convergence in each step, even for misalignment.
- Then the optimization (on the spectral radius constraint) is only done in the first iteration and the *initial optimum stable* weights are also used for the subsequent iterations.
- Finally, the *initial optimum (i.i.d.)* weights consider a setting where the optimization is also applied in the first iteration, but assuming an i.i.d. setting, even if the correlation between the measurements is different.

When simulating our weight morphing method (cf. Section 4.3), we use for the edge-wise morphing function

$$\alpha_{ij}[k] = \frac{1}{1 + \exp\left(100\left(\log\left(m_{ij}^{(1)}[k]\right) - \log(0.3)\right)\right)},$$

which was empirically designed to yield good performance with initial uncorrelated and zero mean measurements.

Finally, for our blending scheme (Section 4.4) we present the results for both stability conditions, the global and the local, respectively. The curves with the latter condition are labeled ’v2’.

4.5.1 MSE Convergence for Static Averaging

The first simulation considers the MSE convergence in WSN with geometric connectivity of $c=2$. We compare the performance of the weight design methods introduced in this chapter for an initial i.i.d. field and a low-pass field with $\theta_{\mathrm{c}}^{(\mathrm{s})}=0.01$. The evolution of the MSE over the number of iterations is illustrated in Fig. 4.13, where the results are averaged over around 500 scenarios. Moreover in Fig. 4.14 we plot the MSE after 80 iterations versus the cut-off frequency $\theta_{\mathrm{c}}^{(\mathrm{s})}$ to provide a different perspective.

In the case of uncorrelated measurements it is seen that the CVX weights initially perform poorly but outperform the MH weights after around 50 iterations. When the measurements are correlated, the CVX weights are always superior to the MH weights.

Our weight morphing scheme achieves in the considered settings always at least the best performance of MH and CVX weights, except when $\theta_{\mathrm{c}}^{(\mathrm{s})}=0.1$ where the morphing result is slightly ($\sim 2\,\mathrm{dB}$) worse compared to the CVX. A different choice of the morphing function would alter this behavior.

Considering the optimum symmetric weights, it is seen that they provide a lower bound for all time-varying symmetric weights, i.e., weight morphing lies around $5\,\mathrm{dB}$ above the bound after 100 iterations for the i.i.d. setting. For low-pass fields the optimum weights lie even further below weight morphing.

Involving the spectral radius condition in the optimization, there is almost no difference for initial uncorrelated measurements, i.e., the additional constraint does not affect the optimization problem significantly. For correlated measurements, however, the stability condition yields a huge performance deterioration, since it does not allow large weights which help to obtain fast convergence for flat fields. Anyway, the stable optimum therefore provide a lower bound for all stable time-varying weight designs and as Fig. 4.14 illustrates, our weight morphing scheme uniformly lies $5\,\mathrm{dB}$ above this bound.

The results for the weights that were optimized in the first iteration (according to Section 4.2) and then kept constant are close to the MH case in the i.i.d. setting. When considering the case when the weights are optimized for an i.i.d. setting and then left constant is also slightly better than the MH weights for all types of $\theta_{\mathrm{c}}^{(\mathrm{s})}$, which is not true for the weights optimized for the real initial performance under the stability constraint. These weights yield an excellent decay in the first iteration but perform worse for non i.i.d. fields.

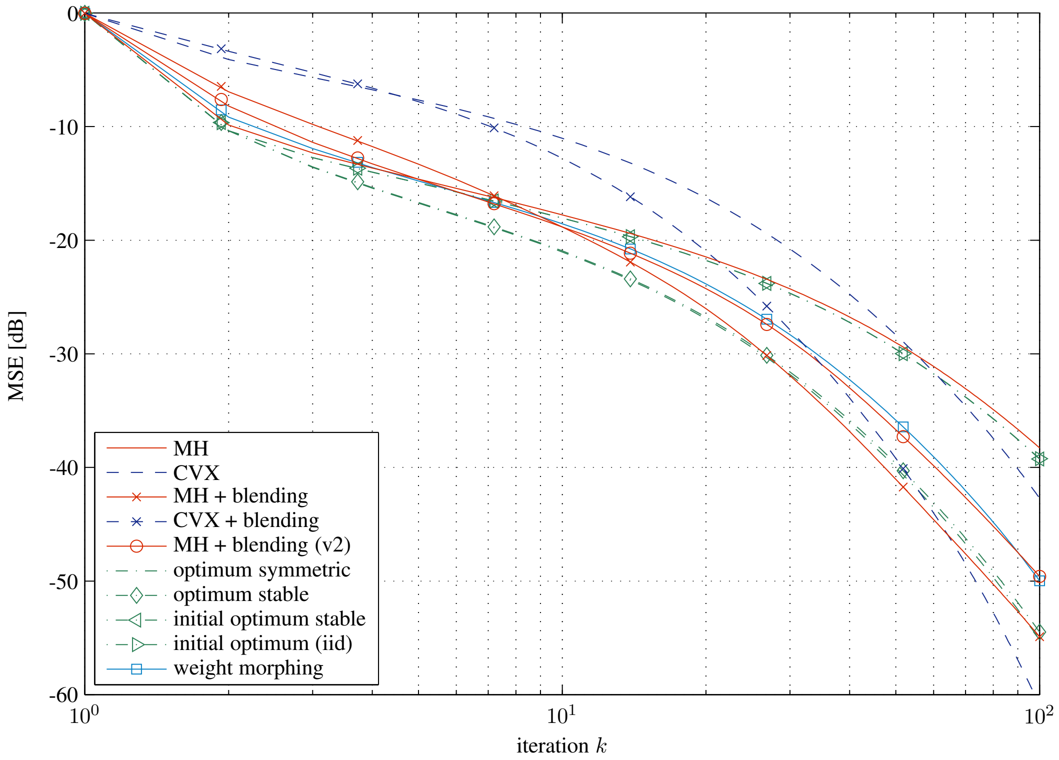
When we apply our blending method of Section 4.4 we see that it achieves superior performance for almost all spatial-frequencies, only the optimum symmetric weights yield better performance at small $\theta_{\mathrm{c}}^{(\mathrm{s})}$. In relation to the corresponding weight design without blending, we obtain a gain for MH weights of around $10\,\mathrm{dB}$ and $13\,\mathrm{dB}$ for CVX weights. This gain is almost independent of the spatial cut-off frequency. Moreover, we see in Fig. 4.14 that MH plus blending achieves almost the same performance as weight morphing and CVX weights, but unfortunately our blending approach for WSN requires global processing so far, as CVX weights do.

In Fig. 4.15 we show the MSE after 80 iterations for the same setting as above with $\theta_{\mathrm{c}}^{(\mathrm{s})}=0.5$ for different graph connectivities c. It is seen that the performance increases for larger connectivities, whereas the ranking remains almost unchanged. The gain of weight morphing compared to CVX weights is around $5\,\mathrm{dB}$ almost independent of the connectivity. Finally it is seen that the effect of blending increases with the connectivity, i.e., the performance gain for $c=2.4$ equals $20\,\mathrm{dB}$, which is quite significant.

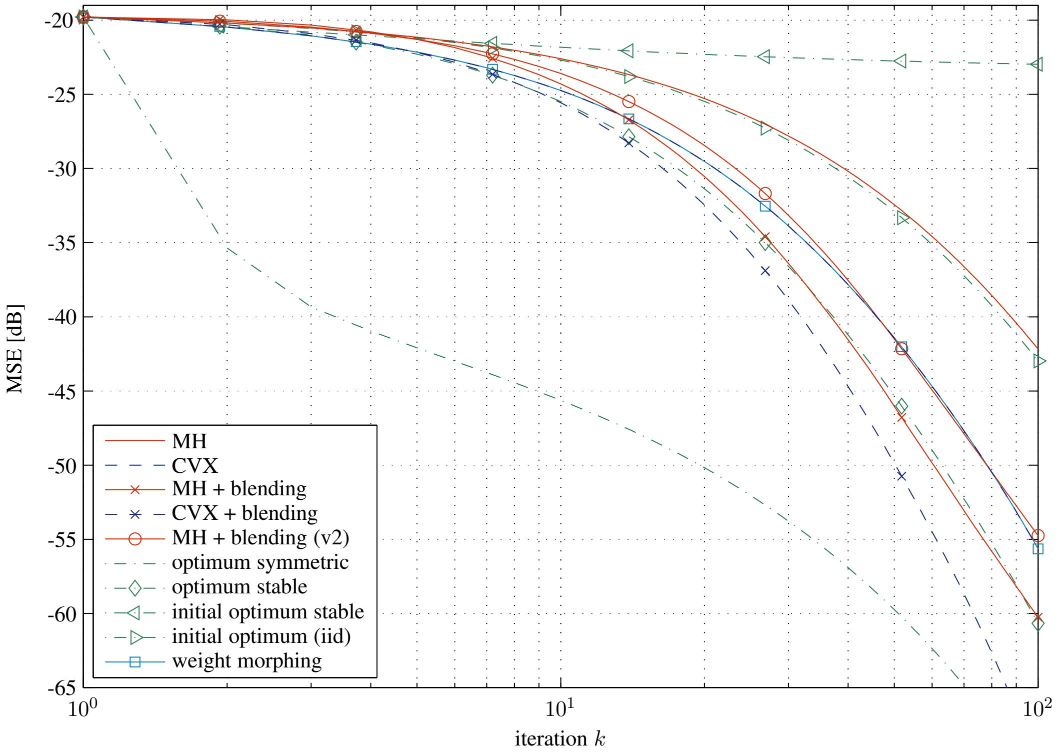
4.5.2 MSE Performance of Dynamic Averaging

Our next numerical investigation considers a dynamic setting, i.e., the measurements vary over time, but the graph structure remains constant (cf. Chapter 3). In the dynamic regime we did not consider our optimal weight design method, since the computation of the corresponding correlation is not trivial. The setting in this section is the same as in the previous, only the field varies according to a low-pass process with cut-off frequency $\theta_{\mathrm{c}}^{(\mathrm{t})}$. For simplicity we also kept the design of the morphing function. The result of the MSE averaged over 1000 Monte-Carlo runs is shown in Fig. 4.16 for two different spatial low-pass fields versus $\theta_{\mathrm{c}}^{(\mathrm{t})}$.

First of all, we compare the MSE of MH and CVX weights. It is seen that MH yields the best performance for the case where the measurements are temporally i.i.d., otherwise CVX is superior. With weight morphing it is possible to switch between both weight designs as in the static case and achieve overall good performance.



(a) uncorrelated states



(b) correlated states

Figure 4.13: *MSE convergence for (a) initially uncorrelated and (b) correlated states for different weight design schemes.*

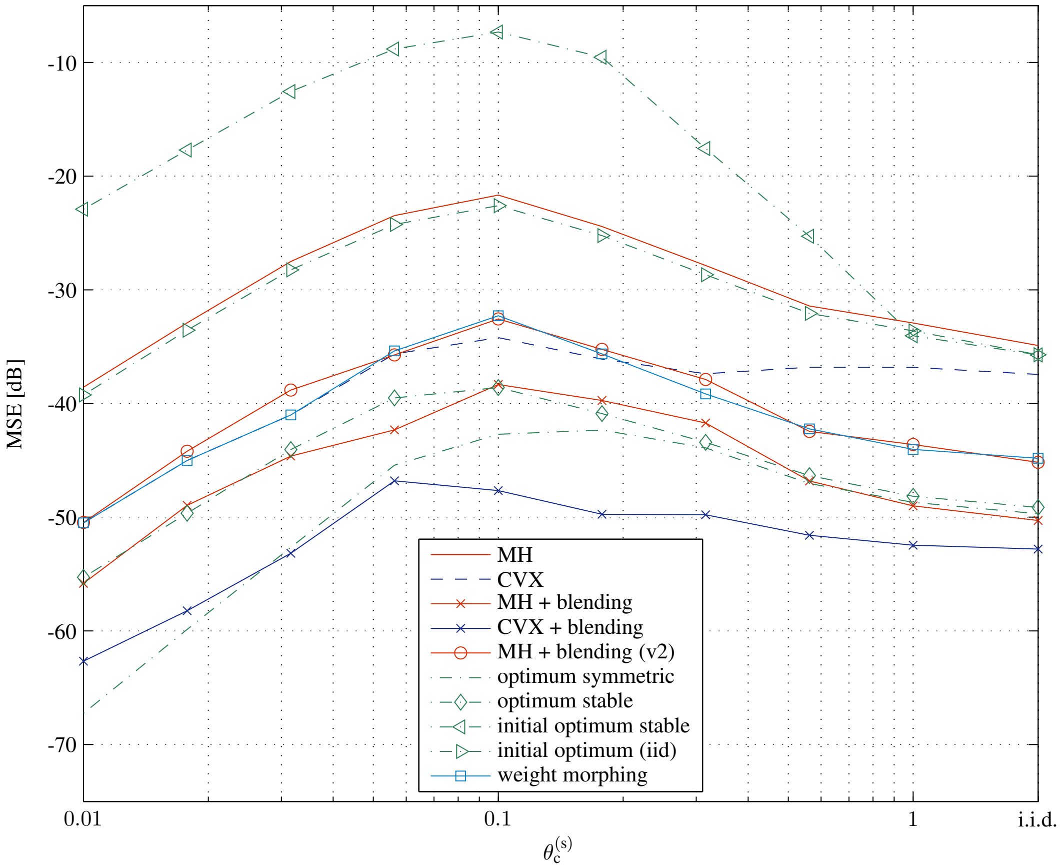


Figure 4.14: *MSE achieved with different different weight designs after 80 iterations versus spatial cut-off frequency of the measured field.*

In Fig. 4.16 weight morphing does not behave as well as MH in the i.i.d. case for $\theta_c^{(t)} = 0.01$ which is due to the fact that the morphing function was not aligned perfectly for the simulated scenario.

Investigating the blending scenarios, we can conclude that for fast fluctuations the MSE is decreased by the advection flow. In particular, the ranking is even reversed, i.e., CVX weights plus blending achieves the best performance for slowly varying fields, but for fast varying fields it yields the worst performance. This can be explained by the property of blending that a direction is added to the information spread, which is beneficial for the convergence speed but works against fast local averaging. In the case of fast varying fields, local averaging is the dominantly factor for yielding a small MSE.

4.5.3 Conclusions

In this chapter we theoretically investigated the MH weights, in particular their convergence properties. We have seen that in almost every practical WSN the widely used regularization term (“+1”) is not necessary for the convergence of AC and only decreases the convergence speed. Then we have shown how to design the weights in average consensus in order to achieve the best MSE performance. We derived a closed-form solution of this problem in terms of a linear system of equations. Optimizing the weights in each step may be practically challenging but provides useful performance bounds for the convergence. In our simulations we

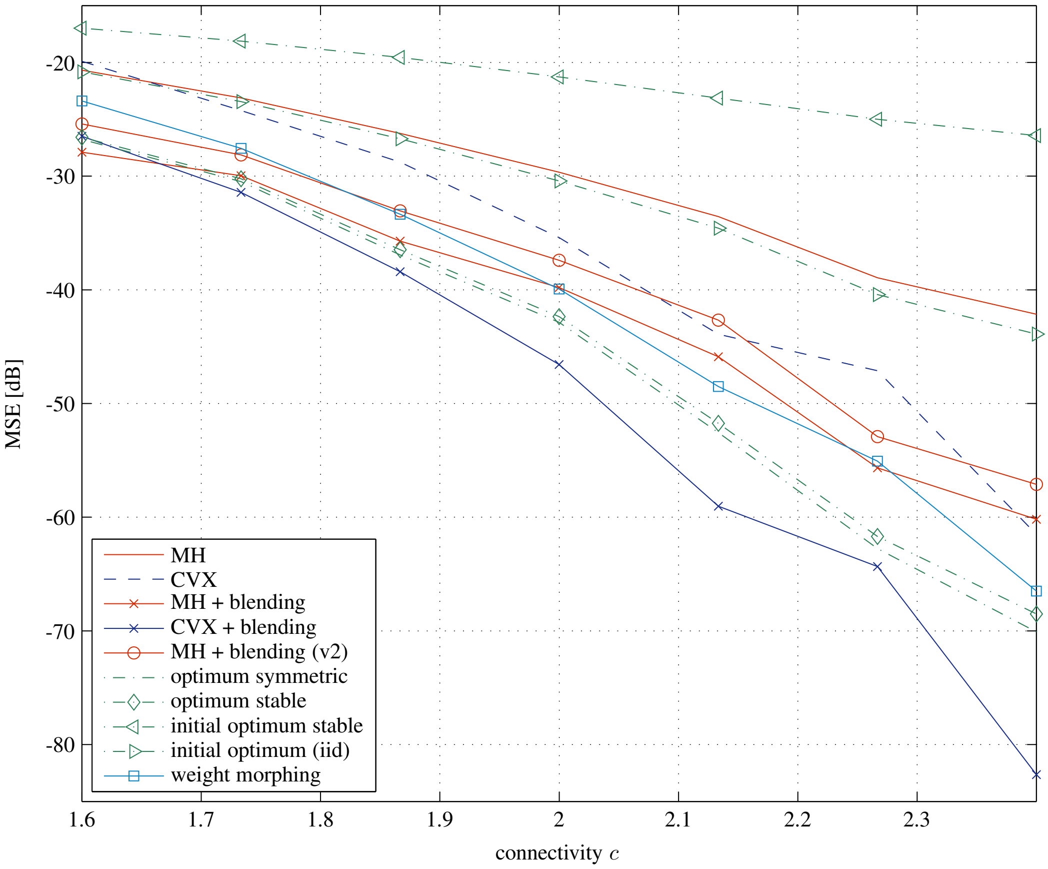


Figure 4.15: *MSE achieved with different different weight designs after 80 iterations versus the graph connectivity* c*.*

have shown how the correlations of the states change over time and that the graph topology plays an important role. The finding that time-varying weights can improve the performance motivated us to propose an adaptive weight morphing scheme which leads to a nonlinear AC scheme. Our method combines the favorable transient behavior of MH weights with the excellent asymptotic performance of CVX weights. Moreover, morphed weights are advantageous also in the dynamic case to deal with different noise levels and signal variations. Finally, we have shown that through an appropriate discretization method it is possible to augment classical AC schemes with rotational flows, thereby achieving blending schemes that achieve a faster mixing of the data. Compared to [35], our approach requires neither toroidal geometries nor a time-varying flow. Our numerical results verify the superiority of rotational flows. In particular, their use in general WSNs provides a tremendous performance increase and even outperforms the per-step optimum symmetric weights. In dynamic scenarios, however, blending provides only a benefit if the temporal variation is slow enough.

4.A Alternative Convergence Analysis

In this section we extend the von Neumann analysis of [47] for advection-diffusion so that it provides sufficient convergence conditions for blending with arbitrary topologies. With the von Neumann analysis the propagation

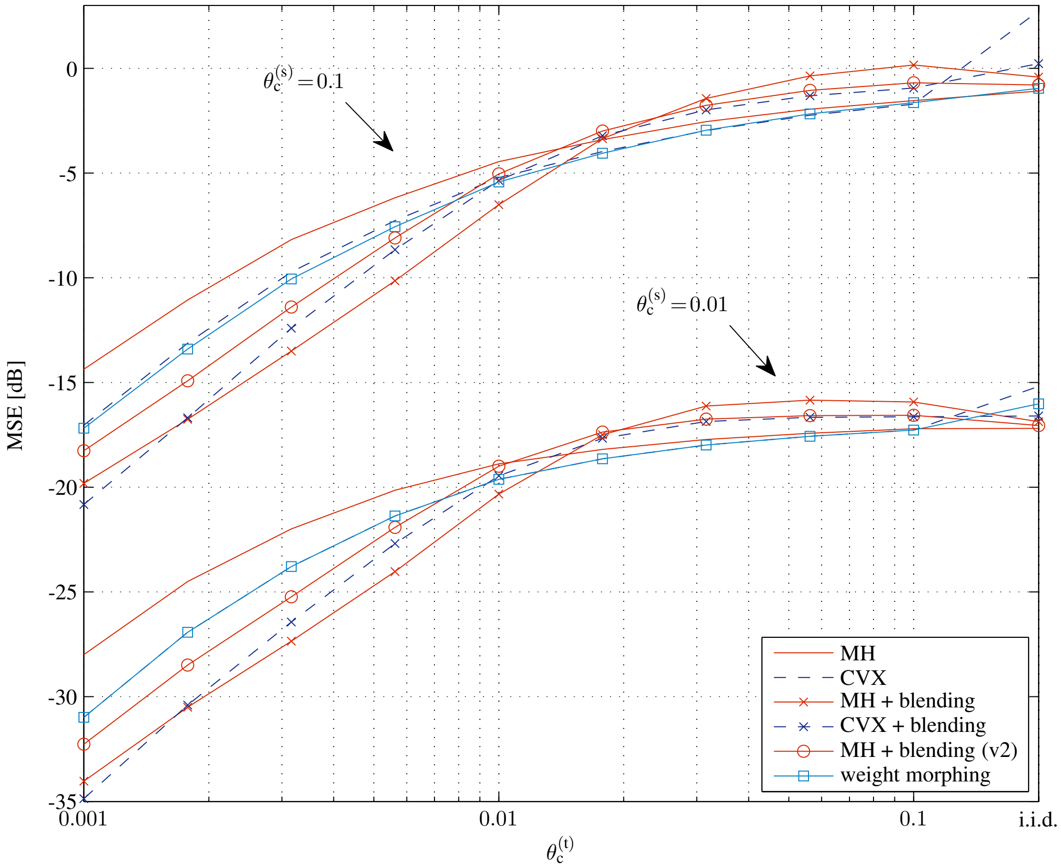


Figure 4.16: *MSE performance of time-varying low-pass fields for different weight design schemes.*

conditions of all possible spatial Fourier modes are investigated. If all of them decrease over time it can be concluded for the given discretization all possible advection-diffusion fields (with no sources and sinks) evolve to a constant flat field as the time increases, similar to the continuous case.

To this end, we have the AC update equation

$$\mathbf{x}[k] = \widetilde{\mathbf{W}}\mathbf{x}[k-1] = (\mathbf{W} - \mathbf{C})\,\mathbf{x}[k-1]\,,$$

where $\mathbf{W}$ is the symmetric weight matrix (diffusion) and $\widetilde{\mathbf{W}}$ is the asymmetric AC weight matrix, which includes the advection process, i.e., $\tilde{w}_{ij} = w_{ij} - v_{ij}$. Moreover we define the error vector $\mathbf{e}[k]$ as

$$\mathbf{e}[k] = \mathbf{x}[k] - \frac{1}{I}\mathbf{1}\mathbf{1}^T\mathbf{x}[k]\,.$$

It represents the deviation of each inner state from the true mean. Using this error vector, the MSE at iteration k is given by

$$\epsilon[k] = \|\mathbf{e}[k]\|_2^2\,,$$

and the maximum squared error reads

$$\tilde{\epsilon}[k] = \|\mathbf{e}[k]\|_\infty^2\,.$$

Due to the properties of $\widetilde{\mathbf{W}}$ (every constant vector lies in its null space, and so does the average vector) we can formulate the temporal evolution of the error vector as

$$\mathbf{e}[k] = \widetilde{\mathbf{W}}\mathbf{e}[k-1]\,.$$

For the convergence of AC it is necessary that the MSE converges to zero for $k \to \infty$. An equivalent formulation is that the infinity norm converges to zero, which is ensured by a step-wise decrease, i.e., $\tilde{\epsilon}[k+1] < \tilde{\epsilon}[k]$. This can be guaranteed through the local decay $e_i[k+1] < e_i[k]$, where $e_i[k]$ is the ith element of $\mathbf{e}[k]$. Therefore, we are investigating the local error updates in the following which are defined as

$$e_i[k+1] = e_i[k] + \sum_{j\in\mathcal{N}(i)} w_{ij}(e_j[k] - e_i[k]) - \sum_{j\in\mathcal{N}(i)} v_{ij}e_j[k]\,.$$

We can represent the error $e_i[k]$ as the sum of Fourier modes (the index c denotes the corresponding mode, subsequently). At node i we have at most $d_i = |\mathcal{N}(\cdot)|$ spatial frequencies on which the Fourier modes can depend. Hence, in a graph we have $M \leq \max\{d_i\}$ frequency components. All those spatial frequencies have to propagate in orthogonal directions. To that end, using the Fourier modes $f_i^{(c)}[k]$ (cf. Section 4.4.5) and defined in (4.34) with c indexing the mode number we obtain

$$e_i[k] = \sum_c f_i^{(c)}[k] = \sum_c \left(\xi'^{(c)}\right)^k \exp\left\{\iota \sum_{m=1}^{d_i} (\boldsymbol{\theta}')_m\, (\mathbf{p}'_i)_m\right\} = \sum_c \left(\xi^{(c)}\right)^k \exp\left\{\iota \sum_{m=1}^{M} (\boldsymbol{\theta})_m\, (\mathbf{p}_i)_m\right\}$$

where $(\boldsymbol{\theta})_m$ is the phase of the spatial wave which propagates in the dimension m and $\mathbf{p}_i$ denotes the virtual position of node i (which is related to the connectivity pattern of the graph and not to the geographical position). The variables with primes ($\boldsymbol{\theta}'$, $\mathbf{p}'$) are low-dimensional projections of $\boldsymbol{\theta}$ and $\mathbf{p}$ and hence the corresponding Fourier representation is better adjusted locally. But we will use the more general expression, since it is valid for every node. The phase is defined as $-\pi \leq (\boldsymbol{\theta})_m \leq \pi$. An additional requirement for the virtual sensor positions is that they fulfill $d(\mathbf{p_i}, \mathbf{p_j}) = 1$ where $d(\cdot,\cdot)$ denotes the distance. This covers the worst case, where the values of two neighboring nodes are antipodal at phase π and $-\pi$.

The required geometric properties cannot be achieved in a Euclidean space. Therefore, we switch to an $I-1$ dimensional Riemannian manifold (e.g. a hypersphere) where the points $\mathbf{p}_i$ are distributed such that the geodesics between the nodes have distance one (i.e., $d(\mathbf{p_i}, \mathbf{p_j}) = 1$) and the geodesics are orthogonal in each tangential space defined by the points $\mathbf{p}_i$. At each geodesic one wave propagates with phase $\theta_{i\to j}$ (when we consider the geodesic between node i and j) that corresponds to one element of $\boldsymbol{\theta}$.

The Fourier modes are defined as (we omit the index c for simplicity)

$$f_i[k] = \xi^k \exp\left\{\iota \sum_{m=1}^{I-1} (\boldsymbol{\theta})_m\, (\mathbf{p}_i)_m\right\}\,, \tag{4.34}$$

and

$$f_j[k] = \xi^k \exp\left\{\iota \sum_{m=1}^{I-1} (\boldsymbol{\theta})_m\, (\mathbf{p}_i)_m\right\} \exp\left\{\iota\, \theta_{i\to j}\right\} = f_i[k] \exp\left\{\iota\, \theta_{i\to j}\right\}\,. \tag{4.35}$$

Due to linearity we focus only on one Fourier mode and obtain the update equation,

$$f_i[k+1] = f_i[k] + \sum_{j\in\mathcal{N}(i)} w_{ij}(f_j[k] - f_i[k]) - \sum_{j\in\mathcal{N}(i)} v_{ij} f_j[k] \,. \tag{4.36}$$

Inserting (4.34) and (4.35) into (4.34) we obtain

$$\xi = 1 - \sum_{j\in\mathcal{N}(i)} w_{ij}\left(1 - \exp\{\iota\theta_{i\to j}\}\right) - \sum_{j\in\mathcal{N}(i)} v_{ij} \exp\{\iota\theta_{i\to j}\} \;,$$

since we have $\sum_{j\in\mathcal{N}(i)} v_{ij} = 0$ we can simply subtract this expression and obtain,

$$\begin{aligned}|\xi|^2 &= \left(1 - \sum_{j\in\mathcal{N}(i)} (w_{ij} + v_{ij})\left(1 - \cos\theta_{i\to j}\right)\right)^2 + \left(\sum_{j\in\mathcal{N}(i)} (w_{ij} + v_{ij}) \sin\theta_{i\to j}\right)^2 \\ &= \left(1 - \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\left(1 - \cos\theta_{i\to j}\right)\right)^2 + \left(\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij} \sin\theta_{i\to j}\right)^2 \;. \end{aligned} \tag{4.37}$$

Therefore, we have the similar structure as in [47]. First we consider the case when $\boldsymbol{\theta} = \pi\mathbf{1}$ which yields:

$$|\xi|^2 = \left(1 - 2\sum_{j\in\mathcal{N}(i)} (w_{ij} + v_{ij})\right)^2 < 1 \,,$$

and therefore we obtain $\sum_{j\in\mathcal{N}(i)} w_{ij} < 1$ which is a necessary condition for the convergence of AC with advection. Surprisingly, this condition does not involve the advection and is equivalent the conventional convergence condition for AC with positive weights (cf. Section 4.1.1).

Following [47] we consider the limiting case, when $\theta_{i\to j} \to 0$ with $|\theta_{i\to j}| \leq \theta$ for all m. Considering the Taylor expansion we obtain

$$\begin{aligned}|\xi|^2 &= \left(1 - \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\frac{\theta_{i\to j}}{2} + \mathcal{O}(\theta^4)\right)^2 + \left(\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\theta_{i\to j} + \mathcal{O}(\theta^3)\right)^2 \\ &= 1 - \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\theta_{i\to j}^2 + \left(\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\theta_{i\to j}\right)^2 + \mathcal{O}(\theta^4) \\ &= 1 - \boldsymbol{\theta}_i\left(\operatorname{diag}\{\mathbf{u}_i\} - \mathbf{u}_i\mathbf{u}_i^T\right)\boldsymbol{\theta}_i^T + \mathcal{O}(\theta^4) \,, \end{aligned}$$

where $\tilde{w}_{ij}$ for all $j \in \mathcal{N}(i)$ is stacked into $\mathbf{u}_i$. To ensure that $|\xi|^2$ is smaller than 1 the matrix $\operatorname{diag}\{\mathbf{u}_i\} - \mathbf{u}_i\mathbf{u}_i^T$ has to be positive definite. Therefore, the diagonal entries have to be at least positive, i.e., $(w_{ij} + v_{ij}) - (w_{ij} + v_{ij})^2 > 0$ and hence, $(w_{ij} + v_{ij}) < 1$ and $(w_{ij} + v_{ij}) > 0$. Analogously to [47] sufficiency of positive definiteness can be shown, but no additional condition arises.

So far we have two conditions for necessity ($\sum_{j\in\mathcal{N}(i)} w_{ij} < 1$ and $0 < (w_{ij} + v_{ij}) < 1$) that can be shown to be also sufficient (according to [47]). Using the Cauchy-Schwarz inequality we have

$$\left(\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij} \sin\theta_{i\to j}\right)^2 = \left(\sum_{j\in\mathcal{N}(i)} \sqrt{\tilde{w}_{ij}}\left(\sqrt{\tilde{w}_{ij}} \sin\theta_{i\to j}\right)\right)^2$$

$$\leq \underbrace{\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}}_{<1} \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij} \sin^2 \theta_{i\to j}$$
$$< \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij} \sin^2 \theta_{i\to j} \,.$$

A similar bound is obtained for the $1 - \cos\theta_{i\to j}$ expression in (4.37). Using these inequalities, we insert our necessary conditions into (4.37), i.e.,

$$\begin{aligned}
|\xi|^2 &= \left(1 - \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\left(1-\cos\theta_{i\to j}\right)\right)^2 + \left(\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij} \sin\theta_{i\to j}\right)^2 \\
&= 1 - 2\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\left(1-\cos\theta_{i\to j}\right) + \left(\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\left(1-\cos\theta_{i\to j}\right)\right)^2 + \left(\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij} \sin\theta_{i\to j}\right)^2 \\
&< 1 - 2\sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\left(1-\cos\theta_{i\to j}\right) + \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\left(1-\cos\theta_{i\to j}\right)^2 + \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij} \sin^2\theta_{i\to j} \\
&= 1 - \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\left(2 - 2\cos\theta_{i\to j} - 1 + 2\cos\theta_{i\to j} - \cos^2\theta_{i\to j} - \sin^2\theta_{i\to j}\right) \\
&= 1 \,,
\end{aligned}$$

which proves that the necessary conditions are also sufficient.

Our stability result is for a stronger setting as in practice, i.e., we assumed more distinct frequencies than actually can propagate in practice and moreover the smallest possible frequency is definitely larger than 0 since we assume graphs with finite I. Therefore, we have $\theta \geq \frac{2\pi}{I}$ which actually tends to zero if I goes to infinity. Hence, in such scenarios our stability conditions are sufficient, but not necessary. In the following we give an example.

Illustrative example. We consider a graph that consists of three nodes and is fully connected. The graph is shown in Fig. 4.17 on the left hand side. The transformation into a higher dimensional graph located on a sphere is illustrated in the center of Fig. 4.17. It is seen that the geodesics are orthogonal in each tangential space. Apparently, only one frequency can propagate along such a graph, which is a special case of all possible frequency assignments. For that purpose we do a projection into a lower dimension, i.e., we assume a nonlinear projection such that the geodesics are antipodal afterwards but the lengths remain the same. The result is seen on the right side of Fig. 4.17. The dashed line represents a camera projection of the graph on the sphere, when the camera is located outside of the sphere at the point where all three tangential spaces intersect, but for simplicity we assume that we have a projection where we obtain a circle on which the geodesics between the nodes are again 1. This graph can be also redrawn as a 1D graph (line graph) with periodic structure. Finally, it is seen that the lowest frequency that can propagate is $\frac{2}{3}\pi$.

In the following we provide a mathematical description of the example. The formulation of the Fourier modes of the nodes distributed on the sphere is

$$f_i = \xi \exp\{\iota \boldsymbol{\theta}^T \mathbf{p}_i\} \,,$$

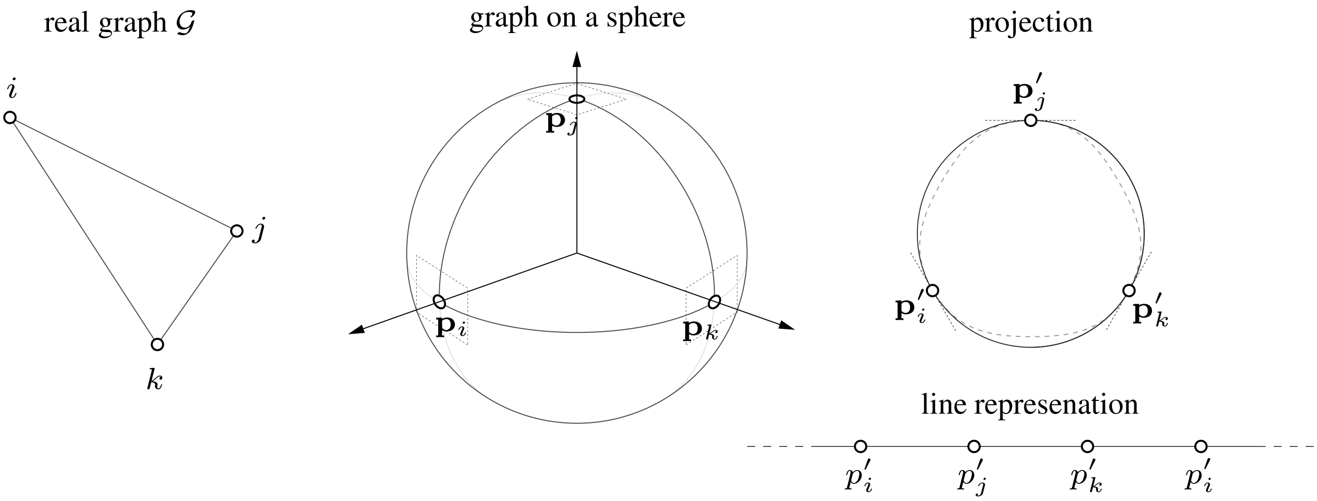


Figure 4.17: *Illustrative example of the node placement on a sphere for analyzing the Fourier modes of the error of AC with blending.*

$$
\begin{aligned}
f_j &= \xi \exp\{\iota \boldsymbol{\theta}^T \mathbf{p}_j\} = \xi \exp\{\iota \boldsymbol{\theta}^T (\mathbf{p}_i + \mathbf{e}_{i\to j})\} = f_i \exp\{\iota \theta_{i\to j}\}\,,\\
f_k &= \xi \exp\{\iota \boldsymbol{\theta}^T \mathbf{p}_k\} = \xi \exp\{\iota \boldsymbol{\theta}^T (\mathbf{p}_i + \mathbf{e}_{i\to k})\} = f_i \exp\{\iota \theta_{i\to k}\}\,,
\end{aligned}
$$

with $\mathbf{e}_{i\to j} = (0\,1\,0)^T$ and $\mathbf{e}_{i\to k} = (0\,0\,1)^T$. We have two different phases, which is clearly overdetermined for our system. Hence, we assume a transformation matrix $\mathbf{T}$ (which is actually a vector in our case) that transforms our system into one dimension. We have $\mathbf{T} = [1\,1\,\,-1]$ and without loss of generality we assume $\mathbf{T}\boldsymbol{\theta} = \theta$. Transforming the point $\mathbf{p}_i$ yields p'_i. Therefore, we obtain

$$
\begin{aligned}
f_i &= \xi \exp\{\iota (\mathbf{T}\boldsymbol{\theta})^T \mathbf{T}\mathbf{p}_i\} = \xi \exp\{\iota \theta p'_i\}\,,\\
f_j &= \xi \exp\{\iota (\mathbf{T}\boldsymbol{\theta})^T \mathbf{T}\mathbf{p}_j\} = \xi \exp\{\iota \theta (\mathbf{T}\mathbf{p}_i + \mathbf{T}\mathbf{e}_{i\to j})\} = f_i \exp\{\iota \theta\}\,,\\
f_k &= \xi \exp\{\iota (\mathbf{T}\boldsymbol{\theta})^T \mathbf{T}\mathbf{p}_k\} = \xi \exp\{\iota \theta (\mathbf{T}\mathbf{p}_i + \mathbf{T}\mathbf{e}_{i\to k})\} = f_i \exp\{-\iota \theta\}\,.
\end{aligned}
$$

Of course we could also take $\mathbf{T} = (1\,1\,1)$ as the transformation matrix, but this would lead (graphically speaking) to the case where the geodesics are congruent and not antipodal, i.e., waves are reflected, but sent back to a different node. It is obvious that for our setting one frequency is sufficient for the convergence study. Hence, our representation on the sphere involves settings which cannot occur in practice, because it is more general. Moreover the Fourier modes after the transformation do not include the property that node j is neighbor of node k which also excludes frequencies which cannot appear.

5

Average Consensus with Mobile Nodes

IN this chapter, we investigate the performance of AC in mobile WSN, i.e., WSN in which (some of) the nodes are moving. To this end, we extend the mean-square error (MSE) analysis of Section 4.2.1 to random geometric graphs including a motion model. Analytical bounds on the MSE performance of AC in mobile WSN and numerical simulations reveal that node mobility is beneficial in the sense that AC convergence is improved (equivalently, the nodes' transmit power or the number of nodes can be reduced). In particular, we show on the one hand that mobility does not increase the MSE, and on the other hand we come up with a lower bound on the MSE convergence that provide us with insights regarding the maximum performance gain resulting from mobility. Both results rely on the assumption that the statistical properties of each graph realization after node movement remain the same. Numerical results confirm our findings.

5.1 AC in Mobile WSN

In general, node mobility will result in a time-varying graph topology, i.e., the edge set (and hence the node neighbors and degrees) changes over time. We model the mobile WSN in terms of a stationary Markovian evolving graph [51] for which the node set V remains unchanged but the edge set changes over a time, thus forming a sequence of graph topologies $\mathcal{T} = \{\mathcal{E}_1, \ldots, \mathcal{E}_k\}$ that constitutes a stationary Markov chain. We denote the set of neighbors of node i at time k by $\mathcal{N}_k(i) = \{j : (i,j) \in \mathcal{E}_k\}$.

The node mobility entails time-varying AC weights $\mathbf{W}_k$. Note that in contrast to time-varying AC weights in static WSN, the weight matrices $\mathbf{W}_k$ here can have different zero patterns since the adjacency matrix of the underlying graph changes over time. The AC update equation thus reads

$$\mathbf{x}[k+1] = \mathbf{W}_{k+1}\,\mathbf{x}[k]\,. \tag{5.1}$$

Combining all state updates since the initial measurements yields (here $\mathbf{s} = (s_1\, s_2\, \ldots\, s_I)^T$)

$$\mathbf{x}[k] = \mathbf{W}_{k\to 1}\mathbf{s}, \qquad \text{with } \mathbf{W}_{k\to l} = \mathbf{W}_k\mathbf{W}_{k-1}\cdots\mathbf{W}_l.$$

In order for (5.1) to converge to $\bar{s}\mathbf{1} = \mathbf{J}\mathbf{s}$ (with $\mathbf{J} = \frac{1}{I}\mathbf{1}\mathbf{1}^T$), it is sufficient that each individual weight matrix $\mathbf{W}_k$ satisfies the convergence conditions of AC, i.e., $\mathbf{W}_k\mathbf{1} = \mathbf{1}$ and $\rho\{\mathbf{W}_k - \mathbf{J}\} < 1$. For MH weights, we will stick to this sufficient condition, since it allows us to use the MH method in each time instant to design the weights $\left(\mathbf{W}_k\right)_{ij}$ based on the current node degrees and thereby guarantee convergence. Note that such a design only requires to locally track the node degrees and hence can easily be implemented in a distributed manner. For CW weights, we can either use the same approach to determine the weights in each iteration, or use the statistical property of the evolving graph to compute the maximum degree in expectation $\mathrm{E}\{\max\{\mathcal{N}_k(.)\}\}$ which is constant over time and invert it to obtain a constant weight similar to the static case. Stability is guaranteed due to [5].

In the following, we aim at quantifying the performance gains achieved by node mobility.

5.1.1 Definition of the MSE

As a performance metric, we use the per-node MSE in iteration k, defined as

$$\bar{\epsilon}^2[k] \triangleq \frac{1}{I}\,\mathrm{E}_{\mathcal{T}}\{\epsilon^2[k]\} \quad \text{with } \epsilon^2[k] \triangleq \mathrm{E}_{\mathbf{s}|\mathcal{T}}\big\{\|\mathbf{x}[k] - \bar{s}\mathbf{1}\|^2\big\}. \tag{5.2}$$

Here, $\mathrm{E}_{\mathcal{T}}$ and $\mathrm{E}_{\mathbf{s}|\mathcal{T}}$ denote expectation with respect to the sequence $\mathcal{T}$ of graph topologies (which determines the time-varying AC weights) and conditional expectation with respect to the measurements s given $\mathcal{T}$. We define $P_{\bar{s}} = \mathrm{E}_{\mathbf{s}|\mathcal{T}}\{\bar{s}^2\} = \frac{1}{I^2}\mathbf{1}^T\mathbf{R}_s\mathbf{1}$ where $\mathbf{R}_s = \mathrm{E}_{\mathbf{s}|\mathcal{T}}\{\mathbf{s}\mathbf{s}^T\}$ denotes the measurement correlation matrix.

We develop the MSE by noting that $\bar{s}\mathbf{1} = \mathbf{J}\mathbf{s}$ and hence,

$$\epsilon^2[k] = \mathrm{E}_{\mathbf{s}|\mathcal{T}}\big\{\|(\mathbf{W}_{k\to 1} - \mathbf{J})\mathbf{s}\|^2\big\} = \mathrm{tr}\big\{\overline{\mathbf{W}}_{k\to 1}\mathbf{R}_s\overline{\mathbf{W}}_{k\to 1}^T\big\}, \tag{5.3}$$

where $\overline{\mathbf{W}}_{k\to l} = \mathbf{W}_{k\to l} - \mathbf{J}$. Using the fact that $\mathbf{W}_{k\to l}\mathbf{J} = \mathbf{J}$ and $\mathbf{R}_{\mathbf{x}[l-1]} = \mathbf{W}_{(l-1)\to 1}\mathbf{R}_s\mathbf{W}_{(l-1)\to 1}^T$, it can be shown that $\overline{\mathbf{W}}_{k\to 1} = \overline{\mathbf{W}}_{k\to l}\overline{\mathbf{W}}_{(l-1)\to 1}$ and $\overline{\mathbf{W}}_{(l-1)\to 1}\mathbf{R}_s\overline{\mathbf{W}}_{(l-1)\to 1}^T = \overline{\mathbf{R}}_{\mathbf{x}[l-1]} \triangleq (\mathbf{I} - \mathbf{J})\mathbf{R}_{\mathbf{x}[l-1]}(\mathbf{I} - \mathbf{J})$. We therefore obtain

$$\epsilon^2[k] = \mathrm{tr}\left\{\overline{\mathbf{W}}_{k\to l}\overline{\mathbf{R}}_{x[l-1]}\overline{\mathbf{W}}_{k\to l}^T\right\},$$

where $\epsilon^2[l] = \operatorname{tr}\left\{\overline{\mathbf{R}}_{x[l-1]}\right\}$, i.e., the MSE is given through the trace of the modified correlation matrix (similar to Section 4.2.1). It is obvious that we obtain the largest initial MSE for a fixed power of the measurements if the correlation matrix represents an i.i.d. setting, which means that $\mathbf{R}_\mathbf{s}$ is a scaled version of the identity matrix. If the measurements are correlated, it is as if AC was processed already such that the correlation is generated accordingly.

5.1.2 Upper Bound on the MSE

For our investigations of the upper bound we investigate initial i.i.d. sensor measurements. Considering a correlated setting is not trivial, since it is hard to relate the correlation of the measurements with the weight matrix and especially with its expectation. Therefore, we assume an initial correlation of $\mathbf{R}_\mathbf{s} = c\mathbf{I}$ where c is an arbitrary constant. The corresponding initial MSE is $\epsilon^2[0] = c\frac{I-1}{I}$. An upper bound on the MSE is stated in the following theorem.

Theorem 4. *For initially i.i.d. measurements ($\mathbf{R}_\mathbf{s} = c\mathbf{I}$ with arbitrary c), the MSE of AC with given weight design scheme does not degrade for stationary Markovian evolving graphs compared to a static setting, i.e.,*

$$\bar{\epsilon}^2[2] \leq \bar{\epsilon}^2_{\mathcal{E}}[2]\,,$$

where the subscript $\mathcal{E}$ indicates that the weight matrix remains constant.

Proof. According to (5.3) we have

$$\epsilon^2[2] = c\operatorname{tr}\left\{\overline{\mathbf{W}}_{2\to1}\overline{\mathbf{W}}_{2\to1}^T\right\} = c\operatorname{tr}\left\{\overline{\mathbf{W}}_1^2\overline{\mathbf{W}}_2^2\right\} \leq \sqrt{c\operatorname{tr}\left\{\overline{\mathbf{W}}_1^4\right\}c\operatorname{tr}\left\{\overline{\mathbf{W}}_2^4\right\}}\,.$$

Since we know that $c\operatorname{tr}\left\{\overline{\mathbf{W}}_1^4\right\}$ is the MSE after two iterations for a static setting with the weight matrix $\mathbf{W}_1$ we can conclude that the MSE for a mobile setting $\epsilon^2[2]$ is always less or equal to the geometric mean of the two static instances of AC with weights $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively. We can upper bound the geometric mean in terms of the arithmetic mean, i.e.,

$$\epsilon^2[2] \leq \frac{1}{2}\left(c\operatorname{tr}\left\{\overline{\mathbf{W}}_1^4\right\} + c\operatorname{tr}\left\{\overline{\mathbf{W}}_2^4\right\}\right).$$

Taking the expectation regarding the topologies we obtain

$$\bar{\epsilon}^2[2] \leq \bar{\epsilon}^2_{\mathcal{E}}[2]\,,$$

since we have

$$\bar{\epsilon}^2_{\mathcal{E}}[2] = \frac{1}{I}\operatorname{E}_{\mathcal{T}}\left\{c\operatorname{tr}\left\{\overline{\mathbf{W}}_1^4\right\}\right\} = \frac{1}{I}\operatorname{E}_{\mathcal{T}}\left\{c\operatorname{tr}\left\{\overline{\mathbf{W}}_2^4\right\}\right\},$$

where equality follows from the stationarity of the graph structure, thereby concluding our proof.

□

Discussion. As we see in the proof, in the case when we have two different weight matrices which are both applied in the same scenario, we always obtain a performance which is better than the worse of the two individually. Hence, in the deterministic sense there are settings where a good choice of the weight matrix (in particular in the sense of the graph topology) may lead to a faster convergence than a mobile setting. In expectation, however, we have proved that mobility will always be superior, or at least yield the same performance, if the statistics of the graph topology is stationary.

For non-i.i.d. correlation we have no such result, but since the trace is an inner product we can relate the inner product for two weight matrices and a given correlation as $\langle \overline{\mathbf{W}}_1^2, \overline{\mathbf{R}}_{\mathbf{x}[k]} \rangle \leq \langle \overline{\mathbf{W}}_2^2, \overline{\mathbf{R}}_{\mathbf{x}[k]} \rangle$, when the MSE of AC after applying $\mathbf{W}_1$ is smaller or equal than for $\mathbf{W}_2$. Hence, if the similarity between $\overline{\mathbf{W}}_1$ and the correlation is smaller in the inner product sense, we obtain a better MSE. It is somehow intuitive, but it was not proven so far, that if for instance $\mathbf{W}_2$ was already used in the sequence of obtaining $\overline{\mathbf{R}}_{\mathbf{x}[k]}$ we obtain a larger inner product in most cases than for a matrix $\mathbf{W}_1$. The key problem for further analytical investigation is that there exist cases which do not follow this rule and they cannot be easily identified in full generality.

5.1.3 Lower Bound on the MSE

For the lower bound we provide analytical results for the performance gain of mobile settings. The main result is given in the following theorem, where we use $\omega_k = \|\mathbf{W}_{2k}\mathbf{W}_{2k-1}\|_F^2$.

Theorem 5. *Consider AC on a stationary Markovian evolving graph and assume that $P_{\bar{s}}$ does not depend on $\mathcal{T}$, that $\mathbf{R}_s$ has full rank, and that $E_{\mathcal{T}}\{\omega_k\omega_{k-1}\} \geq E_{\mathcal{T}}\{\omega_k\}E_{\mathcal{T}}\{\omega_{k-1}\}$. Then, the MSE is lower bounded as*

$$\bar{\epsilon}^2[k] \geq (I-1)P_{\bar{s}}\left[\frac{E_{\mathcal{T}}\{\omega_1\}-1}{I-1}\right]^{\lceil \frac{k}{2} \rceil}. \tag{5.4}$$

Discussion. The assumption that $P_{\bar{s}}$ is independent of $\mathcal{T}$ effectively implies an i.i.d. sensor node placement. Furthermore, the condition $\mathrm{E}_{\mathcal{T}}\{\omega_k\omega_{k-1}\} \geq \mathrm{E}_{\mathcal{T}}\{\omega_k\}\mathrm{E}_{\mathcal{T}}\{\omega_{k-1}\}$ essentially excludes evolving graphs that oscillate between being strongly and weakly connected. It is seen that the MSE lower bound decays exponentially, with an MSE improvement by a factor of $\frac{\mathrm{E}_{\mathcal{T}}\{\omega_1\}-1}{I-1}$ every second iteration. Note that $\mathrm{E}_{\mathcal{T}}\{\omega_1\}$ describes how on average the connectivity of the evolving graph changes from one time instant to the next (see Section 5.1.4).

Proof. Exploiting the fact that the null space of both $\overline{\mathbf{W}}_{k\to l}$ and $\overline{\mathbf{R}}_{x[l-1]}$ is spanned by $\mathbf{1}$ and denoting the smallest non-zero eigenvalue of $\overline{\mathbf{R}}_{x[l-1]}$ by $\lambda_{\min}$, it can be shown that

$$\operatorname{tr}\left\{\overline{\mathbf{W}}_{k\to l}\overline{\mathbf{R}}_{x[l-1]}\overline{\mathbf{W}}_{k\to l}^T\right\} \geq \lambda_{\min}\left\|\overline{\mathbf{W}}_{k\to l}\right\|_F^2. \tag{5.5}$$

This bound is tight for the case where $\overline{\mathbf{R}}_{x[l-1]} = \lambda_{\min}(\mathbf{I}-\mathbf{J})$. With the constraint that the trace of $\mathbf{R}_{x[l-1]}$ is fixed, it follows that

$$\begin{aligned}\operatorname{tr}\{\mathbf{R}_{x[l-1]}\} &= \operatorname{tr}\{\overline{\mathbf{R}}_{x[l-1]}\} + \operatorname{tr}\{\mathbf{J}\mathbf{R}_{x[l-1]}\mathbf{J}\}\\ &= \lambda_{\min}(I-1) + \operatorname{tr}\{\mathbf{J}\mathbf{R}_s\mathbf{J}\},\end{aligned}$$

which further implies

$$\lambda_{\min} = \frac{\operatorname{tr}\{\mathbf{R}_{x[l-1]} - \mathbf{J}\mathbf{R}_s\mathbf{J}\}}{I-1} = \frac{\operatorname{tr}\{\overline{\mathbf{W}}_{(l-1)\to 1}\mathbf{R}_{\mathbf{s}}\overline{\mathbf{W}}_{(l-1)\to 1}\}}{I-1}.$$

Inserting this result into (5.5) and comparing with (5.3), we obtain

$$\epsilon^2[k] \geq \epsilon^2[l-1]\,\frac{\|\mathbf{W}_{k\to l}\|_F^2 - 1}{I-1}, \tag{5.6}$$

where we further used $\|\overline{\mathbf{W}}_{k\to l}\|_F^2 = \|\mathbf{W}_{k\to l}\|_F^2 - 1$. Assuming k even, applying this bound recursively with $l = k-1$, $l = k-3$, etc, and using $\epsilon^2[0] \geq P_{\bar{s}} I(I-1)$ with $P_{\bar{s}} = \mathrm{E}\{\bar{s}^2\} = \frac{1}{I^2}\mathbf{1}^T\mathbf{R}_s\mathbf{1}$, we have

$$\epsilon^2[k] \geq I(I-1)P_{\bar{s}} \prod_{l=1}^{k/2} \frac{\omega_l - 1}{I-1}. \tag{5.7}$$

Note that the iterated application of (5.6) implies that the bound (5.7) becomes less and less tight as k increases. For odd iteration indices $k = 2l-1$, we simply use the fact that $\epsilon^2[2l-1] \geq \epsilon^2[2l]$. Note that the combination of two iterations is important to capture mobility effects. The final result (5.4) is obtained by inserting (5.7) into (5.2), using the independence of $P_{\bar{s}}$ and $\mathcal{T}$, and noting that

$$\mathrm{E}_{\mathcal{T}}\left\{\prod_{l=1}^{k/2} \frac{\omega_l - 1}{I-1}\right\} \geq \prod_{l=1}^{k/2} \mathrm{E}_{\mathcal{T}}\left\{\frac{\omega_l - 1}{I-1}\right\} = \left[\frac{\mathrm{E}_{\mathcal{T}}\{\omega_1\} - 1}{I-1}\right]^{\frac{k}{2}}.$$

Here, we used the assumption $\mathrm{E}_{\mathcal{T}}\{\omega_k\omega_{k-1}\} \geq \mathrm{E}_{\mathcal{T}}\{\omega_k\}\mathrm{E}_{\mathcal{T}}\{\omega_{k-1}\}$ and exploited the fact that for stationary Markovian evolving graphs the sequence ω_l is stationary Markovian, too. ☐

5.1.4 Mean Frobenius Norm

The key quantity in (5.4) is the mean Frobenius norm of $\mathbf{G} \triangleq \mathbf{W}_2\mathbf{W}_1$,

$$\mathrm{E}_{\mathcal{T}}\{\omega_1\} = \sum_{i,j} \mathrm{E}_{\mathcal{T}}\{g_{ij}^2\}, \tag{5.8}$$

where

$$g_{ij} = (\mathbf{G})_{ij} = \sum_{l \in \tilde{\mathcal{N}}_2(i) \cap \tilde{\mathcal{N}}_1(j)} (\mathbf{W}_2)_{il}(\mathbf{W}_1)_{lj}.$$

Here, $\tilde{\mathcal{N}}_k(i) = \{i\} \cup \mathcal{N}_k(i)$. Hence, g_{ij} involves node j at time $k = 1$, node i at time $k = 2$, and the nodes that are neighbors of node j at time $k = 1$ and of node i at time $k = 2$. Note that it is possible that $g_{ij} \neq 0$ even if there is no path between nodes i and j at any given time. This is in contrast to the static case and explains in part why node mobility can be beneficial for distributed averaging. We assume that not necessarily all nodes are moving; the set of moving nodes is defined as $\mathcal{V}_\mathrm{m} \subseteq \mathcal{V}$ and the number of moving nodes is denoted by $I_\mathrm{m} = |\mathcal{V}_\mathrm{m}|$. The complement, i.e., the number of static nodes is denoted as $\bar{I}_\mathrm{m} = I - I_\mathrm{m}$.

We can then distinguish six disjoint types of expectations. For the diagonal elements we have: (i) $\mathrm{E}_{\mathcal{T}}\{|g_{ii}|^2\}$ if node i is moving and (ii) $\mathrm{E}_{\mathcal{T}}\{|g_{ii}|^2\}$ if node i is not moving. The expectation $\mathrm{E}_{\mathcal{T}}\{|g_{ij}|^2\}$ of the off-diagonal elements are completely represented by: (iii) if node i is moving and j is static, (iv) if j is moving and i not, (v) assumes that i and j are both moving, and finally (vi) considers both nodes not moving. Under the assumption of a spatially homogenous node placement, these expectations are independent of the node indices and hence we can collect equivalent terms to express the expected Frobenius norm of $\mathbf{G}$ as

$$\mathrm{E}_{\mathcal{T}}\{\omega_1\} = \bar{I}_\mathrm{m}\mathrm{E}_{\mathcal{T}}\{g_{ii}^2 | i \notin \mathcal{V}_\mathrm{m}\} + I_\mathrm{m}\mathrm{E}_{\mathcal{T}}\{g_{ii}^2 | i \in \mathcal{V}_\mathrm{m}\}$$

$$\begin{aligned}
&+ I_{\mathrm{m}}\bar{I}_{\mathrm{m}}\mathrm{E}_{\mathcal{T}}\big\{g_{ij}^2\big|i \in \mathcal{V}_{\mathrm{m}}, j \notin \mathcal{V}_{\mathrm{m}}\big\}\\
&+ I_{\mathrm{m}}\bar{I}_{\mathrm{m}}\mathrm{E}_{\mathcal{T}}\big\{g_{ij}^2\big|i \notin \mathcal{V}_{\mathrm{m}}, j \in \mathcal{V}_{\mathrm{m}}\big\}\\
&+ I_{\mathrm{m}}(I_{\mathrm{m}}-1)\mathrm{E}_{\mathcal{T}}\big\{g_{ij}^2\big|i \in \mathcal{V}_{\mathrm{m}}, j \in \mathcal{V}_{\mathrm{m}}\big\}\\
&+ \bar{I}_{\mathrm{m}}(\bar{I}_{\mathrm{m}}-1)\mathrm{E}_{\mathcal{T}}\big\{g_{ij}^2\big|i, j \notin \mathcal{V}_{\mathrm{m}}\big\} \;. \qquad (5.9)
\end{aligned}$$

The derivation of $\mathrm{E}_{\mathcal{T}}\big\{g_{ii}^2\big|\cdot\big\}$ and $\mathrm{E}_{\mathcal{T}}\big\{g_{ij}^2\big|\cdot\big\}$ for a CW design are shown in Appendix 5.A and the results depend on seven probabilities which model the graph and the motion model:

$$\begin{aligned}
\underline{p}_1^2 &= \mathrm{P}\left\{k \in \mathcal{N}_1(i) \cap \mathcal{N}_2(i)\right\} ,\\
\underline{p}_2^2 &= \mathrm{P}\left\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_2(j)\right\} ,\\
\underline{p}_3^3 &= \mathrm{P}\left\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_1(l), j \in \mathcal{N}_2(l)\right\} ,\\
\underline{p}_4^3 &= \mathrm{P}\left\{j \in \mathcal{N}_1(i) \cap \mathcal{N}_1(l), i \in \mathcal{N}_2(l)\right\} ,\\
\underline{p}_5^4 &= \mathrm{P}\left\{i \in \mathcal{N}_1(n) \cap \mathcal{N}_1(l), i \in \mathcal{N}_2(n) \cap \mathcal{N}_2(l)\right\} ,\\
\underline{p}_6^4 &= \mathrm{P}\left\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_1(l) \cap \mathcal{N}_2(j), j \in \mathcal{N}_2(l)\right\} ,\\
\underline{p}_7^4 &= \mathrm{P}\left\{k \in \mathcal{N}_1(i) \cap \mathcal{N}_2(j), l \in \mathcal{N}_1(i) \cap \mathcal{N}_2(j)\right\} ,
\end{aligned}$$

where k, l, and n denote nodes of the graph and the superscripts of the probabilities represent the number of node dependencies. Finally, to obtain a closed-form expression for $\mathrm{E}_{\mathcal{T}}\{\omega_1\}$ it is necessary to determine these probabilities for the corresponding model.

5.1.5 Static WSN

In what follows, we restrict our attention to the case of random geometric graphs on the surface of a torus $\mathcal{A}$ (equivalent to a periodically extended rectangular region). The I nodes initially have an i.i.d. uniform distribution. We first consider the static scenario (i.e., $\mathbf{W}_1 = \mathbf{W}_2$), which serves as a basis for the case with node mobility. Here,

$$\mathrm{E}_{\mathcal{T}}\{\omega_1\} = I\,\mathrm{E}_{\mathcal{T}}\big\{g_{ii}^2\big\} + (I-1)I\,\mathrm{E}_{\mathcal{T}}\big\{g_{ij}^2\big|i \neq j\big\} \,.$$

For a constant weight design, the computation of the seven probabilities to derive $\mathrm{E}_{\mathcal{T}}\big\{g_{ij}^2\big\}$ simplifies, since there is no difference between $\mathcal{N}_1(\cdot)$ and $\mathcal{N}_2(\cdot)$. More specifically, there is the pairwise neighborhood probability

$$p = \mathrm{P}\{j \in \mathcal{N}(i)\} = \frac{r^2\pi}{|\mathcal{A}|} \,,$$

which equals the ratio of the area within which a node i can communicate and the overall WSN area. It is obvious that we have $\underline{p}_1^2 = \underline{p}_2^2 = p$ and $\underline{p}_5^4 = p^2$ for a static graph setting.

Furthermore, $\underline{p}_3^3$, $\underline{p}_4^2$, and $\underline{p}_6^4$ simplifies to the triple-neighborhood probability, i.e., the probability that three nodes are mutual neighbors. The underlying geometry is illustrated in Fig. 5.1. Here, nodes i and j have to be neighbors, i.e. $r_{ij} \leq r$ (where r_{ij} is the distance between nodes i and j); furthermore, node l has to lie in the intersection of the communication range of nodes i and j, denoted $\mathcal{A}_{ij}$, which happens with probability $\frac{|\mathcal{A}_{ij}|}{|\mathcal{A}|}$.

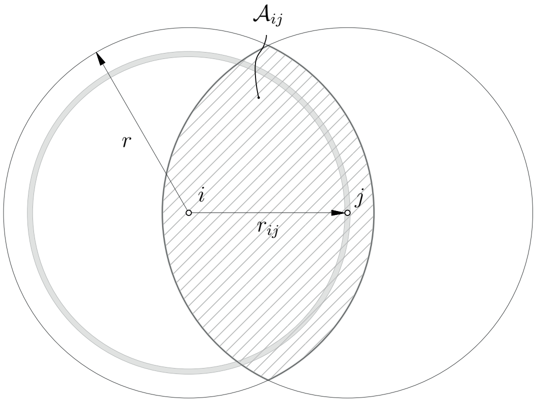


Figure 5.1: *Geometry of triple neighborhoods.*

The area of $\mathcal{A}_{ij}$ can be computed using standard techniques, leading to

$$\underline{p}_2^3 = \mathrm{P}\{j \in \mathcal{N}(i),\, l \in \mathcal{N}(i) \cap \mathcal{N}(j)\} = p^2 \left(1 - \frac{3\sqrt{3}}{4\pi}\right). \tag{5.10}$$

Finally, $\underline{p}_7^4$ boils down to the probability for the existence of a four-hop loop, which can be derived using similar considerations as above. The differences are, that k and l have to lie in the intersection $\mathcal{A}_{ij}$ and that the nodes i and j have to lie with the distance of $2r$, because they do not need to be neighbors. To this end we obtain

$$\underline{p}_3^4 = \mathrm{P}\{k \in \mathcal{N}(i) \cap \mathcal{N}(j), l \in \mathcal{N}(i) \cap \mathcal{N}(j)\} = p^3 \left(1 - \frac{16}{3\pi^2}\right).$$

5.1.6 Mobile WSN

We next consider mobile WSN in which the movements of different nodes are statistically independent and stationary, thus leading to a stationary geometric Markovian evolving graph [51] in which the node positions remain i.i.d. over time.

For simplicity we define our motion model as follows: I_m nodes are selected in a random geometric graph (not necessary the same at each iteration) and each of these nodes is placed on a new position randomly according to a uniform distribution within the considered area. Hence, the random geometric graph structure is preserved. The weights for AC are constant (CW) (see Section 2.2.2); we denote by w the expectation of the constant weight. We determine the seven probabilities defined in Section 5.1.4 in the following, and then the results are plugged into the results of Appendix 5.A to obtain the result for $\mathrm{E}_\mathcal{T}\{\omega_1\}$.

We have to note that besides the probabilities $\underline{p}_1$ to $\underline{p}_7$ illustrated in Fig. 5.2 - 5.8, we also provide the probabilities of the occurrence for the corresponding setting below each figure. Some of these probabilities are conditioned on a certain movement, which may not be feasible in some cases if $I_\mathrm{m} \in \{0, 1, I-1, I\}$. Implicitly these cases vanish due to the prior.

Diagonal elements. First of all we consider $\underline{p}_1$ and $\underline{p}_5$ which are necessary for the diagonal elements, i.e., $\mathrm{E}_\mathcal{T}\{g_{ij}^2\}$. All possible geometric relations are illustrated in Fig. 5.2 and 5.6. We start with computing the probabilities if node i is moving:

$$\underline{p}_1^2 = p^2\, \mathrm{P}\{k \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} + p^2\, \mathrm{P}\{k \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} = p^2\,,$$

$$
\begin{aligned}
\underline{p}_5^4 &= p_3^4\,\mathrm{P}\{k,l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} + p^4\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} \\
&\quad + p^4\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} + p^4\,\mathrm{P}\{k,l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} \\
&= p_3^4 \frac{(\bar{I}_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-1)(I-2)} + p^4\left(1 - \frac{\bar{I}_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-1)(I-2)}\right).
\end{aligned}
$$

In the case when node i is not moving we obtain:

$$
\begin{aligned}
\underline{p}_1^2 &= p\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} + p^2\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} = p\frac{\bar{I}_\mathrm{m}-1}{I-1} + p^2\frac{I_\mathrm{m}}{I-1}, \\
\underline{p}_5^4 &= p^2\,\mathrm{P}\{k,l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} \\
&\quad + p^3\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} + p^4\,\mathrm{P}\{k,l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} \\
&= p^2\frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-1)(I-2)} + p^3\frac{2I_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-1)(I-2)} + p^4\frac{I_\mathrm{m}(I_\mathrm{m}-1)}{(I-1)(I-2)}.
\end{aligned}
$$

Off-diagonal elements. In the following we determine the other probabilities which are necessary for the off-diagonal elements. All relations are illustrated in Fig. 5.3 - 5.5, 5.7, and 5.8. Since we consider $\mathrm{E}_\mathcal{T}\{g_{ij}^2\}$ where $i \neq j$ we have to distinguish between four different cases (all possible combinations of i and j moving or not). We start with i and j being static:

$$
\begin{aligned}
\underline{p}_2^2 &= p, \\
\underline{p}_3^3 &= p_2^3\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} \\
&= p_2^3\frac{\bar{I}_\mathrm{m}-2}{I-2} + p^3\frac{I_\mathrm{m}}{I-2}, \\
\underline{p}_4^3 &= \underline{p}_3, \\
\underline{p}_6^4 &= p_2^3\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} \\
&= p_2^3\frac{\bar{I}_\mathrm{m}-2}{I-2} + p^3\frac{I_\mathrm{m}}{I-2}, \\
\underline{p}_7^4 &= p_3^4\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} \\
&\quad + p^3\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} \\
&= p_3^4\frac{(\bar{I}_\mathrm{m}-2)(\bar{I}_\mathrm{m}-3)}{(I-2)(I-3)} + p^4\left(1 - \frac{(\bar{I}_\mathrm{m}-2)(\bar{I}_\mathrm{m}-3)}{(I-2)(I-3)}\right).
\end{aligned}
$$

The next probabilities model the case when i is moving and j is static:

$$
\begin{aligned}
\underline{p}_2^2 &= p^2, \\
\underline{p}_3^3 &= p_2^3\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} \\
&= p_2^3\frac{\bar{I}_\mathrm{m}-1}{I-2} + p^3\frac{I_\mathrm{m}-1}{I-2}, \\
\underline{p}_4^3 &= p^3\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = p^3, \\
\underline{p}_6^4 &= p_2^3 p\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^4\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} \\
&= p_2^3 p\frac{\bar{I}_\mathrm{m}-1}{I-2} + p^4\frac{I_\mathrm{m}-1}{I-2}, \\
\underline{p}_7^4 &= p_3^4\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\}
\end{aligned}
$$

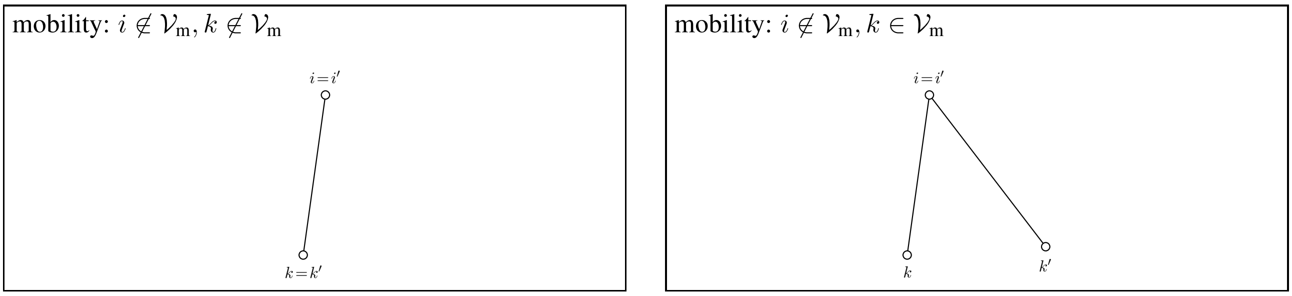
$$
\begin{aligned}
&\quad + p^3\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} \\
&= p_3^4 \frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)} + p^4\left(1 - \frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)}\right) .
\end{aligned}
$$

Furthermore we consider node i being static and j moving and obtain:

$$
\begin{aligned}
\underline{p}_2^2 &= p^2 \,, \\
\underline{p}_3^3 &= p^3\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = p^3 \,, \\
\underline{p}_4^3 &= p_2^3\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} \\
&= p_2^3 \frac{\bar{I}_\mathrm{m}-1}{I-2} + p^3 \frac{I_\mathrm{m}-1}{I-2} \,, \\
\underline{p}_6^4 &= p_2^3 p\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^4\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} \\
&= p_2^3 p \frac{\bar{I}_\mathrm{m}-1}{I-2} + p^4 \frac{I_\mathrm{m}-1}{I-2} \,, \\
\underline{p}_7^4 &= p_3^4\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} \\
&\quad + p^3\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} \,, \\
&= p_3^4 \frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)} + p^4\left(1 - \frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)}\right) .
\end{aligned}
$$

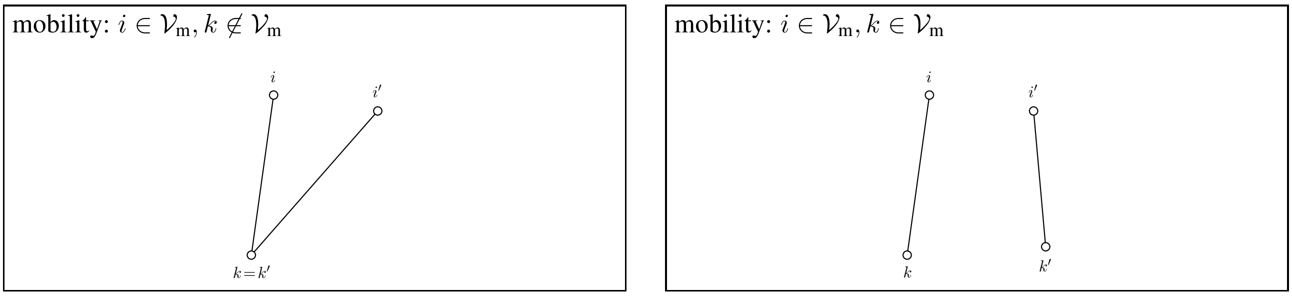
Finally the probabilities of the case when node i and j are moving are given by:

$$
\begin{aligned}
\underline{p}_2^2 &= p^2 \,, \\
\underline{p}_3^3 &= p^3\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = p^3 \,, \\
\underline{p}_4^3 &= \underline{p}_3 \,, \\
\underline{p}_6^4 &= p^4\,\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^4\,\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = p^4 \,, \\
\underline{p}_7^4 &= p_3^4\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} \\
&\quad + p^3\,\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} + p^3\,\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} \,, \\
&= p_3^4 \frac{(\bar{I}_\mathrm{m})(\bar{I}_\mathrm{m}-1)}{(I-2)(I-3)} + p^4\left(1 - \frac{\bar{I}_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-2)(I-3)}\right) .
\end{aligned}
$$


mobility: $i \notin \mathcal{V}_\mathrm{m}, k \notin \mathcal{V}_\mathrm{m}$
$i=i'$
$k=k'$
mobility: $i \notin \mathcal{V}_\mathrm{m}, k \in \mathcal{V}_\mathrm{m}$
$i=i'$
k
k'

(a) $\underline{p}_1^2 = p$, $\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-1}{I-1}$

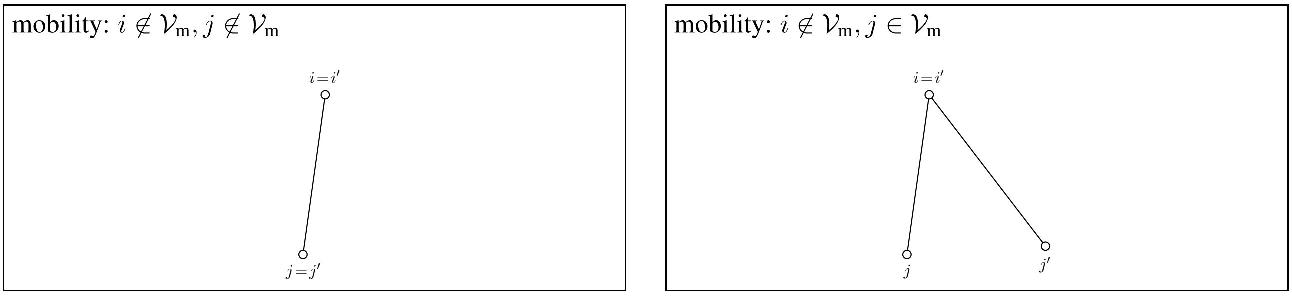
(b) $\underline{p}_1^2 = p^2$, $\mathrm{P}\{k \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}}{I-1}$


mobility: $i \in \mathcal{V}_\mathrm{m}, k \notin \mathcal{V}_\mathrm{m}$
i
i'
$k=k'$
mobility: $i \in \mathcal{V}_\mathrm{m}, k \in \mathcal{V}_\mathrm{m}$
i
i'
k
k'

(c) $\underline{p}_1^2 = p^2$, $\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}}{I-1}$

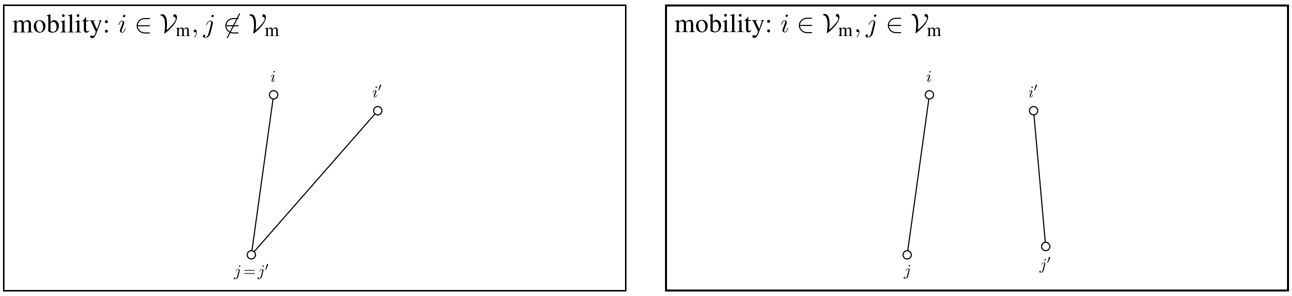
(d) $\underline{p}_1^2 = p^2$, $\mathrm{P}\{k \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-1}{I-1}$

Figure 5.2: *Probability* $\underline{p}_1^2 = \mathrm{P}\left\{k \in \mathcal{N}_1(i) \cap \mathcal{N}_2(i)\right\}$ *for all different mobility settings (a prime indicates a node after movement).*


mobility: $i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}$
$i=i'$
$j=j'$
mobility: $i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}$
$i=i'$
j
j'

(a) $\underline{p}_2^2 = p$

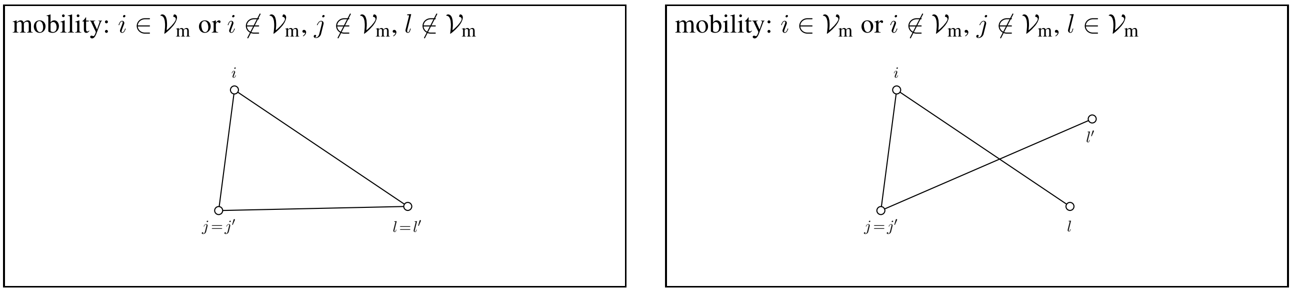
(b) $\underline{p}_2^2 = p^2$


mobility: $i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}$
i
i'
$j=j'$
mobility: $i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}$
i
i'
j
j'

(c) $\underline{p}_2^2 = p^2$

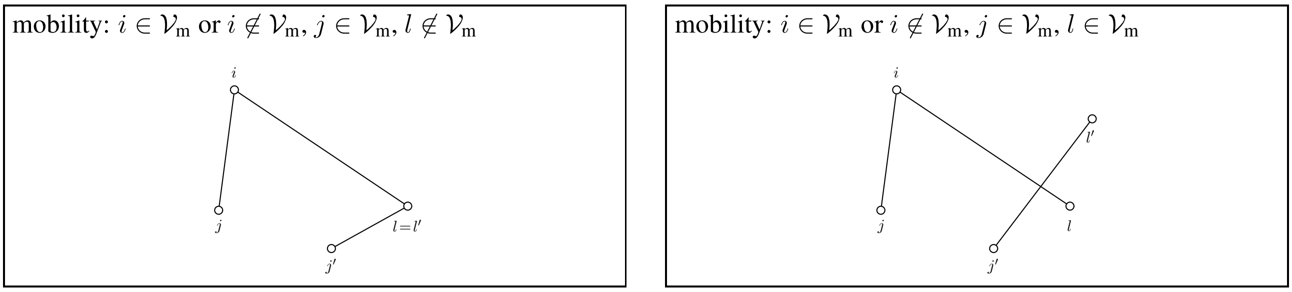
(d) $\underline{p}_2^2 = p^2$

Figure 5.3: *Probability* $\underline{p}_2^2 = \mathrm{P}\left\{j \in \mathcal{N}_1(i) \cap \mathcal{N}_2(i)\right\}$ *for all different mobility settings (a prime indicates a node after movement).*


mobility: $i \in \mathcal{V}_\mathrm{m}$ or $i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
i
$j=j'$
$l=l'$
mobility: $i \in \mathcal{V}_\mathrm{m}$ or $i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
i
l'
$j=j'$
l

(a) $\underline{p}_3^3 = p_3^2$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | j \notin \mathcal{V}_\mathrm{m}, i \notin \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-2}{I-2}$,
$\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | j \notin \mathcal{V}_\mathrm{m}, i \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-1}{I-2}$

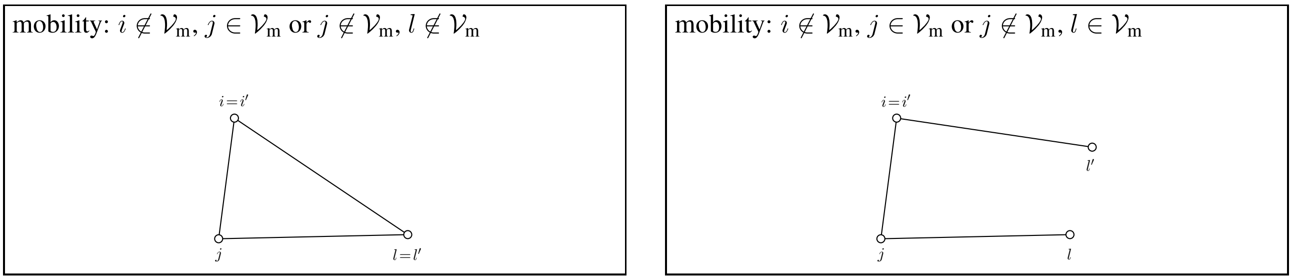
(b) $\underline{p}_3^3 = p^3$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | j \notin \mathcal{V}_\mathrm{m}, i \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}}{I-2}$,
$\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | j \notin \mathcal{V}_\mathrm{m}, i \in \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-1}{I-2}$


mobility: $i \in \mathcal{V}_\mathrm{m}$ or $i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
i
j
$l=l'$
j'
mobility: $i \in \mathcal{V}_\mathrm{m}$ or $i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
i
l'
j
l
j'

(c) $\underline{p}_3^3 = p^3$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | j \in \mathcal{V}_\mathrm{m}, i \notin \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-1}{I-2}$,
$\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | j \in \mathcal{V}_\mathrm{m}, i \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}}{I-2}$

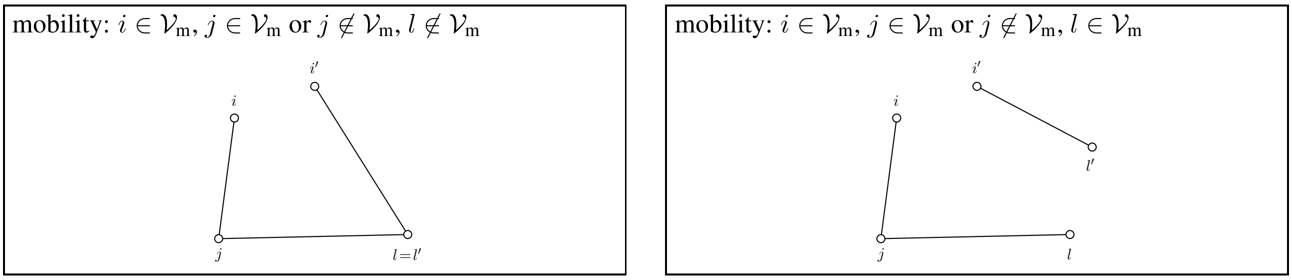
(d) $\underline{p}_3^3 = p^3$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | j \in \mathcal{V}_\mathrm{m}, i \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-1}{I-2}$,
$\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | j \in \mathcal{V}_\mathrm{m}, i \in \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-2}{I-2}$

Figure 5.4: *Probability* $\underline{p}_3^3 = \mathrm{P}\left\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_1(l), j \in \mathcal{N}_2(l)\right\}$ *for all different mobility settings (a prime indicates a node after movement).*


mobility: $i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
$i=i'$
j
$l=l'$
mobility: $i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
$i=i'$
l'
j
l

(a) $\underline{p}_4^3 = p_2^3$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-2}{I-2}$,
$\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-1}{I-2}$

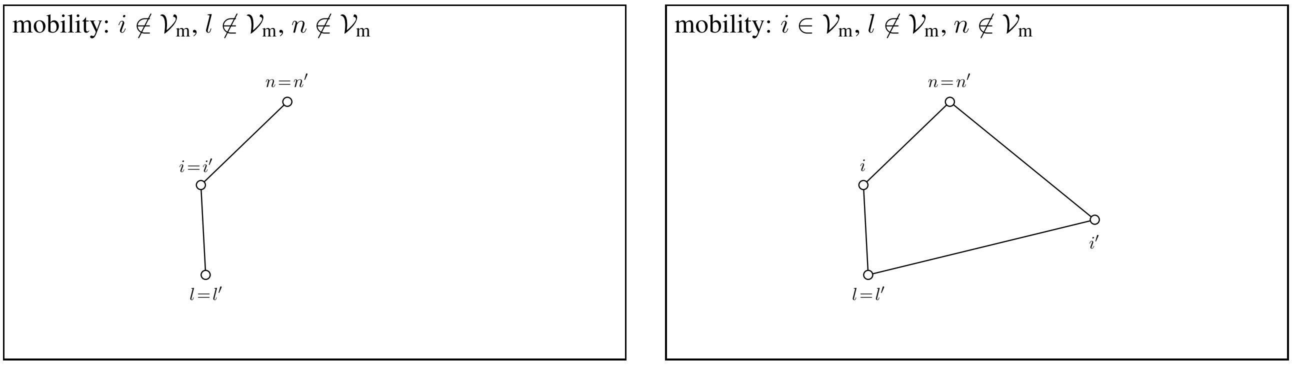
(b) $\underline{p}_4^3 = p^3$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}}{I-2}$,
$\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-1}{I-2}$


mobility: $i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
i'
i
j
$l=l'$
mobility: $i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
i'
i
l'
j
l

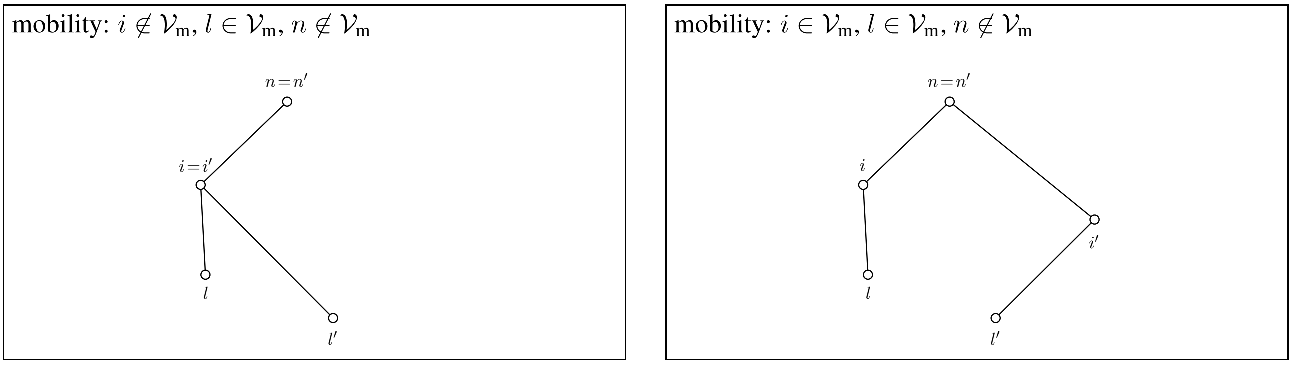
(c) $\underline{p}_4^3 = p^3$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-1}{I-2}$,
$\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}}{I-2}$

(d) $\underline{p}_4^3 = p^3$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-1}{I-2}$,
$\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-2}{I-2}$

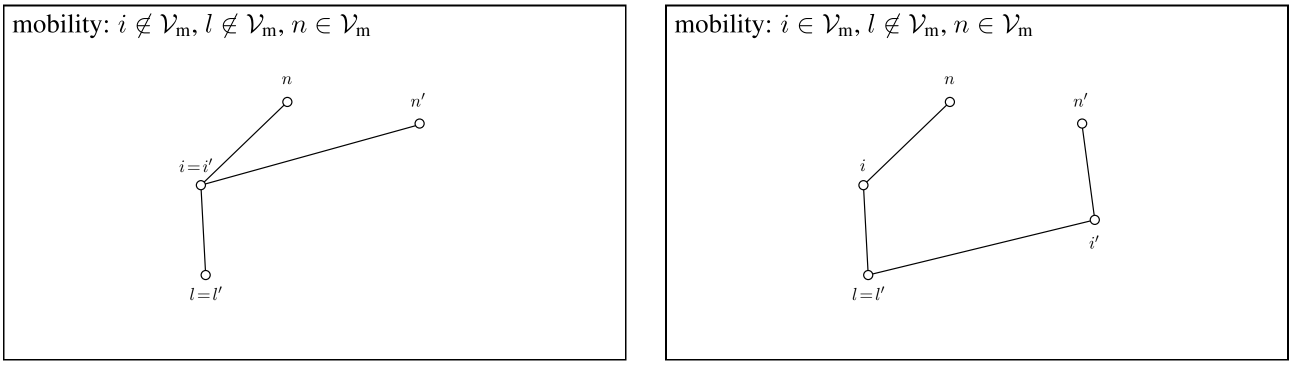
Figure 5.5: *Probability* $\underline{p}_4^3 = \mathrm{P}\left\{j \in \mathcal{N}_1(i) \cap \mathcal{N}_1(l), i \in \mathcal{N}_2(l)\right\}$ *for all different mobility settings (a prime indicates a node after movement).*


mobility: $i \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m}$
$n=n'$
$i=i'$
$l=l'$
mobility: $i \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m}$
$n=n'$
i
i'
$l=l'$

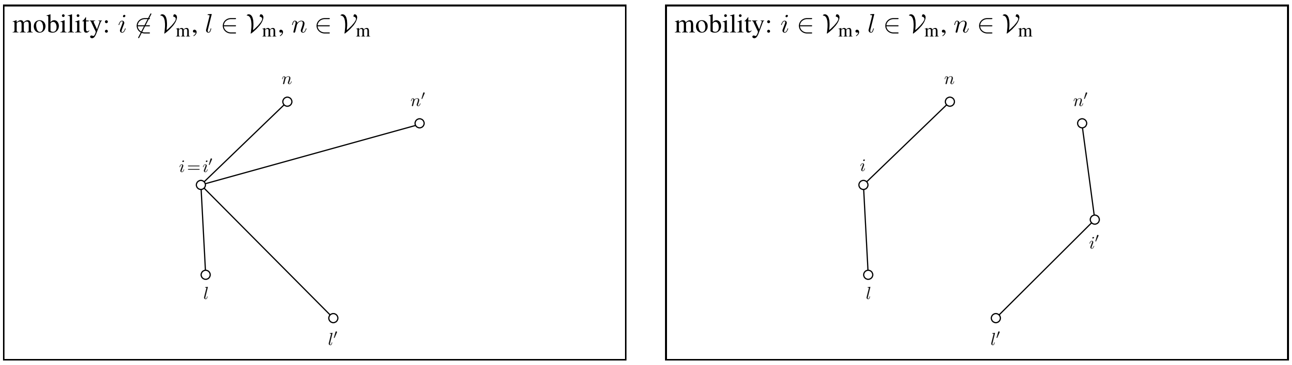
(a) $\underline{p}_5^2 = p^2, \mathrm{P}\{l \notin \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} = \frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-1)(I-2)}$ (b) $\underline{p}_5^2 = p_3^4, \mathrm{P}\{l \notin \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-1)(I-2)}$


mobility: $i \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m}$
$n=n'$
$i=i'$
l
l'
mobility: $i \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m}$
$n=n'$
i
i'
l
l'

(c) $\underline{p}_5^4 = p^3, \mathrm{P}\{l \in \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-1)(I-2)}$ (d) $\underline{p}_5^4 = p^4, \mathrm{P}\{l \in \mathcal{V}_\mathrm{m}, n \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)\bar{I}_\mathrm{m}}{(I-1)(I-2)}$

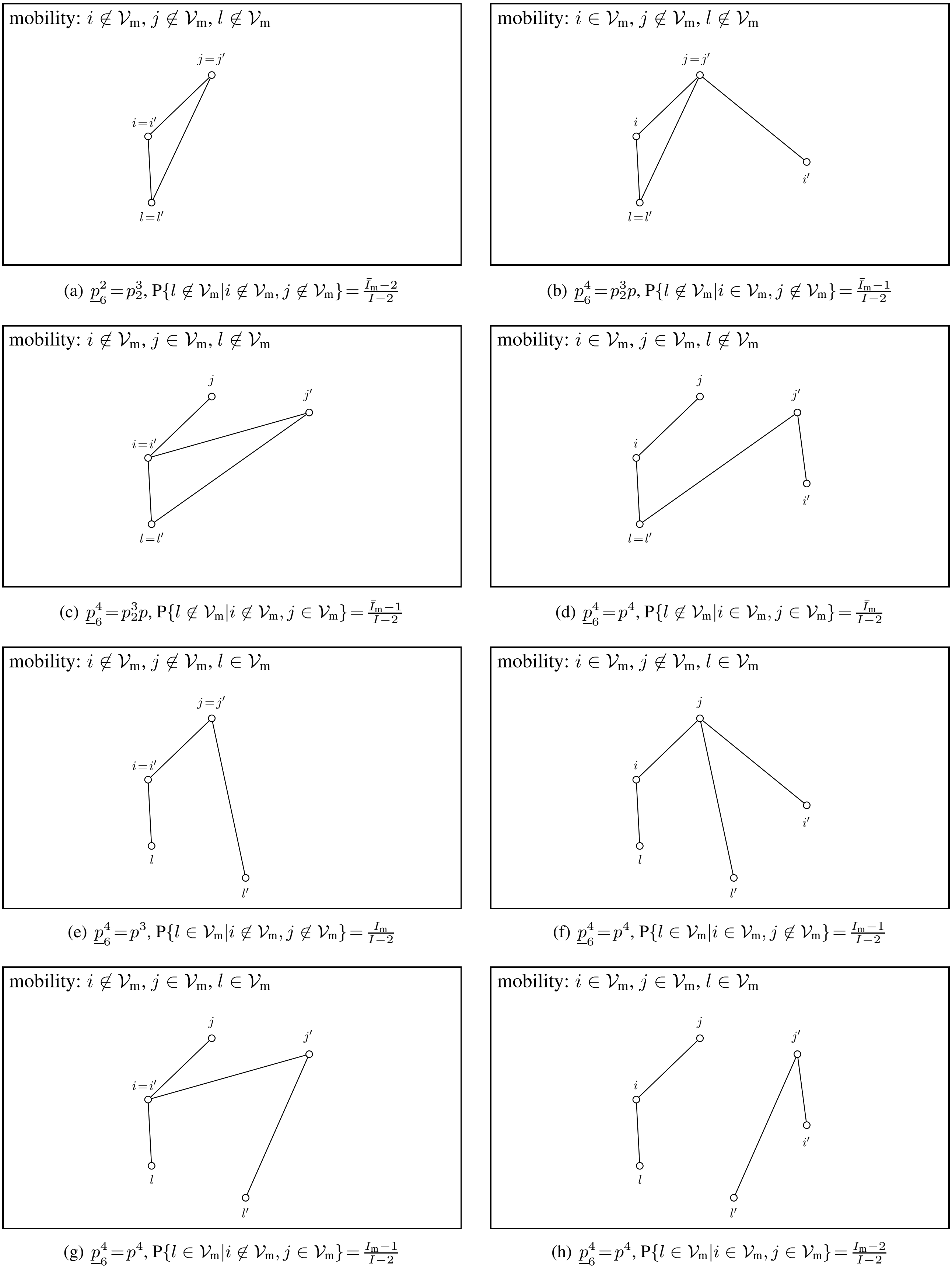

mobility: $i \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m}$
n
n'
$i=i'$
$l=l'$
mobility: $i \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m}$
n
n'
i
i'
$l=l'$

(e) $\underline{p}_5^4 = p^3, \mathrm{P}\{l \notin \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-1)(I-2)}$ (f) $\underline{p}_5^4 = p^4, \mathrm{P}\{l \notin \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)\bar{I}_\mathrm{m}}{(I-1)(I-2)}$


mobility: $i \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m}$
n
n'
$i=i'$
l
l'
mobility: $i \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m}$
n
n'
i
i'
l
l'

(g) $\underline{p}_5^4 = p^4, \mathrm{P}\{l \in \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}(I_\mathrm{m}-1)}{(I-1)(I-2)}$ (h) $\underline{p}_5^4 = p^4, \mathrm{P}\{l \in \mathcal{V}_\mathrm{m}, n \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)(I_\mathrm{m}-2)}{(I-1)(I-2)}$

Figure 5.6: *Probability* $\underline{p}_5^4 = \mathrm{P}\left\{i \in \mathcal{N}_1(n) \cap \mathcal{N}_1(l), i \in \mathcal{N}_2(n) \cap \mathcal{N}_2(l)\right\}$ *for all different mobility settings (a prime indicates a node after movement).*


mobility: $i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
$j=j'$
$i=i'$
$l=l'$
mobility: $i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
$j=j'$
i
i'
$l=l'$
mobility: $i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
j
j'
$i=i'$
$l=l'$
mobility: $i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m}$
j
j'
i
i'
$l=l'$
mobility: $i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
$j=j'$
$i=i'$
l
l'
mobility: $i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
j
i
i'
l
l'
mobility: $i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
j
j'
$i=i'$
l
l'
mobility: $i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m}$
j
j'
i
i'
l
l'

(a) $\underline{p}_6^2 = p_2^3$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-2}{I-2}$

(b) $\underline{p}_6^4 = p_2^3 p$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-1}{I-2}$

(c) $\underline{p}_6^4 = p_2^3 p$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}-1}{I-2}$

(d) $\underline{p}_6^4 = p^4$, $\mathrm{P}\{l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}}{I-2}$

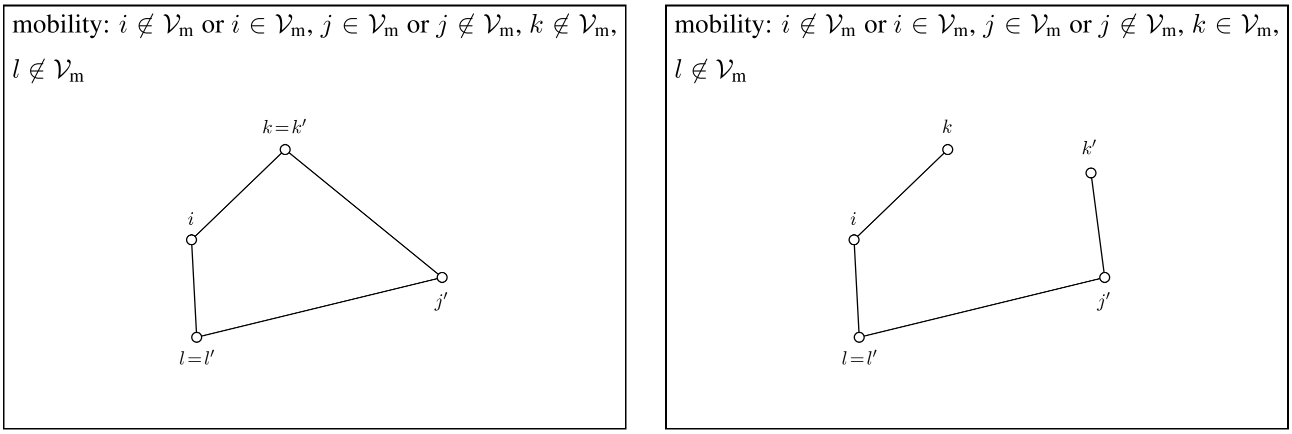
(e) $\underline{p}_6^4 = p^3$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}}{I-2}$

(f) $\underline{p}_6^4 = p^4$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-1}{I-2}$

(g) $\underline{p}_6^4 = p^4$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-1}{I-2}$

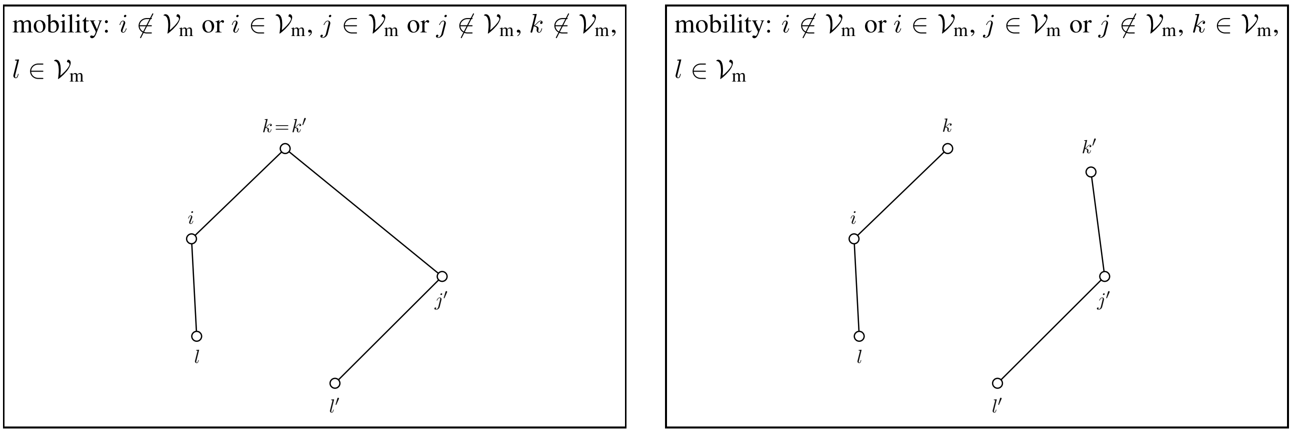
(h) $\underline{p}_6^4 = p^4$, $\mathrm{P}\{l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}-2}{I-2}$

Figure 5.7: *Probability* $\underline{p}_6^4 = \mathrm{P}\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_1(l) \cap \mathcal{N}_2(j), j \in \mathcal{N}_2(l)\}$ *for all different mobility settings (a prime indicates a node after movement).*


mobility: $i \notin \mathcal{V}_\mathrm{m}$ or $i \in \mathcal{V}_\mathrm{m}$, $j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}$, $k \notin \mathcal{V}_\mathrm{m}$, $l \notin \mathcal{V}_\mathrm{m}$
$k{=}k'$
i
j'
$l{=}l'$
mobility: $i \notin \mathcal{V}_\mathrm{m}$ or $i \in \mathcal{V}_\mathrm{m}$, $j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}$, $k \in \mathcal{V}_\mathrm{m}$, $l \notin \mathcal{V}_\mathrm{m}$
k
k'
i
j'
$l{=}l'$

(a) $\underline{p}_7^4 = p_3^4$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{(\bar{I}_\mathrm{m}-2)(\bar{I}_\mathrm{m}-3)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{(\bar{I}_\mathrm{m}-1)(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{\bar{I}_\mathrm{m}(\bar{I}_\mathrm{m}-1)}{(I-2)(I-3)}$

(b) $\underline{p}_7^4 = p^4$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)(\bar{I}_\mathrm{m}-1)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)(\bar{I}_\mathrm{m}-1)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \notin \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-2)\bar{I}_\mathrm{m}}{(I-2)(I-3)}$


mobility: $i \notin \mathcal{V}_\mathrm{m}$ or $i \in \mathcal{V}_\mathrm{m}$, $j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}$, $k \notin \mathcal{V}_\mathrm{m}$, $l \in \mathcal{V}_\mathrm{m}$
$k{=}k'$
i
j'
l
l'
mobility: $i \notin \mathcal{V}_\mathrm{m}$ or $i \in \mathcal{V}_\mathrm{m}$, $j \in \mathcal{V}_\mathrm{m}$ or $j \notin \mathcal{V}_\mathrm{m}$, $k \in \mathcal{V}_\mathrm{m}$, $l \in \mathcal{V}_\mathrm{m}$
k
k'
i
j'
l
l'

(c) $\underline{p}_7^4 = p^4$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}(\bar{I}_\mathrm{m}-2)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)(\bar{I}_\mathrm{m}-1)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)(\bar{I}_\mathrm{m}-1)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \notin \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-2)\bar{I}_\mathrm{m}}{(I-2)(I-3)}$

(d) $\underline{p}_7^4 = p^4$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{I_\mathrm{m}(I_\mathrm{m}-1)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \notin \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)(I_\mathrm{m}-2)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \notin \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-1)(I_\mathrm{m}-2)}{(I-2)(I-3)}$,
$\mathrm{P}\{k \in \mathcal{V}_\mathrm{m}, l \in \mathcal{V}_\mathrm{m} | i \in \mathcal{V}_\mathrm{m}, j \in \mathcal{V}_\mathrm{m}\} = \frac{(I_\mathrm{m}-2)(I_\mathrm{m}-3)}{(I-2)(I-3)}$

Figure 5.8: *Probability* $\underline{p}_7^4 = \mathrm{P}\left\{k \in \mathcal{N}_1(i) \cap \mathcal{N}_2(j), l \in \mathcal{N}_1(i) \cap \mathcal{N}_2(j)\right\}$ *for all different mobility settings (a prime indicates a node after movement).*

Putting all intermediate results together we obtain a rather cumbersome expression for the expected Frobeniusnorm,

$$
\begin{aligned}
&\mathrm{E}_{\mathcal{T}}\{g_{ij}^2\} = \\
&I_{\mathrm{m}}^3 \Bigg(-6p^4w^4 + \frac{(27\pi^2 + 9\sqrt{3}\pi - 80)p^3w^4}{3\pi^2} - \left(\frac{3\sqrt{3}}{2\pi} + 3\right) p^2w^4 \Bigg) \\
&+ I_{\mathrm{m}}^2 \Bigg(\Bigg(-5p^4w^4 + \frac{(15\pi^2 + 9\sqrt{3}\pi - 16)p^3w^4}{3\pi^2} \Bigg) I^2 \\
&\quad + \Bigg(34p^4w^4 - p^3 \left(\frac{(165\pi^2 + 63\sqrt{3}\pi - 240)w^4}{3\pi^2} - 12w^3 \right) + p^2 \left(\frac{(42\pi + 15\sqrt{3})w^4}{2\pi} - \frac{(12\pi + 3\sqrt{3})w^3}{\pi} \right) \Bigg) I \\
&\quad - 30p^4w^4 + p^3 \left(\frac{(189\pi^2 + 63\sqrt{3}\pi - 176)w^4}{3\pi^2} - 24w^3 \right) - p^2 \left(\frac{(82\pi + 21\sqrt{3})w^4}{2\pi} - \frac{(40\pi + 6\sqrt{3})w^3}{\pi} + 8w^2 \right) + p\big(8w^4 - 16w^3 + 8w^2\big) \Bigg) \\
&+ I_{\mathrm{m}} \Bigg(\Bigg(10p^4w^4 - \frac{(30\pi^2 + 18\sqrt{3}\pi - 32)p^3w^4}{3\pi^2} \Bigg) I^3 \\
&\quad + \Bigg(-55p^4w^4 + p^3 \left(\frac{(264\pi^2 + 108\sqrt{3}\pi - 256)w^4}{3\pi^2} - 24w^3 \right) - p^2 \left(\frac{(66\pi + 21\sqrt{3})w^4}{2\pi} - \frac{(24\pi + 6\sqrt{3})w^3}{\pi} \right) \Bigg) I^2 \\
&\quad - \Bigg(76p^4w^4 + p^3 \left(-\frac{(462\pi^2 + 162\sqrt{3}\pi - 352)w^4}{3\pi^2} + 60w^3 \right) + p^2 \left(\frac{(188\pi + 48\sqrt{3})w^4}{2\pi} - \frac{(92\pi + 15\sqrt{3})w^3}{\pi} + 16w^2 \right) - p\big(16w^4 - 32w^3 + 16w^2\big) \Bigg) I \\
&\quad + 24p^4w^4 + p^3 \left(\frac{(162\pi^2 + 54\sqrt{3}\pi - 96)w^4}{3\pi^2} - 24w^3 \right) - p^2 \left(\frac{(76\pi + 18\sqrt{3})w^4}{2\pi} - \frac{(40\pi + 6\sqrt{3})w^3}{\pi} + 8w^2 \right) + p\big(8w^4 - 16w^3 + 8w^2\big) \Bigg) \\
&+ p^4w^4I^5 + \Bigg(-10p^4w^4 + p^3 \left(\left(\frac{3\sqrt{3}}{\pi} + \frac{3\pi^2 - 16}{3\pi^2} + 8 \right) w^4 - 4w^3 \right) \Bigg) I^4 \\
&\quad + \Bigg(35p^4w^4 - p^3 \left(\left(\frac{18\sqrt{3}}{\pi} + \frac{6\pi^2 - 32}{\pi^2} + 48 \right) w^4 - 24w^3 \right) + p^2 \left(\frac{(38\pi + 9\sqrt{3})w^4}{2\pi} - \left(\frac{3\sqrt{3}}{\pi} + 20 \right) w^3 + 6w^2 \right) \Bigg) I^3 \\
&\quad + \Bigg(-50p^4w^4 + p^3 \left(\left(\frac{33\sqrt{3}}{\pi} + \frac{33\pi^2 - 176}{3\pi^2} + 88 \right) w^4 - 44w^3 \right) - p^2 \left(\frac{(114\pi + 27\sqrt{3})w^4}{2\pi} - \left(\frac{9\sqrt{3}}{\pi} + 60 \right) w^3 + 18w^2 \right) + p\big(8w^4 - 16w^3 + 12w^2 - 4w\big) \Bigg) I^2 \\
&\quad + \Bigg(24p^4w^4 - p^3 \left(\left(\frac{18\sqrt{3}}{\pi} + \frac{6\pi^2 - 32}{\pi^2} + 48 \right) w^4 - 24w^3 \right) + p^2 \left(\frac{(76\pi + 18\sqrt{3})w^4}{2\pi} - \left(\frac{6\sqrt{3}}{\pi} + 40 \right) w^3 + 12w^2 \right) - p\big(8w^4 - 16w^3 + 12w^2 - 4w\big) + 1 \Bigg) I
\end{aligned}
$$

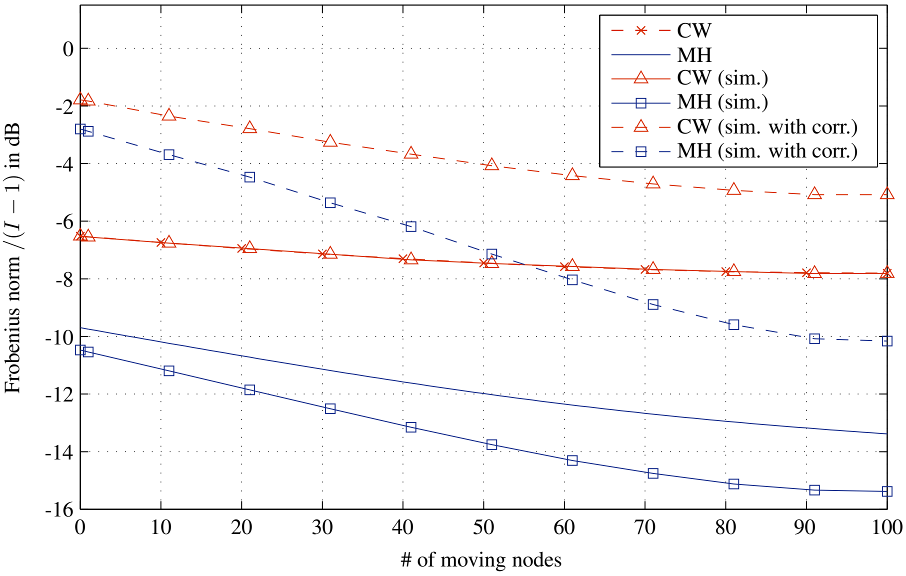


Figure 5.9: *Frobenius norm versus the number of moving nodes for different weight designs and initial correlations.*

5.2 Numerical Results

Throughout this section we are using periodic (toroidal) random geometric graphs with $I = 100$ and $\mathcal{A} = [0, 1] \times [0, 1]$. $I_{\mathrm{m}} \leq 100$ of these nodes move around randomly, with their positions in each time instant being i.i.d. uniformly distributed. Unless stated otherwise, the graph connectivity was $c = 1.6$ (cf. Section 2.1.2) and the initial measurements were i.i.d. Gaussian with zero mean.

Fig. 5.9 shows analytical and numerical results for the mean Frobenius norm $\mathrm{E}_{\mathcal{T}}\{\omega_1\}$ obtained with the CW and MH weight design. For CW, we choose w equal to the reciprocal of the average maximum degree, estimated from 500 Monte Carlo runs. It is seen that for CW, analytical and numerical results match exactly, thus confirming the analysis from the previous Section. While this analysis does not apply directly to MH, using the CW result and replacing w with the mean of the arithmetic average of the weights yields a reasonable (but slightly pessimistic) approximation. For both weight designs, the Frobenius norm decreases with increasing number of mobile nodes. While the gains seem to be moderate in this figure, it should be kept in mind that $\mathrm{E}_{\mathcal{T}}\{\omega_1\}$ quantifies the MSE improvement for i.i.d. measurements after only two iterations. For comparison, we also show $\mathrm{E}_{\mathcal{T}}\{\mathrm{tr}\{\mathbf{R}_s \mathbf{W}_{2\to 1}^T \mathbf{W}_{2\to 1}\}\}$ for the case of correlated measurements, where the mobility gain is even larger.

We next study the convergence of AC in static and mobile WSN (with 20 nodes moving). Fig. 5.10 shows the MSE versus k, averaged over 100 scenarios, for the static case with three different weight designs and for the mobile case with MH and CW (including our analytical lower bound for CW). Clearly, both MH and CW in mobile WSN outperform the static case by far, with mobile MH performing best. The lower bound for mobile CW is exact for $k = 2$ but becomes less tight as k increases. While both mobile CW and mobile MH are realistic for practical mobile WSN applications, the MH design requires to determine the local node degrees in each iteration step. By contrast, the CW design has to be done only once. Finally, since the graph realizations

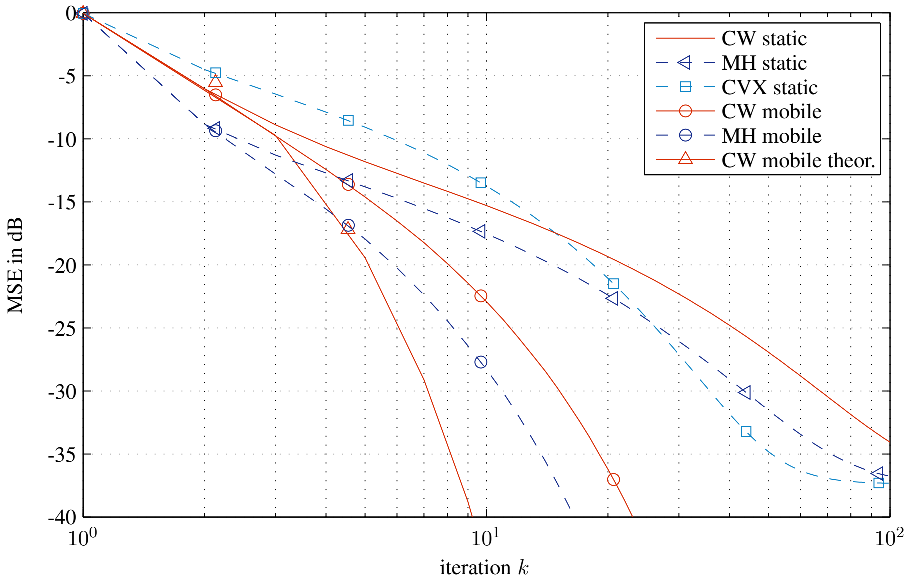


Figure 5.10: *MSE behavior over the iteration number* k *for* 20 *nodes moving.*

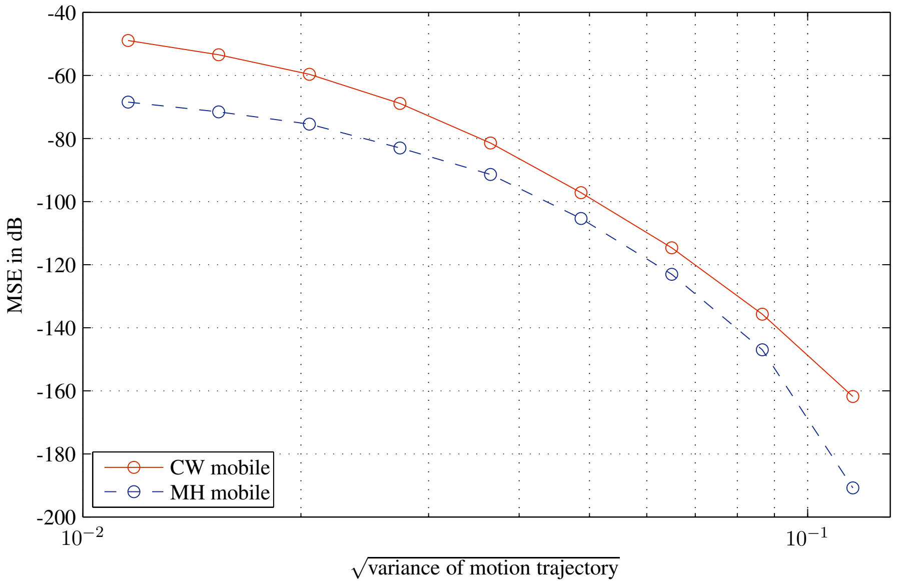


Figure 5.11: *MSE versus the standard deviation of the distance traveled by the nodes.*

are not necessarily connected, the MSE saturates for static settings.

The next result illustrates the dependence of the MSE after 100 iterations on the amount of mobility. Here, 50 of the 100 nodes are moving a random distance in a given direction, perturbed by a small Gaussian jitter; the direction is initially chosen randomly and then remains constant. Fig. 5.11 shows the MSE versus the standard deviation of the distance traveled by the mobile nodes, which quantifies the amount of mobility. It is seen that the MSE decreases rapidly with increasing node mobility. Compared to the static case, where CW and MH achieve an MSE of about -30 dB and -42 dB, respectively, the MSE improvement with high mobility is more than 130 dB.

Our last simulation result, shown in Fig. 5.12, shows the MSE achieved with CW and MH after 50 and 150

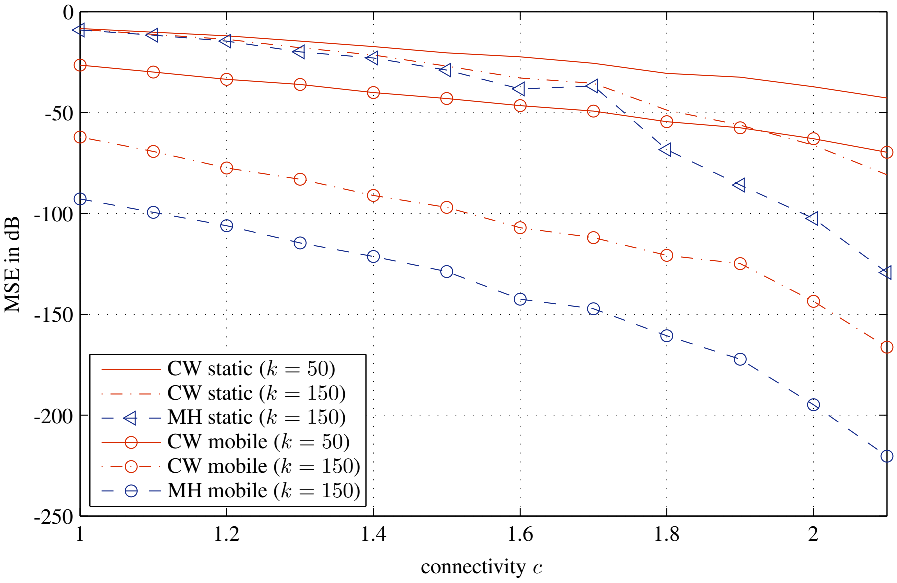


Figure 5.12: *Impact of graph connectivity on the MSE.*

iterations for different graph connectivities c (again averaged over 100 scenarios). For the mobile case, 20 out of 100 nodes are moving according the model described for the previous simulation (the standard deviation of the traveled distance here was 0.114). For very low connectivity, it is very likely that the graphs are not connected. Hence, for the static case there is little MSE improvement when going from 50 to 150 iterations below the connectivity threshold of $c \leq 1.7$. The behavior for the mobile WSN is quite different. Apart from performing uniformly better than the static WSN, AC works very well even way below the connectivity threshold. This can be attributed to the fact that node motion allows information to diffuse even between graph components that are temporarily not connected. Since the communication range relates directly to the node transmit power, it follows that mobile WSN can achieve the same performance as static WSN with much less transmit power. For example, to achieve an MSE of -100 dB after 150 iterations with MH, the transmit power in the mobile WSN can be about 6 dB smaller than in the static WSN.

5.3 Conclusion

In this chapter we studied the MSE performance of average consensus (AC) in mobile WSN. Modeling the mobile WSN via stationary Markovian evolving graphs, we derived a closed-form lower and upper bound bound for the MSE that corroborates the beneficial impact of node mobility on the performance of distributed averaging. The bounds and corresponding numerical results reveal that the MSE gain resulting from node mobility increases with increasing number of mobile nodes and higher node mobility. Interestingly, AC in mobile WSN can also overcome the limits of graph connectivity. Furthermore, the improved MSE translates into shorter averaging times or reduced transmit power.

5.A Expectation of the Frobenius Norm

In the following we show how to derive the expectation of the Frobenius norm of $\mathbf{G} = \mathbf{W}_1\mathbf{W}_2$. To do so, we denote element $[\mathbf{W}_1]_{ij}$ simply as $w_{ij}^{(1)}$ and $[\mathbf{W}_2]_{ij}$ as $w_{ij}^{(2)}$, respectively. The derivation is divided into two parts, in the first one we compute $\mathrm{E}_{\mathcal{T}}\{g_{ii}^2 | \cdot\}$, in the second $\mathrm{E}_{\mathcal{T}}\{g_{ij}^2 | \cdot\}$. Both expectations are determined through probabilities which are dependent on whether the corresponding node and its neighbors move or not. For simplicity we are going to use $\mathrm{E}\{\cdot\}$ instead of $\mathrm{E}_{\mathcal{T}}\{\cdot|\cdot\}$ in this section.

5.A.1 Diagonal Elements

The expectation of any diagonal element can be separated as

$$\mathrm{E}\{g_{ii}^2\} = \sum_{\forall k \in \mathcal{V}} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{ik}^{(2)}\right)^2\right\} + \sum\sum_{\forall k \in \mathcal{V}, \forall j \in \mathcal{V}\setminus k} \mathrm{E}\left\{w_{ik}^{(1)} w_{il}^{(1)} w_{ik}^{(2)} w_{il}^{(2)}\right\}, \tag{5.11}$$

where the first term yields

$$\sum_{\forall k \in \mathcal{V}} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{ik}^{(2)}\right)^2\right\} = \mathrm{E}\left\{\left(w_{ii}^{(1)} w_{ii}^{(2)}\right)^2\right\} + \sum_{\forall k \in \mathcal{V}\setminus i} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{ik}^{(2)}\right)^2\right\}. \tag{5.12}$$

It is easy to see that we have

$$\sum_{\forall k \in \mathcal{V}\setminus i} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{ik}^{(2)}\right)^2\right\} = \underline{p}_1^2 w^4 (I-1),$$

with $\underline{p}_1^2 = \mathrm{P}\left\{k \in \mathcal{N}_1(i) \cap \mathcal{N}_2(i)\right\}$, which describes the probability of node i being a neighbor of node k before and after movement, and w is the expected weight for a constant weight (CW) design.

Unfortunately the first term of (5.12) cannot be computed in such a trivial way, i.e.,

$$\begin{aligned}
\mathrm{E}\left\{\left(w_{ii}^{(1)} w_{ii}^{(2)}\right)^2\right\} &= \mathrm{E}\left\{\left(1 - \sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)}\right)^2 \left(1 - \sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)}\right)^2\right\} \\
&= 1 - 2\mathrm{E}\left\{\sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)}\right\} - 2\mathrm{E}\left\{\sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)}\right\} + \mathrm{E}\left\{\left(\sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)}\right)^2\right\} \\
&\quad + \mathrm{E}\left\{\left(\sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)}\right)^2\right\} + 4\mathrm{E}\left\{\sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)} \sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)}\right\} \\
&\quad - 2\mathrm{E}\left\{\sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)} \left(\sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)}\right)^2\right\} - 2\mathrm{E}\left\{\left(\sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)}\right)^2 \sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)}\right\} \\
&\quad + \mathrm{E}\left\{\left(\sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)}\right)^2 \left(\sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)}\right)^2\right\}.
\end{aligned} \tag{5.13}$$

The expression $w_{ii} = 1 - \sum_{j \in \mathcal{N}(i)} w_{ij}$ is equal to $w_{ii} = 1 - \sum_{\forall j \in \mathcal{V}\setminus i} w_{ij}$ since we have $w_{ij} = 0$ if $(i,j) \notin \mathcal{E}$.

In the following we provide the results for each term of (5.13). First of all, using p (which represents the probability that any two nodes are neighbors) we have

$$2\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i} w_{in}^{(1)}\right\} + 2\mathrm{E}\left\{\sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right\} = 4pw(I-1)\,,$$

which is intuitive, since we expect that we have $p(I-1)$ neighbors of node i and all weights equal w. Then, the next term computes as

$$\mathrm{E}\left\{\left(\sum_{\forall n\in\mathcal{V}\backslash i} w_{in}^{(1)}\right)^2\right\} + \mathrm{E}\left\{\left(\sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right)^2\right\} = 2w^2(p(I-1)+p^2(I-1)(I-2))\,,$$

where we have again only static dependencies which can be derived very simple.

The next expectation again involves movement by containing $\underline{p}_1^2$, i.e.,

$$\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i} w_{in}^{(1)} \sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right\} = w^2(p^2(I-1)(I-2)+\underline{p}_1^2(I-1))\,, \tag{5.14}$$

where the first term follows from the property that the events of n being a neighbor of i before movement and m being neighbor after movement are statistically independent. For the case $n=m$ we have to model the relation with $\underline{p}_1^2$.

The next expectation is a bit more complicated. In the first step we are separating the expression into two terms,

$$\begin{aligned}\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i} w_{in}^{(1)}\left(\sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right)^2\right\} &= \mathrm{E}\left\{\left(\sum_{\forall n\in\mathcal{V}\backslash i} w_{in}^{(1)}\right)^2 \sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right\}\\ &= \mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i}\left(w_{in}^{(1)}\right)^2 \sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right\} + \mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\}}\sum w_{in}^{(1)}w_{il}^{(1)} \sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right\}\,,\end{aligned}$$

where the equality in the first row follows from the property that our considered movement could also work the other way around. The resulting two terms are computed separately, i.e.,

$$\begin{aligned}\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i}\left(w_{in}^{(1)}\right)^2 \sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right\} &= \mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i}\left(w_{in}^{(1)}\right)^2 w_{in}^{(2)}\right\} + \mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i}\left(w_{in}^{(1)}\right)^2 \sum_{\forall m\in\mathcal{V}\backslash\{i,n\}} w_{im}^{(2)}\right\}\\ &= w^3\left(p^2(I-1)(I-2)+\underline{p}_1^2(I-1)\right)\,,\end{aligned}$$

which is obtained similarly to (5.14). The second term yields

$$\begin{aligned}\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\}}\sum w_{in}^{(1)}w_{il}^{(1)} \sum_{\forall m\in\mathcal{V}\backslash i} w_{im}^{(2)}\right\} &= \mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\}}\sum w_{in}^{(1)}w_{il}^{(1)}w_{in}^{(2)}\right\}\\ &+\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\}}\sum w_{in}^{(1)}w_{il}^{(1)}w_{il}^{(2)}\right\} + \mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\}}\sum w_{in}^{(1)}w_{il}^{(1)} \sum_{\forall m\in\mathcal{V}\backslash\{i,n,l\}} w_{im}^{(2)}\right\}\end{aligned}$$

$$= p\underline{p}_1^2 w^3(I-1)(I-2) + p\underline{p}_1^2 w^3(I-1)(I-2) + p^3 w^3(I-1)(I-2)(I-3)\,,$$

which is done straightforward. Therefore, we have

$$\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\setminus i} w_{in}^{(1)}\left(\sum_{\forall m\in\mathcal{V}\setminus i} w_{im}^{(2)}\right)^2\right\} = w^3(I-1)(p^2(I-2)(1+p(I-3)) + \underline{p}_1^2(1+2p(I-2)))\,.$$

Finally, we consider the last term of (5.13) where we first develop the squares

$$\begin{aligned}
&\mathrm{E}\left\{\left(\sum_{\forall n\in\mathcal{V}\setminus i} w_{in}^{(1)}\right)^2\left(\sum_{\forall m\in\mathcal{V}\setminus i} w_{im}^{(2)}\right)^2\right\}\\
&\quad= \mathrm{E}\left\{\left(\sum_{\forall n\in\mathcal{V}\setminus i}\left(w_{in}^{(1)}\right)^2 + \sum\sum_{\forall n\in\mathcal{V}\setminus i,\forall l\in\mathcal{V}\setminus\{n,i\}} w_{in}^{(1)}w_{il}^{(1)}\right)\left(\sum_{\forall m\in\mathcal{V}\setminus i}\left(w_{im}^{(2)}\right)^2 + \sum\sum_{\forall m\in\mathcal{V}\setminus i,\forall k\in\mathcal{V}\setminus\{m,i\}} w_{im}^{(2)}w_{ik}^{(2)}\right)\right\}\,.
\end{aligned} \tag{5.15}$$

We omit to write down the result after multiplying out (5.15) and directly derive the results for the four resulting terms. The first term yields

$$\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\setminus i}\left(w_{in}^{(1)}\right)^2\sum_{\forall m\in\mathcal{V}\setminus i}\left(w_{im}^{(2)}\right)^2\right\} = w^4(I-1)(\underline{p}_1^2 + p^2(I-2))\,.$$

The second and third therm are equal and we obtain

$$\begin{aligned}
&\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\setminus i}\left(w_{in}^{(1)}\right)^2\sum\sum_{\forall m\in\mathcal{V}\setminus i,\forall k\in\mathcal{V}\setminus\{m,i\}} w_{im}^{(2)}w_{ik}^{(2)}\right\} = \mathrm{E}\left\{\sum\sum_{\forall n\in\mathcal{V}\setminus i,\forall l\in\mathcal{V}\setminus\{n,i\}} w_{in}^{(1)}w_{il}^{(1)}\sum_{\forall m\in\mathcal{V}\setminus i}\left(w_{im}^{(2)}\right)^2\right\}\\
&\quad= pw^4(I-1)(I-2)(2\underline{p}_1^2 + p^2(I-3))\,.
\end{aligned}$$

The computation of the fourth term is slightly more involved. First we group it into similar expressions, which we can then evaluate, i.e.,

$$\begin{aligned}
&\mathrm{E}\left\{\sum\sum_{\forall n\in\mathcal{V}\setminus i,\forall l\in\mathcal{V}\setminus\{n,i\}} w_{in}^{(1)}w_{il}^{(1)}\sum\sum_{\forall m\in\mathcal{V}\setminus i,\forall k\in\mathcal{V}\setminus\{m,i\}} w_{im}^{(2)}w_{ik}^{(2)}\right\}\\
&\quad= \mathrm{E}\left\{\sum\sum_{\forall n\in\mathcal{V}\setminus i,\forall l\in\mathcal{V}\setminus\{n,i\}} w_{in}^{(1)}w_{il}^{(1)}w_{in}^{(2)}w_{il}^{(2)}\right\}\\
&\qquad+ \mathrm{E}\left\{\sum\sum_{\forall m\in\mathcal{V}\setminus i,\forall k\in\mathcal{V}\setminus\{m,i\}} w_{im}^{(1)}w_{ik}^{(1)}w_{im}^{(2)}w_{ik}^{(2)}\right\}\\
&\qquad+ \mathrm{E}\left\{\sum\sum_{\forall n\in\mathcal{V}\setminus i,\forall l\in\mathcal{V}\setminus\{n,i\}} w_{in}^{(1)}w_{il}^{(1)}w_{in}^{(2)}\sum_{\forall m\in\mathcal{V}\setminus\{i,n,l\}} w_{im}^{(2)}\right\}\\
&\qquad+ \mathrm{E}\left\{\sum\sum_{\forall n\in\mathcal{V}\setminus i,\forall l\in\mathcal{V}\setminus\{n,i\}} w_{in}^{(1)}w_{il}^{(1)}w_{in}^{(2)}\sum_{\forall k\in\mathcal{V}\setminus\{i,n,l\}} w_{ik}^{(2)}\right\}
\end{aligned}$$

$$
\begin{aligned}
&+\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\}}\sum w_{in}^{(1)}w_{il}^{(1)}w_{il}^{(2)}\sum_{\forall m\in\mathcal{V}\backslash\{i,n,l\}}w_{im}^{(2)}\right\}\\
&+\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\}}\sum w_{in}^{(1)}w_{il}^{(1)}w_{il}^{(2)}\sum_{\forall k\in\mathcal{V}\backslash\{i,n,l\}}w_{ik}^{(2)}\right\}\\
&+\mathrm{E}\left\{\sum_{\forall n\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{n,i\},\forall m\in\mathcal{V}\backslash\{n,l,i\},\forall k\in\mathcal{V}\backslash\{n,l,m,i\}}\sum\sum\sum w_{in}^{(1)}w_{il}^{(1)}w_{im}^{(2)}w_{ik}^{(2)}\right\}\\
&= w^4(I-1)(I-2)(2\underline{p}_5^4+4\underline{p}_1^2p^2(I-3)+p^4(I-3)(I-4))
\end{aligned}
$$

with $\underline{p}_5^4 = \mathrm{P}\left\{i\in\mathcal{N}_1(n)\cap\mathcal{N}_1(l), i\in\mathcal{N}_2(n)\cap\mathcal{N}_2(l)\right\}$. Combining the above derived expressions, we obtain for (5.15)

$$
\begin{aligned}
&\mathrm{E}\left\{\left(\sum_{\forall n\in\mathcal{V}\backslash i}w_{in}^{(1)}\right)^2\left(\sum_{\forall m\in\mathcal{V}\backslash i}w_{im}^{(2)}\right)^2\right\}\\
&\qquad= w^4(I-1)((\underline{p}_1^2+p^2(I-2))+2p(I-2)(2\underline{p}_1^2+p^2(I-3))+(I-2)(\underline{p}_5^42+4\underline{p}_1^2p(I-3)+p^4))\,.
\end{aligned}
$$

The last missing part is to compute the second term of (5.11). Again we start the derivation by grouping similar components, i.e.,

$$
\begin{aligned}
&\sum_{\forall k\in\mathcal{V},\forall l\in\mathcal{V}\backslash k}\sum\mathrm{E}\left\{w_{ik}^{(1)}w_{il}^{(1)}w_{ik}^{(2)}w_{il}^{(2)}\right\}\\
&\quad=\sum_{\forall l\in\mathcal{V}\backslash i}\mathrm{E}\left\{w_{ii}^{(1)}w_{il}^{(1)}w_{ii}^{(2)}w_{il}^{(2)}\right\}+\sum_{\forall k\in\mathcal{V}\backslash i}\mathrm{E}\left\{w_{ik}^{(1)}w_{ii}^{(1)}w_{ik}^{(2)}w_{ii}^{(2)}\right\}+\sum_{\forall k\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{k,i\}}\sum\mathrm{E}\left\{w_{ik}^{(1)}w_{il}^{(1)}w_{ik}^{(2)}w_{il}^{(2)}\right\}
\end{aligned}
$$

where the third term is simply

$$
\sum_{\forall k\in\mathcal{V}\backslash i,\forall l\in\mathcal{V}\backslash\{k,i\}}\sum\mathrm{E}\left\{w_{ik}^{(1)}w_{il}^{(1)}w_{ik}^{(2)}w_{il}^{(2)}\right\}=\underline{p}_5^4w^4(I-1)(I-2)\,.
$$

The first and second terms are equal, i.e.,

$$
\sum_{\forall l\in\mathcal{V}\backslash i}\mathrm{E}\left\{w_{ii}^{(1)}w_{il}^{(1)}w_{ii}^{(2)}w_{il}^{(2)}\right\}=\sum_{\forall k\in\mathcal{V}\backslash i}\mathrm{E}\left\{w_{ik}^{(1)}w_{ii}^{(1)}w_{ik}^{(2)}w_{ii}^{(2)}\right\}\,,
$$

and it is seen that all elements of the sum are equal (invariant of l and k, respectively) and since they can be computed separately as

$$
\begin{aligned}
\mathrm{E}\left\{w_{ii}^{(1)}w_{il}^{(1)}w_{ii}^{(2)}w_{il}^{(2)}\right\}&=\mathrm{E}\left\{\left(1-w_{il}^{(1)}-\sum_{\forall n\in\mathcal{V}\backslash i}w_{in}^{(1)}\right)w_{il}^{(1)}\left(1-w_{il}^{(2)}-\sum_{\forall m\in\mathcal{V}\backslash i}w_{im}^{(2)}\right)w_{il}^{(2)}\right\}\\
&=\mathrm{E}\left\{w_{il}^{(1)}w_{il}^{(2)}-w_{il}^{(1)}\left(w_{il}^{(2)}\right)^2-\left(w_{il}^{(1)}\right)^2w_{il}^{(2)}+\left(w_{il}^{(1)}w_{il}^{(2)}\right)^2\right.\\
&\qquad-w_{il}^{(1)}w_{il}^{(2)}\sum_{\forall n\in\mathcal{V}\backslash i}w_{in}^{(1)}-w_{il}^{(1)}w_{il}^{(2)}\sum_{\forall m\in\mathcal{V}\backslash i}w_{im}^{(2)}+w_{il}^{(1)}\left(w_{il}^{(2)}\right)^2\sum_{\forall n\in\mathcal{V}\backslash i}w_{in}^{(1)}
\end{aligned}
$$

$$
\begin{aligned}
&\left. + \left(w_{il}^{(1)}\right)^2 w_{il}^{(2)} \sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)} + w_{il}^{(1)} w_{il}^{(2)} \sum_{\forall n \in \mathcal{V}\setminus i} w_{in}^{(1)} \sum_{\forall m \in \mathcal{V}\setminus i} w_{im}^{(2)} \right\} \\
&= \underline{p}_1^2 w^2 \left(1 - 2w + w^2 - 2pw(I-2)(1-w) + p^2 w^2 (I-2)(I-3)\right) + \underline{p}_5^4 w^4 (I-2) \,.
\end{aligned}
$$

Putting all results of this section together we finally obtain an expression for the expectation of a diagonal element:

$$
\begin{aligned}
\mathrm{E}\{g_{ii}^2\} = {} & \underline{p}_1^2 w^4 (I-1) \\
& + 1 - 4pw(I-1) + 2w^2(p(I-1) + p^2(I-1)(I-2)) \\
& + 4(w^2(p^2(I-1)(I-2) + \underline{p}_1^2(I-1))) \\
& - 4(w^3(I-1)(p^2(I-2)(1 + p(I-3)) + \underline{p}_1^2(1 + 2p(I-2)))) \\
& + w^4(I-1)((\underline{p}_1^2 + p^2(I-2)) + 2p(I-2)(2\underline{p}_1^2 + p^2(I-3)) + (I-2)(2\underline{p}_5^4 + 4\underline{p}_1^2 p^2 (I-3) + p^4(I-3)(I-4))) \\
& + \underline{p}_5^4 w^4 (I-1)(I-2) \\
& + 2(I-1)\left(\underline{p}_1^2 w^2 \left(1 - 2w + w^2 - 2pw(I-2)(1-w) + p^2 w^2 (I-2)(I-3)\right) + \underline{p}_5^4 w^4 (I-2)\right) \,,
\end{aligned}
$$

where we kept the order of the summands as we derived them.

5.A.2 Off-diagonal Elements

In this section we derive the expression for the expectation of the off-diagonal elements. We proceed in the similar fashion as in the previous section. We start with the definition and separate it as

$$
\mathrm{E}\{g_{ij}^2\} = \sum_{\forall k \in \mathcal{V}} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{jk}^{(2)}\right)^2\right\} + \sum_{\forall k \in \mathcal{V}, \forall j \in \mathcal{V}\setminus k} \sum \mathrm{E}\left\{w_{ik}^{(1)} w_{il}^{(1)} w_{jk}^{(2)} w_{jl}^{(2)}\right\} \,. \tag{5.16}
$$

First, we pay our attention to the first term of (5.16). Developing the square leads to

$$
\sum_{\forall k \in \mathcal{V}} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{jk}^{(2)}\right)^2\right\} = \mathrm{E}\left\{\left(w_{ii}^{(1)} w_{ji}^{(2)}\right)^2\right\} + \mathrm{E}\left\{\left(w_{ij}^{(1)} w_{jj}^{(2)}\right)^2\right\} + \sum_{\forall k \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{jk}^{(2)}\right)^2\right\} \,, \tag{5.17}
$$

where the last term yields

$$
\sum_{\forall k \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{\left(w_{ik}^{(1)} w_{jk}^{(2)}\right)^2\right\} = p^2 w^4 (I-2) \,,
$$

because it contains no information about the movement model. The first and second term of (5.17) are equal and we can derive

$$
\begin{aligned}
\mathrm{E}\left\{\left(w_{ii}^{(1)} w_{ji}^{(2)}\right)^2\right\} & = \mathrm{E}\left\{\left(w_{ij}^{(1)} w_{jj}^{(2)}\right)^2\right\} \\
& = \mathrm{E}\left\{\left(\left(1 - w_{ij}^{(1)} - \sum_{\forall n \in \mathcal{V}\setminus\{i,j\}} w_{in}^{(1)}\right) w_{ij}^{(2)}\right)^2\right\} \\
& = \mathrm{E}\left\{ \left(w_{ij}^{(2)}\right)^2 + \left(w_{ij}^{(1)} w_{ij}^{(2)}\right)^2 + \left(w_{ij}^{(2)}\right)^2 \left(\sum_{\forall n \in \mathcal{V}\setminus\{i,j\}} w_{in}^{(1)}\right)^2 - 2 w_{ij}^{(1)} \left(w_{ij}^{(2)}\right)^2 \right.
\end{aligned}
$$

$$
\begin{aligned}
&- 2\left(w_{ij}^{(2)}\right)^2 \sum_{\forall n\in\mathcal{V}\backslash\{i,j\}} w_{in}^{(1)} + 2w_{ij}^{(1)}\left(w_{ij}^{(2)}\right)^2 \sum_{\forall n\in\mathcal{V}\backslash\{i,j\}} w_{in}^{(1)}\Bigg\} \\
&= pw^2 + \underline{p}_2^2 w^4 + w^4 p\left(p(I-2)+p^2(I-2)(I-3)\right) - 2\underline{p}_2^2 w^3 - 2p^2w^3(I-2) + 2\underline{p}_2^2 p w^4 (I-2)
\end{aligned}
$$

with $\underline{p}_2^2 = \mathrm{P}\left\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_2(j)\right\}$. It is seen that $\underline{p}_2^2$ models the probability that node i is in the neighborhood of node j before and after movement.

In the next step we derive the expected values of the second term in (5.16). To do so, we split the addition into similar terms, i.e.,

$$
\begin{aligned}
\sum\sum_{\forall k\in\mathcal{V},\forall l\in\mathcal{V}\backslash k} \mathrm{E}\left\{w_{ik}^{(1)}w_{il}^{(1)}w_{jk}^{(2)}w_{jl}^{(2)}\right\} &= \mathrm{E}\left\{w_{ii}^{(1)}w_{ij}^{(1)}w_{ij}^{(2)}w_{jj}^{(2)}\right\} + \mathrm{E}\left\{w_{ii}^{(1)}w_{ij}^{(1)}w_{ij}^{(2)}w_{jj}^{(2)}\right\} \\
&+ \sum_{\forall l\in\mathcal{V}\backslash\{i,j\}} \mathrm{E}\left\{w_{ii}^{(1)}w_{il}^{(1)}w_{ij}^{(2)}w_{jl}^{(2)}\right\} + \sum_{\forall k\in\mathcal{V}\backslash\{i,j\}} \mathrm{E}\left\{w_{ik}^{(1)}w_{ii}^{(1)}w_{jk}^{(2)}w_{ji}^{(2)}\right\} \\
&+ \sum_{\forall l\in\mathcal{V}\backslash\{i,j\}} \mathrm{E}\left\{w_{ij}^{(1)}w_{il}^{(1)}w_{jj}^{(2)}w_{jl}^{(2)}\right\} + \sum_{\forall k\in\mathcal{V}\backslash\{i,j\}} \mathrm{E}\left\{w_{ik}^{(1)}w_{ij}^{(1)}w_{jk}^{(2)}w_{jj}^{(2)}\right\} \\
&+ \sum\sum_{\forall k\in\mathcal{V}\backslash\{i,j\},\forall l\in\mathcal{V}\backslash\{k,i,j\}} \mathrm{E}\left\{w_{ik}^{(1)}w_{il}^{(1)}w_{ik}^{(2)}w_{jl}^{(2)}\right\}\,, \qquad (5.18)
\end{aligned}
$$

where the last term yields

$$
\sum\sum_{\forall k\in\mathcal{V}\backslash\{i,j\},\forall l\in\mathcal{V}\backslash\{k,i,j\}} \mathrm{E}\left\{w_{ik}^{(1)}w_{il}^{(1)}w_{jk}^{(2)}w_{jl}^{(2)}\right\} = \underline{p}_7^4 w^4 (I-2)(I-3)
$$

with $\underline{p}_7^4 = \mathrm{P}\left\{k \in \mathcal{N}_1(i) \cap \mathcal{N}_2(j), l \in \mathcal{N}_1(i) \cap \mathcal{N}_2(j)\right\}$. The probability $\underline{p}_7^4$ models the relation between node k and i before and k and j after movement, and the relation between node l, i, and j in the opposite way.

Further we consider the first and second term of (5.18) which are equal and obtain the expectation by substituting the elements $w_{ii}^{(1)}$ and $w_{jj}^{(2)}$ in a convenient way. We obtain

$$
\begin{aligned}
&\mathrm{E}\left\{w_{ii}^{(1)}w_{ij}^{(1)}w_{ij}^{(2)}w_{jj}^{(2)}\right\} = \mathrm{E}\left\{w_{ii}^{(1)}w_{ij}^{(1)}w_{ij}^{(2)}w_{jj}^{(2)}\right\} \\
&= \mathrm{E}\left\{\left(1 - w_{ij}^{(1)} - \sum_{\forall k\in\mathcal{V}\backslash\{i,j\}} w_{ik}^{(1)}\right)\left(1 - w_{ij}^{(2)} - \sum_{\forall l\in\mathcal{V}\backslash\{i,j\}} w_{jl}^{(2)}\right) w_{ij}^{(1)}w_{ij}^{(2)}\right\} \\
&= \mathrm{E}\Bigg\{ w_{ij}^{(1)}w_{ij}^{(2)} - (w_{ij}^{(1)})^2 w_{ij}^{(2)} - w_{ij}^{(1)}(w_{ij}^{(2)})^2 - w_{ij}^{(1)}w_{ij}^{(2)} \sum_{\forall k\in\mathcal{V}\backslash\{i,j\}} w_{ik}^{(1)} - w_{ij}^{(1)}w_{ij}^{(2)} \sum_{\forall l\in\mathcal{V}\backslash\{i,j\}} w_{jl}^{(2)} \\
&\quad + \left(w_{ij}^{(1)}w_{ij}^{(2)}\right)^2 + w_{ij}^{(1)}\left(w_{ij}^{(2)}\right)^2 \sum_{\forall k\in\mathcal{V}\backslash\{i,j\}} w_{ik}^{(1)} + \left(w_{ij}^{(1)}\right)^2 w_{ij}^{(2)} \sum_{\forall l\in\mathcal{V}\backslash\{i,j\}} w_{jl}^{(2)} \\
&\quad + w_{ij}^{(1)}w_{ij}^{(2)} \sum_{\forall k\in\mathcal{V}\backslash\{i,j\}} w_{ik}^{(1)} \sum_{\forall l\in\mathcal{V}\backslash\{i,j\}} w_{jl}^{(2)} \Bigg\} \\
&= \underline{p}_2^2 w^2 - 2\underline{p}_2^2 w^3 - 2(I-2)\underline{p}_2^2 p w^3 + \underline{p}_2^2 w^4 + 2(I-2)\underline{p}_2^2 p w^4 + w^4(\underline{p}_6^4(I-2) + \underline{p}_2^2 p^2 (I-2)(I-3))\,,
\end{aligned}
$$

where we used

$$
\mathrm{E}\left\{ w_{ij}^{(1)}w_{ij}^{(2)} \sum_{\forall k\in\mathcal{V}\backslash\{i,j\}} w_{ik}^{(1)} \sum_{\forall l\in\mathcal{V}\backslash\{i,j\}} w_{jl}^{(2)} \right\}
$$

$$= \mathrm{E}\left\{ w_{ij}^{(1)} w_{ij}^{(2)} \sum_{\forall k \in \mathcal{V}\setminus\{i,j\}} w_{ik}^{(1)} w_{jk}^{(2)} \right\} + \mathrm{E}\left\{ w_{ij}^{(1)} w_{ij}^{(2)} \sum_{\forall k \in \mathcal{V}\setminus\{i,j\}} w_{ik}^{(1)} \sum_{\forall l \in \mathcal{V}\setminus\{i,j,k\}} w_{jl}^{(2)} \right\} .$$

The probability $\underline{p}_6^4 = \mathrm{P}\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_1(l) \cap \mathcal{N}_2(j), j \in \mathcal{N}_2(l)\}$ models the case when node i is the neighbor of node j and l before and still the neighbor of node j after movement, and additionally the case j being neighbor of l after movement is taken into account.

The next derivation considers the third and forth term of (5.18) (which are equal),

$$\begin{aligned}
&\sum_{\forall l \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ w_{ii}^{(1)} w_{il}^{(1)} w_{ij}^{(2)} w_{jl}^{(2)} \right\} = \sum_{\forall k \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ w_{ik}^{(1)} w_{ii}^{(1)} w_{jk}^{(2)} w_{ji}^{(2)} \right\} \\
&= \sum_{\forall l \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ \left(1 - w_{il}^{(1)} - w_{ij}^{(1)} - \sum_{\forall n \in \mathcal{V}\setminus\{i,j,l\}} w_{in}^{(1)} \right) w_{il}^{(1)} w_{ij}^{(2)} w_{jl}^{(2)} \right\} \\
&= \sum_{\forall l \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ w_{il}^{(1)} w_{ij}^{(2)} w_{jl}^{(2)} - \left(w_{il}^{(1)}\right)^2 w_{ij}^{(2)} w_{jl}^{(2)} - w_{ij}^{(1)} w_{il}^{(1)} w_{ij}^{(2)} w_{jl}^{(2)} - w_{il}^{(1)} w_{ij}^{(2)} w_{jl}^{(2)} \sum_{\forall n \in \mathcal{V}\setminus\{i,j,l\}} w_{in}^{(1)} \right\} \\
&= (I-2)(w^3 \underline{p}_4^3 - w^4 \underline{p}_4^3 - w^4 \underline{p}_6^4 - w^4 p \underline{p}_4^3 (I-3) ,
\end{aligned}$$

with $\underline{p}_4^3 = \mathrm{P}\{j \in \mathcal{N}_1(i) \cap \mathcal{N}_1(l), i \in \mathcal{N}_2(l)\}$ which represents the condition that node j is neighbor of node i and l before the movement, and i is in the neighborhood of node l afterwards.

The fifth and sixth term of (5.18) are very similar to the third and forth, but with the difference that we have $w_{jj}^{(2)}$ instead of $w_{ii}^{(1)}$. Therefore, the result is almost the same as above, i.e.,

$$\begin{aligned}
&\sum_{\forall l \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ w_{ij}^{(1)} w_{il}^{(1)} w_{jj}^{(2)} w_{jl}^{(2)} \right\} = \sum_{\forall k \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ w_{ik}^{(1)} w_{ij}^{(1)} w_{jk}^{(2)} w_{jj}^{(2)} \right\} \\
&= \sum_{\forall l \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ w_{ij}^{(1)} w_{il}^{(1)} w_{jl}^{(2)} \left(1 - w_{jl}^{(2)} - w_{ij}^{(2)} - \sum_{\forall n \in \mathcal{V}\setminus\{i,j,l\}} w_{jn}^{(2)} \right) \right\} \\
&= \sum_{\forall l \in \mathcal{V}\setminus\{i,j\}} \mathrm{E}\left\{ w_{ij}^{(1)} w_{il}^{(1)} w_{jl}^{(2)} - w_{ij}^{(1)} w_{il}^{(1)} \left(w_{jl}^{(2)}\right)^2 - w_{ij}^{(1)} w_{il}^{(1)} w_{jl}^{(2)} w_{ij}^{(2)} - w_{ij}^{(1)} w_{il}^{(1)} w_{jl}^{(2)} \sum_{\forall n \in \mathcal{V}\setminus\{i,j,l\}} w_{jn}^{(2)} \right\} \\
&= (I-2)(w^3 \underline{p}_3^3 - w^4 \underline{p}_3^3 - w^4 \underline{p}_6^4 - w^4 p \underline{p}_3^3 (I-3)) ,
\end{aligned}$$

and instead of $\underline{p}_4^3$ we have $\underline{p}_3^3 = \mathrm{P}\{i \in \mathcal{N}_1(j) \cap \mathcal{N}_1(l), j \in \mathcal{N}_2(l)\}$. Since we condition on the mobility of node i, we have to distinguish between $\underline{p}_3^3$ and $\underline{p}_4^3$.

Finally, we obtain for the expectation of the off-diagonal elements the following expression by combining the results of this section, that is,

$$\begin{aligned}
\mathrm{E}\{g_{ij}^2\} &= p^2 w^4 (I-2) \\
&\quad + 2 \left(p w^2 + \underline{p}_2^2 w^4 + w^4 p \left(p(I-2) + p^2 (I-2)(I-3) \right) - 2 \underline{p}_2^2 w^3 - 2 p^2 w^3 (I-2) + 2 \underline{p}_2^2 p w^4 (I-2) \right) \\
&\quad + \underline{p}_7^4 w^4 (I-2)(I-3) \\
&\quad + 2 \left(\underline{p}_2^2 w^2 - 2 \underline{p}_2^2 w^3 - 2(I-2) \underline{p}_2^2 p w^3 + \underline{p}_2^2 w^4 + 2(I-2) \underline{p}_2^2 p w^4 + w^4 (\underline{p}_6^4 (I-2) + \underline{p}_2^2 p^2 (I-2)(I-3)) \right)
\end{aligned}$$

$$
\begin{aligned}
&+ 2(I-2)(w^3\underline{p}_4^3 - w^4\underline{p}_4^3 - w^4\underline{p}_6^4 - w^4p\underline{p}_4^3(I-3)) \\
&+ 2(I-2)(w^3\underline{p}_3^3 - w^4\underline{p}_3^3 - w^4\underline{p}_6^4 - w^4p\underline{p}_3^3(I-3))\,.
\end{aligned}
$$

6

Conclusions and Outlook

In the last chapter of this thesis we summarize our contributions, provide short conclusions (detailed conclusions are available at the end of the major sections), and give an outlook for potential further research topics related to our work. In the first part of this work we gave an overview of the necessary prerequisite for the understanding of our main contributions, mainly focusing on WSN and distributed averaging (CP and AC).

The first major contribution concerned distributed averaging in WSN. Here, we showed how to extend CP such that it can track time-varying signals and provided theoretical stability findings. Through the transformation in the frequency domain, we gained insight into the performance of dynamic CP and AC. Then we presented a medium access method for CP and analyzed its behavior.

In the next part of this work we presented several weight design methods for AC. The first method yields the per-step optimum weights for a given correlation of the measurements. Then we proposed a scheme where the weights are locally computed in an adaptive manner by combining MH and CVX weights. We called this method weight morphing. Finally we extended an existing method of including an advection process in the weight design. Our approach revealed to be more convenient for the usage in general WSN.

The last contribution of our work was the investigation of AC with mobile nodes. We showed through analytical bounds that under mild conditions mobility in WSN increases the performance.

6.1 Conclusions

We briefly list the main conclusions of this thesis.

- The dynamic extension of CP allows to track time-varying signals in WSN. Analytical and numerical results show that especially in (sparse) WSN it is reasonable to use dynamic CP as alternative to dynamic AC.
- Our medium access protocol allows to apply CP in WSN easily. On the one hand broadcasting reduces the communication energy, and on the other hand allowing collisions even improves the performance.
- Field estimation can be written as a least squares problem that can be solved via CP in a distributed manner.
- The regularization of AC/MH by adding 1 to the weight denominator in order to ensure stability of MH decreases the convergence speed and is not necessary in most WSN.
- Even if the optimum weight design method is unfeasible in practice, it establishes a base-line for the use of time-varying weights with AC.
- The weight morphing scheme switches between MH and CVX weights such that the performance of AC equals the better of the two, if the morphing function is designed correspondingly.
- Using our discretization of the advection-diffusion process, it is possible to apply flows even in general WSN that improve the convergence of AC, as long as the measurements vary slowly over time.
- Mobile nodes help to increase the convergence speed of AC. Hence, mobility allows to decrease the communication power without a loss of performance of AC.

6.2 Outlook

In the following we enumerate possible future research topics which arose in the context of this thesis.

Dynamic averaging in WSN

- The $\mathbf{Q}$ matrix is a parameter of CP that assigns different weights to the edges and its impact has not been studied so far. Therefore, there could be a gain in the convergence speed of CP similar to AC when tuning the weight matrix.
- Throughout our numerical investigations we observed that there do not exist standardized simulation settings to compare the performance of distributed dynamic averaging algorithms. We therefore suggest to define common settings that allow to quantify and compare the performance of distributed averaging methods.

- Our research on distributed (dynamic) averaging is practically inspired but still awaits real-world tests. We expect that implementing dynamic distributed averaging on a real WSN will raise new interesting questions and requirements even for well studied algorithms.
- An intermediate step between pure simulation and an autonomous testbed is to apply distributed dynamic averaging on real data. This means that the data need to be sampled in a real environment and are then used as the source signal in the simulations.

Weight design methods for AC

- We have seen that the optimum symmetric weights achieve excellent performance, in particular when the measurements are correlated. To allow these weights to be applied also in practice, it would be necessary to compute these weights also in a (recursive) distributed fashion.
- A similar open problem arises with the additional advection field. So far we designed the advection in a centralized manner, but since we have local conditions for the stability available, it would be worthwhile to search for a distributed method for designing these fields.
- A minor additional improvement could be achieved if the structure of the optimization methods considered with AC weight design is exploited by writing individual numerical solves. This could reduce the simulation effort significantly.

AC with mobile nodes

- For our theoretical analysis we considered a simple hopping mobility model. A evident extension is to consider more sophisticated models that provide new insights into the effect of practical node mobility.
- Our numerical results verify that our lower-bound on the MSE becomes lose with increasing number of iterations. Therefore, it is interesting to work on this bound to tighten it.

Bibliography

[1] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*. Chichester, UK: Wiley, 2010.

[2] A. Swami, Q. Zhao, Y.-W. Hong, and L. Tong, eds., *Wireless Sensor Networks: Signal Processing and Communications Perspectives*. Chichester, UK: Wiley, Oct. 2007.

[3] J. N. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*. PhD thesis, Massachusetts Institute of Technology, Dec. 1984.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[5] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[6] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2508–2530, June 2006.

[7] C. C. Moallemi and B. Van Roy, "Consensus propagation," *IEEE Trans. Inf. Theory*, vol. 52, pp. 4753–4766, Nov. 2006.

[8] C. C. Moallemi, *A Message-Passing Paradigm For Optimization*. PhD thesis, Stanford University, Sept. 2007.

[9] J. Pearl, "Reverend Bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the Second National Conference on Artificial Intelligence*, (Pittsburgh, PA), pp. 133–136, California: AAAI Press, Aug. 1982.

[10] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Understanding belief propagation and its generalizations*, pp. 239–269. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[11] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, pp. 2173–2200, Oct. 2001.

[12] J. Johnson, D. Malioutov, and A. Willsky, "Walk-sum interpretation and analysis of Gaussian belief propagation," in *Advances in Neural Information Processing Systems 18* (Y. Weiss, B. Schölkopf, and J. Platt, eds.), pp. 579–586, Cambridge, MA: MIT Press, 2006.

[13] D. Bickson, *Gaussian Belief Propagation: Theory and Application*. PhD thesis, The Hebrew University of Jerusalem, 2008.

[14] O. Shental, P. Siegel, J. Wolf, D. Bickson, and D. Dolev, "Gaussian belief propagation solver for systems of linear equations," in *Proc. ISIT*, pp. 1863–1867, July 2008.

[15] G. Zhang and R. Heusdens, "Generalized linear coordinate-descent message-passing for convex optimization," in *Proc. IEEE ICASSP-2012*, (Kyoto, JP), pp. 2009–2012, March 2012.

[16] P. Denantes, F. Benezit, P. Thiran, and M. Vetterli, “Which distributed averaging algorithm should I choose for my sensor network?,” in *Proc. IEEE INFOCOM-2008*, (Phoenix, AZ), pp. 986–994, April 2008.

[17] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray, “Asynchronous distributed averaging on communication networks,” *IEEE/ACM Trans. on Networking*, vol. 15, pp. 512–520, June 2007.

[18] T. Aysal, M. Yildiz, and A. Scaglione, “Broadcast gossip algorithms,” in *Proc. IEEE ITW-2008*, (Porto, Portugal), pp. 343–347, May 2008.

[19] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems & Control Letters*, vol. 53, pp. 65–78, Sept. 2004.

[20] T. C. Aysal, B. N. Oreshkin, and M. J. Coates, “Accelerated distributed average consensus via localized node state prediction,” *IEEE Trans. Signal Processing*, vol. 57, pp. 1563–1576, April 2009.

[21] L. Georgopoulos and M. Hasler, “Nonlinear average consensus,” in *Proc. Int. Symp. Nonlinear Theory and its Applications*, p. 10, Oct. 2009.

[22] S. Kar and J. Moura, “Sensor networks with random links: Topology design for distributed consensus,” *IEEE Trans. Signal Processing*, vol. 56, pp. 3315–3326, July 2008.

[23] S. Patterson, B. Bamieh, and A. El Abbadi, “Distributed average consensus with stochastic communication failures,” in *Proc. 46th IEEE Conf. Decision and Control*, (New Orleans, LA), pp. 4215–4220, Dec. 2007.

[24] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, “Energy-efficient, collision-free medium access control for wireless sensor networks,” *Wireless Networks*, vol. 12, no. 1, pp. 63–78, 2006.

[25] M. Haenggi, J. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti, “Stochastic geometry and random graphs for the analysis and design of wireless networks,” *IEEE J. Sel. Areas Comm.*, vol. 27, pp. 1029–1046, Sept. 2009.

[26] S. Vanka, M. Haenggi, and V. Gupta, “Convergence speed of the consensus algorithm with interference and sparse long-range connectivity,” *IEEE J. Sel. Topics Signal Processing*, vol. 5, pp. 855–865, Aug. 2011.

[27] R. Freeman, P. Yang, and K. Lynch, “Stability and convergence properties of dynamic average consensus estimators,” in *Proc. IEEE Conf. Decision and Control*, pp. 338 –343, 13-15 2006.

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

[29] M. Zhu and S. Martínez, “Discrete-time dynamic average consensus,” *Automatica*, vol. 46, issue 2, pp. 322–329, Feb. 2009.

[30] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, and M. Rupp, “Likelihood consensus: Principles and application to distributed particle filtering,” in *Proc. 44th Asilomar Conf. Signals, Systems, Computers*, (Pacific Grove, CA), Nov. 2010.

[31] O. Hlinka, O. Slučiak, F. Hlawatsch, P. M. Djurić, and M. Rupp, “Likelihood consensus and its application to distributed particle filtering,” *IEEE Trans. Signal Processing*, vol. 60, pp. 4334–4349, Aug. 2012.

[32] L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” in *Int. Conf. on Information Processing in Sensor Networks*, (Los Angeles, CA), pp. 63–70, April 2005.

[33] V. Schwarz and G. Matz, “Distributed reconstruction of time-varying spatial fields based on consensus propagation,” in *Proc. IEEE ICASSP-2010*, (Dallas, TX), pp. 2926–2929, March 2010.

[34] W. Ren, R. Beard, and E. Atkins, “Information consensus in multivehicle cooperative control,” *IEEE Control Systems Magazine*, vol. 27, pp. 71–82, April 2007.

[35] S. Sardellitti, M. Giona, and S. Barbarossa, “Fast distributed average consensus algorithms based on advection-diffusion processes,” *IEEE Trans. Signal Processing*, vol. 58, pp. 826–842, Feb. 2010.

[36] V. Schwarz, C. Novak, and G. Matz, "Broadcast-based dynamic consensus propagation in wireless sensor networks," in *Proc. 43th Asilomar Conf. Signals, Systems, Computers*, (Pacific Grove, CA), Nov. 2009.

[37] V. Schwarz and G. Matz, "System-theoretic formulation and analysis of dynamic consensus propagation," in *Proc. IEEE ICASSP-2011*, (Prague, CZ), pp. 3020–3023, May 2011.

[38] V. Schwarz and G. Matz, "Distributed averaging in wireless sensor networks under an ALOHA-like communication protocol," in *Proc. 44th Asilomar Conf. Signals, Systems, Computers*, (Pacific Grove, CA), Nov. 2010.

[39] N. Abramson, "THE ALOHA SYSTEM: another alternative for computer communications," in *Proc. Fall Joint Computer Conference*, pp. 281–285, ACM, Nov. 1970.

[40] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Review*, vol. 46, pp. 667–689, Dec. 2004.

[41] L. Xiao, S. Boyd, and S. Lall, "Distributed average consensus with time-varying metropolis weights," *submitted to Automatica*, 2006.

[42] V. Schwarz, G. Hannak, and G. Matz, "On the convergence of average consensus with generalized metropolis-hasting weights," in *Proc. IEEE ICASSP-2014*, (Florence, IT), May 2014.

[43] V. Schwarz and G. Matz, "Mean-square optimal weight design for average consensus," in *Proc. IEEE SPAWC-2012*, (Cesme, TR), pp. 374–378, June 2012.

[44] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Consensus in correlated random topologies: weights for finite time horizon," in *Proc. IEEE ICASSP-2010*, (Dallas, TX), pp. 2974–2977, March 2010.

[45] V. Schwarz and G. Matz, "Nonlinear average consensus based on weight morphing," in *Proc. IEEE ICASSP-2012*, (Kyoto, JP), pp. 3129–3132, March 2012.

[46] V. Schwarz and G. Matz, "Average consensus in wireless sensor networks: will it blend?," in *Proc. IEEE ICASSP-2013*, (Vancouver, BC), pp. 4584–4588, May 2013.

[47] A. C. Hindmarsh, P. M. Gresho, and D. F. Griffiths, "The stability of explicit Euler time-integration for certain finite difference approximations of the multi-dimensional advection-diffusion equation," *International Journal for Numerical Methods in Fluids*, vol. 4, no. 9, pp. 853–897, 1984.

[48] A. Sarwate and A. Dimakis, "The impact of mobility on gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 58, pp. 1731 –1742, March 2012.

[49] S.-Y. Tu and A. H. Sayed, "Mobile adaptive networks," *IEEE J. Sel. Topics Signal Processing*, vol. 5, pp. 649–664, Aug. 2011.

[50] V. Schwarz and G. Matz, "On the performance of average consensus in mobile wireless sensor networks," in *Proc. IEEE SPAWC-2013*, (Darmstadt, GER), pp. 175–179, June 2013.

[51] E. F. C. Andrea, A. Monti, F. Pasquale, and R. Silvestri, "Information spreading in stationary Markovian evolving graphs," in *Proc. IEEE Int. Symp. on Parallel Distributed Processing*, pp. 1–12, May 2009.

[52] J. Beutel, *Design and deployment of wireless networked embedded systems*. PhD thesis, Swiss Federal Institue of Technology Zurich, 2005.

[53] G. Reise, G. Matz, and K. Grochenig, "Distributed field reconstruction in wireless sensor networks based on hybrid shift-invariant spaces," *IEEE Trans. Signal Processing*, vol. 60, no. 10, pp. 5426–5439, 2012.

[54] H. Karl and A. Willig, *Protocols and architectures for wireless sensor networks*. Chichester, UK: Wiley, April 2005.

[55] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Comm.*, vol. 23, pp. 809 – 819, April 2005.

[56] R. Diestel, *Graphentheorie*. Springer, Sept. 2000.

[57] D. A. Spielman, "Spectral graph theory and its applications," in *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science FOCS '07*, pp. 29–38, 21–23 Oct. 2007.

[58] G. R. C. Godsil, *Algebraic Graph Theory*. Springer, 2001.

[59] D. Cvetković, "New theorems for signless Laplacian eigenvalues," *Bulletin T.CXXXVII de l'Académie serbe des sciences et des arts*, vol. 137, no. 33, pp. 131–146, 2008.

[60] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003.

[61] R. Albert and A.-L. Barabási, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, Oct. 1999.

[62] B. Bollobás, *Random Graphs*. Cambridge Univ Press, 2001.

[63] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, pp. 28–41, Jan. 2004.

[64] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Contr.*, vol. 49, pp. 1520–1533, Sept. 2004.

[65] L. C. Evans, *Partial differential equations*, vol. 19 of *Graduate Studies in Mathematics*. Oxford University Press, May 1998.

[66] W. Ames, *Numerical Methods for Partial Differential Equations*. Orlando, FL: Academic Press, 2nd ed., 1977.

[67] A. Quarteroni and A. Valli, *Numerical approximation of partial differential equations*, vol. 23 of *Springer series in computational mathematics*. Berlin: Springer, 1994.

[68] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*. Washington, DC: Hemisphere, 1980.

[69] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 3rd ed., 1991.

[70] F. Chung, "Lectures on spectral graph theory," *CBMS Lectures, Fresno*, 1996.

[71] R. Kindermann and J. Laurie, *Markov random fields and their applications*. American Mathematical Society, 1980.

[72] T. Kailath, *Linear Systems*. Englewood Cliffs (NJ): Prentice Hall, 1980.

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

[74] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs (NJ): Prentice Hall, 1993.

[75] T. Davidson, "Enriching the art of FIR filter design via convex optimization," *IEEE Signal Processing Magazine*, vol. 27, pp. 89 –101, May 2010.

[76] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming (web page and software), Stanford University, CA (http://stanford.edu/$\sim$boyd/cvx).

[77] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge (UK): Cambridge Univ. Press, 1999.

[78] X.-D. Zhang and R. Luo, "The spectral radius of triangle-free graphs," *Australasian Journal of Combinatorics*, vol. 26, pp. 33–40, Sept. 2002.

[79] M. Kaykobad, "Positive solutions of positive linear systems," *Linear Algebra and its Applications*, vol. 64, pp. 133–140, 1985.

[80] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge (UK): Cambridge Univ. Press, Dec. 2004.

[81] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hungar. Acad. Sci.*, vol. 5, pp. 17–61, 1960.

[82] J. G. Charney, R. Fjörtoft, and J. von Neumann, "Numerical integration of the barotropic vorticity equation," *Tellus*, vol. 2, no. 4, pp. 237–254, 1950/2010.