Graphical Model Driven Methods in Adaptive System Identification

by

Atulya Yellepeddi

B.Tech., Indian Institute of Technology, Roorkee (2010)
S.M., Massachusetts Institute of Technology (2012)

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

and the

WOODS HOLE OCEANOGRAPHIC INSTITUTION

September 2016

Author . . . .

Department of Electrical Engineering and Computer Science and
Joint Program in Applied Ocean Science and Engineering
June 24, 2016

Certified by . . . .

James C. Preisig
Scientist Emeritus, Woods Hole Oceanographic Institution
Thesis Supervisor

Accepted by . . . .

Leslie A. Kolodziejski
Chairman, EECS Department Committee on Graduate Theses

Accepted by . . . .

Henrik Schmidt
Chair, Joint Committee for Applied Ocean Science and Engineering

Graphical Model Driven Methods in Adaptive System Identification

by

Atulya Yellepeddi

Submitted to the Department of Electrical Engineering and Computer Science and
the Joint Program in Applied Ocean Science and Engineering
on June 24, 2016, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Identifying and tracking an unknown linear system from observations of its inputs and outputs is a problem at the heart of many different applications. Due to the complexity and rapid variability of modern systems, there is extensive interest in solving the problem with as little data and computation as possible.

This thesis introduces the novel approach of reducing problem dimension by exploiting statistical structure on the input. By modeling the input to the system of interest as a graph-structured random process, it is shown that a large parameter identification problem can be reduced into several smaller pieces, making the overall problem considerably simpler.

Algorithms that can leverage this property in order to either improve the performance or reduce the computational complexity of the estimation problem are developed. The first of these, termed the graphical expectation-maximization least squares (GEM-LS) algorithm, can utilize the reduced dimensional problems induced by the structure to improve the accuracy of the system identification problem in the low sample regime over conventional methods for linear learning with limited data, including regularized least squares methods.

Next, a relaxation of the GEM-LS algorithm termed the relaxed approximate graph structured least squares (RAGS-LS) algorithm is obtained that exploits structure to perform highly efficient estimation. The RAGS-LS algorithm is then recast into a recursive framework termed the relaxed approximate graph structured recursive least squares (RAGS-RLS) algorithm, which can be used to track time-varying linear systems with low complexity while achieving tracking performance comparable to much more computationally intensive methods.

The performance of the algorithms developed in the thesis in applications such as channel identification, echo cancellation and adaptive equalization demonstrate that the gains admitted by the graph framework are realizable in practice. The methods have wide applicability, and in particular show promise as the estimation and adaptation algorithms for a new breed of fast, accurate underwater acoustic modems.

The contributions of the thesis illustrate the power of graphical model structure in simplifying difficult learning problems, even when the target system is not directly structured.

Thesis Supervisor: James C. Preisig
Title: Scientist Emeritus, Woods Hole Oceanographic Institution

Acknowledgments

> Piglet noticed that even though he had a Very Small Heart, it could hold a rather large amount of Gratitude.
>
> *Winnie the Pooh*
> A.A.Milne

My years at MIT have been about so much more than a degree. In the past six years you have introduced to more worlds than I ever knew existed. As this long and wonderful journey ends, I find that my heart can, too, hold so much gratitude, for everything that you have given.

Of course, I haven't said who "you" is, so here goes.

It is impossible to write any list of acknowledgements without having at the very top my thesis advisor Dr. James Preisig, who has truly been a wide-angle lens for my research and life. There is much that Jim has shared with me about learning and research and I treasure every piece of advice he has ever given me. His way of drawing a straight line between theory and the real world, and of developing insight into hard problems, has had a deep influence on my own research style.

I'd also like to express my gratitude to Prof. Gregory Wornell, for giving me a research home at MIT, for teaching me to both learn and teach, and for all the advice and direction during the course of this work. Greg's approach to problem solving pervades this thesis, and, I hope, the rest of my professional life as well.

I am deeply indebted to my other thesis committee members, Prof. Arthur Baggeroer and Dr. Steven Smith, whose relentless pursuit of excellence has made this thesis so much richer. Their encyclopaedic knowledge and attention to detail has greatly improved the quality of this document.

My interactions with the members of the Multimedia, Interaction and Communication group at Microsoft Research, especially my mentor Dr. Dinei Florencio; and at the Communications Group at MERL, and in particular Dr. Phil Orlik, have made me a far better scientist and engineer. I will always be grateful for their support and perspective on my work.

My thanks are also due to Prof. Raj Nadakuditi at the University of Michigan, who hosted me when I visited UM and provided a valuable outside perspective to my work; and to Prof. Polina Golland, with whom I taught 6.008 at MIT, for, among many things, first teaching me about graphs and for ideas and wisdom shared over the years.

My colleagues at MIT and WHOI—especially my groupmates at the SIA group—who have shared this journey with me deserve a special mention. Particular thanks to Milutin, Gauri, Qing, Lisa, Sefa, Ananya, Beatrice, Ballard, Ying-Zhong, Gal, Ganesh and Da; for being willing participants in the exercise of being confused by their labmates' ideas!

All my friends within and outside MIT over the years have been instrumental in keeping me sane by being there to make endless cups of *chai* at 2:00 A.M., play table tennis at all hours of the day, pick up the phone in the middle of the night in India, get up and leave for hiking trips with no notice, and so many other things both little and large. My world has been a wonderful place because of these folks.

This work would not have been possible without the support of the wonderful staff at MIT and WHOI, especially Janet Fischer, Julia Westwater, Linda Cannata, Alicia Duarte, Leonara Fraser, Sheila Hurst, and most of all, Tricia O'Donnell.

The work in this thesis was supported primarily by the Office of Naval Research through an ONR Special Research Award in Ocean Acoustics; and at various times by the National Science Foundation, the WHOI Academic Programs Office and the MIT Presidential Fellowship Program. I am grateful to all of these agencies for their generous support.

And finally, after all is said and done, my deepest gratitude to my parents and sister for always being there to listen and remind me of what is important in life, my entire extended family for their undying support, and, most of all, to Rutu for more things than I am capable of naming. MIT has given me much over the years, but she is the greatest gift of all.

Contents

List of Figures

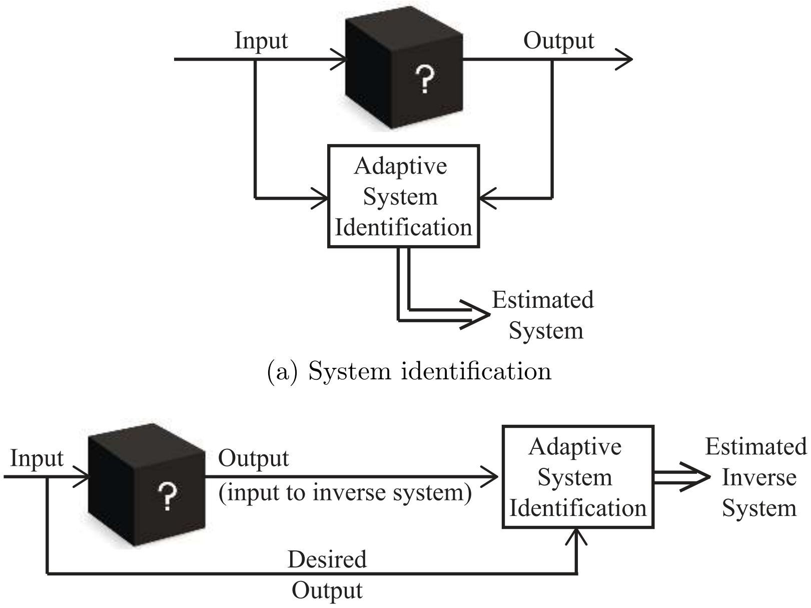
Chapter 1

Introduction

Imagine being presented with a mysterious system and being tasked with determining what it does. A good first step may be to plug an input into the system and observe what comes out. From repeated observations of the input and output, insight into the operation of the system can often be obtained.

In signal processing, the formalization of this concept is termed *adaptive system identification* [77], [78], [107]. The objective of adaptive system identification is to estimate an unknown linear system from observations of the inputs to the systems and the corresponding outputs (non-linear systems are sometimes considered as well, depending on the application domain). A conceptual diagram of such a learning problem is shown in Figure 1-1a. The adaptive system identification block makes an estimate of the black box system using the input and output of the system.

A modification of the problem to estimate an *inverse system*, i.e., a system that "undoes" the effect of the black box, is shown in Figure 1-1b. In this case, the system identification receives as an input the output of the unknown system, and the input to the unknown system—the output that the estimated system should ideally produce—serves as the desired output. A second important difference between the forward and inverse system identification problems is that for the inverse system identification problem, the system being identified is the inverse of the system that generates the input to the system identification algorithm. In other words, the system being identified (the black box) and the input signal to that system are not statistically independent. This effect is ignored throughout this work for simplicity.



(a) System identification

(b) Inverse system identification

Figure 1-1: Conceptual view of the adaptive system identification problem. Two kinds of problems are shown—the system identification problem and the so-called inverse system identification problem. Note that for the inverse system identification problem, the adaptive algorithm is attempting to estimate the inverse of the black-box. In other words, the black box is not the desired unknown system—rather its inverse is.

The forward system and inverse system estimation problem frameworks are used in different contexts, but the underlying question is similar—how to identify a system based on observations of its inputs and outputs. This framework can be applied in a multitude of domains, ranging from communication systems (identifying unknown channels and equalizers) to process systems (thermal, chemical systems, and so on), biology and medicine to social systems [7], [78], [167]. Different communities, from econometrics to machine learning, also have different subcultures and formulations of the same basic question, resulting in different viewpoints and approaches to tackling the problem [101].

The roots of system identification can be traced to works by Legendre and Gauss in the 19th century [159].[1] The modern statistical interpretation of the field dates approximately to the works of Fisher [60] and Kalman [83]. A detailed history of the field from these early

[1] The discovery of the method of least squares, one of the key algorithms in system identification by Adrien-Marie Legendre and Carl Friedrich Gauss lead to one of the more famous priority disputes in the history of statistics. An interesting documentation of this rather public quarrel was written by Plackett [138].

works through the early 2000s was summarized by Deistler [48].

In more recent years, signal processing applications have come to be characterized by the need to estimate ever larger systems, and by the dearth of observations with which to estimate them. In array and signal processing literature, this has sometimes been termed *snapshot deficient* or *rank deficient* processing [176]. It is intuitively obvious that this makes the learning problem more challenging—the less time one has to observe a system's effect, the worse the chances of understanding it.

An important generalization of the problem described above is obtained when the unknown black box of Figure 1-1 is allowed to vary with time. In this case, the adaptive system identification problem becomes one of *tracking* the unknown time-varying system of interest.

The term "adaptive system identification" is often used for both the problems of estimating a time-invariant system and tracking a time-varying system. However, the two problems do differ in some important respects. In particular, as data is received, the solution needs to be modified to incorporate new data, rather than being re-computed each time from scratch—a notion formalized by the terms *recursive*, *on-line* or *real-time* processing. Additionally, data needs to be weighted so as to forget the past. In this thesis, therefore a distinction is made between the problems of *identifying* or *estimating* a time-invariant system and *tracking* a time-varying system.

1.1 The Basic Question of this Thesis

The objective of this thesis is to investigate how algorithms for adaptive system identification and tracking may be improved by exploiting structure on the input to the system.

Common experience indicates that such improvements may be possible. The idea behind exploiting input structure is to use that structure to separate a large system identification problem into smaller pieces. This notion is quite intuitively appealing, as isolating the operation of separate parts of a system by, say, choosing inputs that operate on one part of the system at a time is a widespread practice when identifying the system.

As an informed reader will observe, the input to the linear system features prominently in most typical algorithms for linear system identification. The approach being considered herein is to judiciously choose those inputs to be structured in such a way as to simplify or reduce the dimension of the estimation problem when a limited amount of data is available.

In contrast to the strategy of exploiting a constrained input as considered in this work, the most common approach to system identification with limited data in the literature appears to be constraining the solution—in other words, placing some kinds of constraints on what form the estimated solution can take. For several applications, this is a reasonable approach that has led to a number of powerful algorithms.

Moreover, exploiting input constraints can be challenging. When a constrained input is passed through a system, the output is also constrained, albeit in a way that is potentially difficult to capture and exploit. When the effect of the constrained input on the output of the system is not considered, however, the estimation algorithm may not perform as expected.

Notwithstanding the arguments above, there are a variety of reasons to consider input structure as an alternative to system constraints as a way to simplify difficult system identification problems. Primary among these is that the input can be observed, whereas the underlying system can not. In other words, any assumptions made on how the input is structured can be verified. Indeed, in some applications, it is possible to control what the inputs to the system are—in such instances, one can simply choose inputs that satisfy the desired constraints. In contrast, assumptions made about the underlying system can not be verified, and while physical insight often guides the choice, the resulting estimate is only as good as the unverifiable constraint.

The above is a very high-level characterization of the problem being considered. A full survey of the literature concerning existing methods of constraining the solution space and an appreciation of the reasons for considering the input structure are better obtained once the problem is stated mathematically, and so these discussions are postponed to Chapter 2.

It is worth considering at this juncture what exactly is meant by "structure on the input data," and, in particular, what kind of structure is being considered. The kind of structure that is exploited in this work is statistical structure, which refers to a pattern of conditional independence statements amongst subvectors of the input. One particularly useful method to represent such structure is by using *probabilistic graphical models*.

The reason that the graphical model framework is useful is that it helps to divide the overall problem into subproblems. Conditionally independent parts of the input to the system give conditionally independent parts of the output due to the linearity of the system operation. This property implies that the structure on the entire set of observations, including both the inputs and outputs of the system, can be represented consistently within the probabilistic graphical model framework. This is what leads to the power of graphical models in this problem. It is also worth noting that the graphical model structure of interest is present in several real-world problems (see Chapter 3), making the advantages obtained from the framework widely applicable.

1.2 A Motivating Example: Adaptive Equalization in Underwater Acoustic Communication

Before proceeding, consider the following concrete example that illustrates the basic concept of the thesis that motivated the development of this work—namely, the problem of adaptive equalization in underwater acoustic communication. This problem will be revisited from a more mathematical perspective in Section 3.4.6, but a description to motivate the general ideas is provided here.

The underwater acoustic channel is one of the harshest communication channels commonly considered in signal processing. It is characterized by relatively long, time-varying multipath propagation that leads to significant intersymbol interference [38]. A particular realization of the time-varying impulse response of the channel collected from field data[2] is shown in Figure 1-2. The channel shows significant variation even over the relatively short time-frame (3 seconds) shown in the figure.

The objective of adaptive equalization is to filter the output of the channel to obtain an estimate of the transmitted signal by canceling the intersymbol interference. Adaptive equalization of the underwater communication channel is one of the most challenging aspects of communication across this channel, due to the length of the channel and rapid time-variation. This is one of the applications tackled in the thesis, and in Section 5.5.2 it is

[2] The field data is from the SPACE08 experiment—the experimental parameters will be explained in detail in Section 3.6

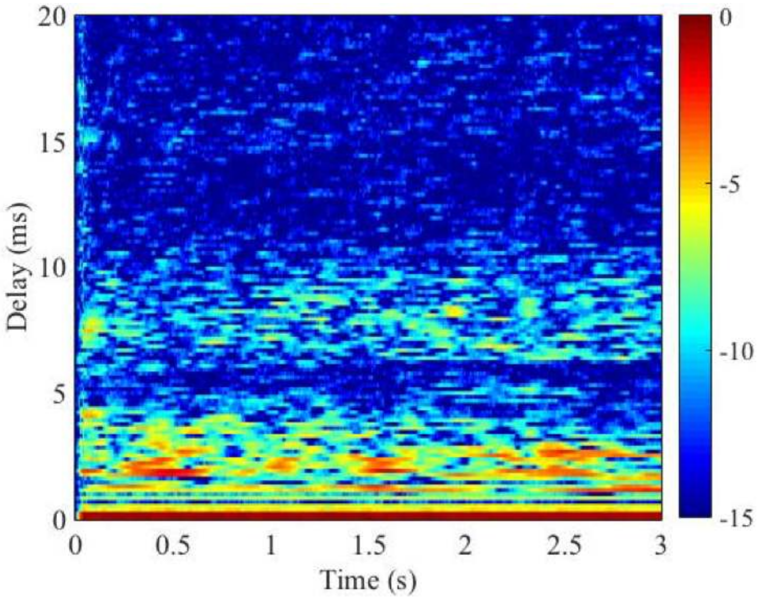


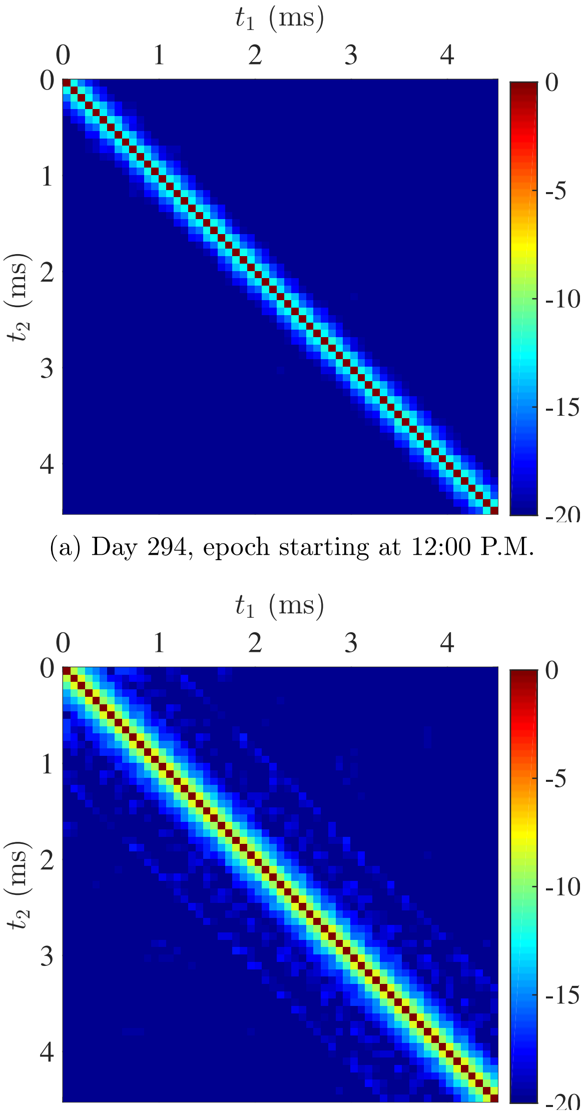
Figure 1-2: Example of the time-varying channel impulse response of the underwater acoustic communication channel. This particular realization is obtained from 200 meter range field data collected in the SPACE08 experiment conducted off Martha's Vineyard (Julian Day 300 at 8:00 AM). A more complete description of the experiment and data is provided in Section 3.6.

shown that the exploitation of graphical model input structure can considerably simplify the adaptive equalization problem.

The input to the equalizer is the output of the channel. As Section 1.1 implies, the algorithms presented in the thesis utilize structure on the input to the problem, which, in this case, would be graphical model structure on the output of the channel. It may not immediately be apparent that structure may be present in the output of a channel such as that of Figure 1-2.

As explained in Section 2.4.1, graphical model structure is intimately related to the sparse structure of the *inverse* covariance matrix. The importance of the thesis in the context of the adaptive equalization problem would thus be evident if it could be argued that the inverse covariance matrix of the channel output had predictable sparse structure.

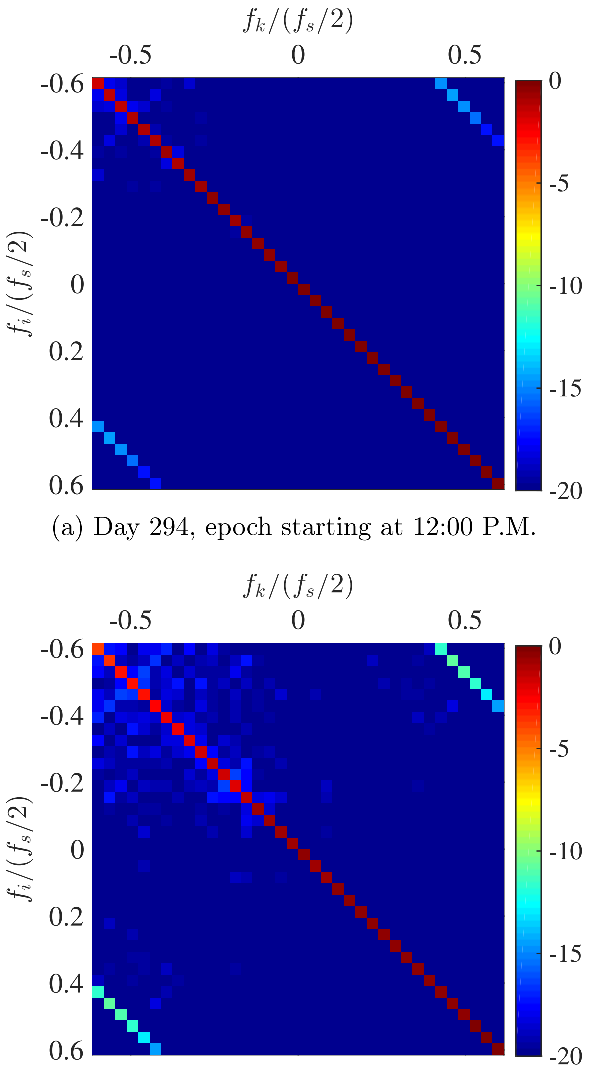
Figure 1-3 shows the inverse covariance matrices estimated from field data collected at the same location with the same experimental set-up at two different times. The exact days and times are indicated on the figures, but that is unimportant for the moment—it suffices to note



(a) Day 294, epoch starting at 12:00 P.M.

(b) Day 300, epoch starting at 8:00 A.M.

Figure 1-3: Inverse covariance matrices of the channel output at two different times (technically, epochs) of the SPACE08 experiments. The matrices are computed with an averaging window of 400 samples. The relative magnitudes of the elements are shown on a dB-scale. While there does appear to be some sparse structure, the structure is very different at the two different times.



(a) Day 294, epoch starting at 12:00 P.M.

(b) Day 300, epoch starting at 8:00 A.M.

Figure 1-4: Inverse covariance matrices of the frequency-domain channel output at two different times (technically, epochs) of the SPACE08 experiments. The matrices are computed with an averaging window of 400 samples. The relative magnitudes of the elements are shown on a dB-scale. The frequency-domain structure is predictable across environmental conditions.

that the environmental conditions are very different.[3] The absolute values of the elements are shown on a dB-scale referenced to the largest element of the respective matrices. While some structure does appear to exist in the matrices, the structure is dramatically different for the two different environmental conditions. If this structure had to be exploited, the wide variation of the structure with environmental conditions would be very challenging to handle, as it requires modeling, sensing and tracking the environment, which is beyond the capabilities of most real-time systems for this application.

However, as shown in Figure 1-4, converting the channel output into the frequency-domain reveals an inverse covariance structure that is not only much sparser the corresponding time-domain structure, but also much more predictable in that it shows less variation with environmental conditions. A theoretical proof that the output signals of a time-varying channel can have graphical model structure is presented in Section 3.4.5. For now, a useful insight from Figure 1-3 and Figure 1-4 is that channels with vastly different environmental characteristics can have very similar statistical properties on a suitable time-scale. Graphical models serve as a way to capture those properties in a manner useful to inference, learning and tracking.

Sections 3.4.5 and 3.6 will show how to represent the statistical properties of the output of an underwater acoustic communication channel using a graphical model; and the algorithms in Chapters 4 and 5 will show how to exploit the structure. It is emphasized that the bulk of the algorithm development and analysis in the thesis is done without reference to any particular application, although the algorithms are subsequently applied to several applications. While adaptive equalization is one of the key applications considered in the thesis, it is not the only domain in which the methods of the thesis are application, and so the development is kept as general as possible.

A chapter-by-chapter overview of the thesis is now presented.

[3] Specifically, day 294 corresponds to low wind conditions, whereas day 300 had high wind conditions. For more details, see Figure 5-5.

1.3 Thesis Overview

In this chapter, the problem of adaptive system identification with input structure has been described at a non-technical level and a motivating example presented.

In Chapter 2, a more detailed explanation of the problem is presented and the approach taken in this work is placed in the context of existing work in the field of system identification. The notion of probabilistic graphical models is introduced. Various properties of the models that are useful to the development of the algorithms are also described.

Then, in Chapter 3, it is shown that a graphical model description exists for the input data in three applications: acoustic echo cancellation, channel identification and adaptive equalization. Having explained the structure under consideration and shown that it characterizes some interesting applications, algorithm development is considered.

In Chapter 4, a detailed graphical model framework for the adaptive system identification problem with input graphical model structure is presented. It is shown that the framework leads to an iterative algorithm, termed the *graphical expectation-maximization least squares* (GEM-LS) algorithm, whose properties are then extensively analyzed and verified using simulation. The algorithm is then applied to the problems of channel identification in communication systems and initialization of acoustic echo cancellation filters and is shown to outperform various state-of-the-art methods in regimes with limited data and large noise power. In particular, considerable performance improvements are demonstrated over ℓ_2 and ℓ_1 regularized least squares methods in regimes with very limited data.

While the GEM-LS algorithm is capable of effecting performance improvements, it is alternatively possible to exploit the graphical model to reduce the computation required to solve the linear system identification problem. To that end, a relaxation of the GEM-LS algorithm, termed the *relaxed approximate graph-structured least squares* (RAGS-LS) algorithm, is presented and analyzed in Chapter 5. The RAGS-LS algorithm can be very computationally efficient but at the expense of performance. Nonetheless, its performance can come close to that of conventional LS at very low Signal-to-Noise Ratios (SNRs), or when very little data is available. It is shown that the reason for this behavior is that the algorithm trades off solving very small problems for a large amount of effective noise in each of those problems.

In a recursive context, it is possible to mitigate the effective noise problem. Section 5.4 introduces *relaxed approximate graph-structured recursive least squares* (RAGS-RLS), which is useful in tracking time varying systems very efficiently with little loss of performance over conventional RLS. Two practical applications of the RAGS-RLS algorithm are then considered: tracking the coefficients of acoustic echo cancellation filters, and tracking adaptive equalizer coefficients for underwater acoustic communication systems. These applications involve tracking large numbers of coefficients, and in both cases, it is shown that the RAGS-RLS algorithm is as good or better than algorithms conventionally used for the applications with considerably lower complexity.

The thesis thus demonstrates various opportunities for exploiting input structure to improve the performance and reduce the complexity of adaptive system identification. A summary of the contributions of the thesis and recommendations for future work are contained in Chapter 6.

1.4 Common Notation

Notation that is used throughout the thesis is established in Figure 1-5.

Notation	**Represents**	**Comments**
Boldface math $\boldsymbol{x}$	Vectors, Matrices	No distinction made between vectors and matrices in notation, difference should be inferred from context.
x^*	Complex conjugate of scalar x	
$\boldsymbol{X}^\dagger$	Hermitian (complex conjugate transpose) of vector/matrix $\boldsymbol{X}$	
$\boldsymbol{I}_d$	$d \times d$ Identity matrix	
Sans-serif math font x	Random variables	Random vectors/matrices are represented with boldface Sans Serif fonts, such as $\boldsymbol{\mathsf{x}}$.
$p_{\boldsymbol{\mathsf{x}}}(\boldsymbol{x})$	Probability density function of $\boldsymbol{\mathsf{x}}$ evaluated at $\boldsymbol{x}$	
$p_{\boldsymbol{\mathsf{x}}\mid\boldsymbol{\mathsf{y}}}(\boldsymbol{x} \mid \boldsymbol{y})$	Probability density function of $\boldsymbol{\mathsf{x}}$ evaluated at $\boldsymbol{x}$ conditioned upon $\boldsymbol{\mathsf{y}} = \boldsymbol{y}$	
$p_{\boldsymbol{\mathsf{x}}}(\boldsymbol{x}\ ;\ \boldsymbol{h})$	Probability density function of $\boldsymbol{\mathsf{x}}$ evaluated at $\boldsymbol{x}$ parameterized by $\boldsymbol{h}$	$\boldsymbol{h}$ is a *deterministic* quantity, so this kind of parameterization is distinct from conditioning, although the effect is quite similar. It does not make sense in this context to apply Bayes' theorem, or even to define $p_{\boldsymbol{h}}(\boldsymbol{h})$.
$\mathsf{v} \sim p_{\mathsf{v}}(v)$	Random variable v has distribution $p_{\mathsf{v}}(v)$	
$\mathsf{v}(1), \mathsf{v}(2), \ldots \overset{\text{iid}}{\sim} p_{\mathsf{v}}(v)$	Random variables $\mathsf{v}(1), \mathsf{v}(2), \ldots$ are independent and identically distributed according to $p_{\mathsf{v}}(v)$	
$\mathcal{CN}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$	Circularly symmetric complex Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$	All complex Gaussian random variables in this work are assumed to be circularly symmetric unless otherwise stated.
$\mathbb{E}\left[\cdot\right]$	Expected Value	
$\mathsf{x} \perp\!\!\!\perp \mathsf{y}$	Random variables x and y are independent	
$\boldsymbol{X}_{a:b,c:d}$	Sub-matrix of $\boldsymbol{X}$ between rows a through b and columns c through d	Similarly for vectors
$\boldsymbol{X}_{:,c:d}$	Sub-matrix formed by all rows and columns c through d of $\boldsymbol{X}$	

Figure 1-5: Notation used throughout the thesis

Chapter 2

System Identification Survey

The goals of this chapter are two-fold. First, a detailed explanation is given of the various problems that are considered in the rest of this work. Second, along with the description of the problems and goals of the thesis, a detailed survey of existing techniques and prior work is provided, and methods prevalent in the literature are compared and contrasted to the approaches considered here.

The mathematical model for the time-invariant system identification problem with limited data is presented in Section 2.1. Then a brief survey of existing methods for solving these problems is undertaken. These methods essentially involve regularization. Various regularizers used in practice are described in Section 2.1.4. Some drawbacks of regularization and the need to consider alternate approaches are explained, and the notion of input structure as an alternate approach is introduced. In Section 2.2.1, the system model for the more involved system tracking problem is explained. The set of problems that are of interest in system tracking are described.

Section 2.3 explains what is meant by input structure, and an intuitive explanation of why it is expected to be helpful. It is explained why, in particular, graphical model input structure is of interest in the problems considered in this thesis. Finally, Section 2.4 reviews the concepts of graphical models with particular reference to ideas important to this work.

2.1 Linear Time-Invariant System Identification

As previously explained, the problem of linear time-invariant (LTI) system identification is one of the fundamental challenges of signal processing. Mathematically, this is the problem of estimating a system $\boldsymbol{h}_0$ from observations of the inputs $\boldsymbol{x}(n)$ and the corresponding outputs $y(n)$, where

$$y(n) = \boldsymbol{h}_0^\dagger \boldsymbol{x}(n) + v(n),\ n = 1, 2, \ldots, N\,. \tag{2.1}$$

In the above. $v(n)$ represents additive noise in the system at time n and $\boldsymbol{h}_0$ is the system to be identified. The dimension of $\boldsymbol{h}_0$ is denoted M, and the number of data samples available is N. The relationship between M and N will be important in this work.

A large number of problems fit into this framework. Examples abound in communication [142], room acoustics [187], underwater communication [38], systems biology [95], seismic tomography [7], quantitative spectroscopy [65] and many others [167]. With such a vast variety of possible applications, the space of algorithms and solutions is also huge. However, one of the most fundamental algorithms continues to be one of the most popular—the least squares algorithm [78], which is discussed next.

2.1.1 The Least Squares Algorithm

The *least squares* (LS) solution of (2.1) is the value of $\boldsymbol{h}$ that minimizes the "sum of squared prediction error" cost criterion, i.e.,

$$\hat{\boldsymbol{h}}_{\mathrm{ls}} = \arg\min_{\boldsymbol{h} \in \mathbb{C}^M} \sum_{n=1}^{N} \left| y(n) - \boldsymbol{h}^\dagger \boldsymbol{x}(n) \right|^2\,. \tag{2.2}$$

For $N \geq M$, the closed form solution to the above is given by

$$\hat{\boldsymbol{h}}_{\mathrm{ls}} = \left(\sum_{n=1}^{N} \boldsymbol{x}(n) \boldsymbol{x}^\dagger(n) \right)^{-1} \left(\sum_{n=1}^{N} \boldsymbol{x}(n) y^*(n) \right) \tag{2.3}$$

The LS solution is widely used due to its simplicity, robustness and excellent performance in practical applications [78].

2.1.2 Statistical Interpretation of Least Squares

In the above description of the problem, nothing has been said about the way the input signal or the noise have been generated. While it is not necessary to make any assumptions about these to obtain an LS solution—the cost criterion of (2.2) is reasonable for many classes of signal and noise statistics—with suitable assumptions, the LS estimate can be interpreted in some statistically useful ways, which provide interesting insight into its performance. Indeed, the statistical interpretation may provide some insight into *why* the cost criterion is a reasonable one to begin with.

The assumption is that $\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots$ are independent realizations of a multivariate complex random vector $\boldsymbol{\mathsf{x}}$, where $\boldsymbol{\mathsf{x}} \sim \mathcal{CN}\left(0\,,\, \boldsymbol{R}_{\boldsymbol{\mathsf{x}}}\right)$. Additionally, assume that the noise realizations $v(1), v(2), \ldots$ are independent realizations of the random variable v, where $\mathsf{v} \sim \mathcal{CN}\left(0\,,\, \sigma^2\right)$, i.e., the noise is white circularly symmetric complex Gaussian noise with variance σ^2.

Observe that, under these assumptions, the outputs $y(n)$ can be modeled as independent realizations of a random variable y, where the marginal distribution of y and conditional distribution of y conditioned upon $\boldsymbol{\mathsf{x}}$ are given by

$$p_{\mathsf{y}}(y(n)\ ;\ \boldsymbol{h}_0) = \mathcal{CN}\left(0\,,\, \boldsymbol{h}_0^{\dagger}\boldsymbol{R}_{\boldsymbol{\mathsf{x}}}\boldsymbol{h}_0 + \sigma^2\right) \tag{2.4a}$$

$$p_{\mathsf{y}|\boldsymbol{\mathsf{x}}}(y(n) \mid \boldsymbol{x}(n)\ ;\ \boldsymbol{h}_0) = \mathcal{CN}\left(\boldsymbol{h}_0^{\dagger}\boldsymbol{x}(n)\,,\, \sigma^2\right) \tag{2.4b}$$

In other words, it is assumed that the random variables $\boldsymbol{\mathsf{x}}$ and y are related by

$$\mathsf{y} = \boldsymbol{h}_0^{\dagger}\boldsymbol{\mathsf{x}} + \mathsf{v}\,, \tag{2.5}$$

and N independent observations of $\boldsymbol{\mathsf{x}}$ and the corresponding realizations of y are available to estimate $\boldsymbol{h}_0$, which is modeled as a *deterministic* parameter of the joint distribution of $(\boldsymbol{\mathsf{x}}, \mathsf{y})$. In other words, the set up is that of a *non-Bayesian* parameter estimation problem. Non-Bayesian means that no a-priori statistical model is imposed upon $\boldsymbol{h}_0$.

In this framework, the LS estimate is the maximum-likelihood (ML) estimate of the parameter $\boldsymbol{h}_0$. To see this, note that, by the definition of the ML estimate,

$$\begin{aligned}
\hat{\boldsymbol{h}}_{\mathrm{ml}} &= \arg\max_{\boldsymbol{h}\in\mathbb{C}^M} \prod_{n=1}^{N} p_{\boldsymbol{x},y}(\boldsymbol{x}(n), y(n)\ ;\ \boldsymbol{h}) \\
&= \arg\max_{\boldsymbol{h}\in\mathbb{C}^M} \prod_{n=1}^{N} p_{y|\boldsymbol{x}}(y(n) \mid \boldsymbol{x}(n)\ ;\ \boldsymbol{h}) && \text{(2.6a)} \\
&= \arg\max_{\boldsymbol{h}\in\mathbb{C}^M} \sum_{n=1}^{N} \log p_{y|\boldsymbol{x}}(y(n) \mid \boldsymbol{x}(n)\ ;\ \boldsymbol{h}) && \text{(2.6b)} \\
&= \arg\min_{\boldsymbol{h}\in\mathbb{C}^M} \sum_{n=1}^{N} \left|y(n) - \boldsymbol{h}^{\dagger}\boldsymbol{x}(n)\right|^2 \\
&= \hat{\boldsymbol{h}}_{\mathrm{ls}}\,. && \text{(2.6c)}
\end{aligned}$$

In the above, (2.6a) arises becauses $p_{\boldsymbol{x}}(\cdot)$ does not depend upon $\boldsymbol{h}$, i.e., the input is generated without regard to the system, (2.6b) due to the monotonicity of the log function and the rest is algebra.

As mentioned in Chapter 1, the assumption that the input distribution is independent of the system being identified that is made above is not valid for inverse problems. In that case, the input is generated by a system whose inverse is being estimated. To the best of the author's knowledge, models and analysis that incorporate the effects of dependence of the system on the input are relatively rare. Such modeling is outside the scope of this work—it is henceforth assumed that the input to the system is independent of the system.

ML estimators in general have a number of useful properties in the regime $N \to \infty$. For instance, they are *consistent*, meaning that

$$\lim_{N\to\infty} \hat{\boldsymbol{h}}_{\mathrm{ml}} = \lim_{N\to\infty} \hat{\boldsymbol{h}}_{\mathrm{ls}} = \boldsymbol{h}_0\,. \tag{2.7}$$

Additionally, the distribution of the ML estimate is asymptotically normal, and the estimator itself is asymptotically efficient, meaning that it is Cramér-Rao bound achieving as $N \to \infty$ [126].

However, while ML seems like a good choice of solution given the right statistical model, LS is only an ML solution for the Gaussian model. Therefore, the question may be, why should Gaussianity be a good assumption?

In practice, Central Limit Theorem arguments can often be used to justify Gaussianity as a good noise model, as physics often dictates that there are many sources of noise and their sum leads to Gaussian-distributed noise. However, there is an information theoretic justification for the model that does not require such a physical interpretation.

In particular, it has been shown [125], [168] that the differential entropy of a zero-mean complex-valued random vector with a fixed covariance matrix is upper bounded by that of a circularly symmetric Gaussian random vector with the same covariance matrix. In other words, the circularly symmetric Gaussian random vector is the maximum entropy distribution on $\mathbb{C}^M$ with a fixed covariance matrix.

This justifies the choice of input data model made. Assuming that the data covariance matrix is fixed, the maximum entropy distribution is a reasonable choice of model for the data [90] because from an information-theoretic point of view, this is the distribution that maximizes "randomness" and makes the fewest assumptions about the data. Moreover, for a Gaussian-distributed input vector, and for a fixed noise variance, the noise model that *minimizes* the mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$ is the Gaussian noise model [133].

Put a different way, suppose that the second moments of the signal and noise are fixed, and nature gets to choose the *worst* distributions for the input and noise, i.e., the distributions that *maximize* the entropy of the input and *minimize* the mutual information between the input and output of the system, respectively. Then the solution to that game, from nature's point of view is to choose a multivariate Gaussian input and Gaussian noise.

Thus, least squares computes the maximum likelihood estimate of the parameter that corresponds to the *worst* possible model for the data and noise from an information theory viewpoint. At an intuitive level (although no attempt is made here to make this statement mathematically precise), this can viewed as a minimax estimate. The minimax approach is known in general to be very robust [177], which provides insight into why LS performs so well in a wide variety of settings.

With the statistical model imposed upon the problem, the LS solution can be expressed in terms of the empirical statistics of the input and output. Specifically, we may write:

$$\hat{\boldsymbol{h}}_{\text{ls}} = \hat{\boldsymbol{R}}_{\boldsymbol{\mathsf{x}}}^{-1}(N)\hat{\boldsymbol{r}}_{\boldsymbol{\mathsf{x}}\mathsf{y}}(N)\,, \tag{2.8}$$

where

$$\hat{\boldsymbol{R}}_{\boldsymbol{\mathsf{x}}}(N) = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)\boldsymbol{x}^{\dagger}(n) \tag{2.9a}$$

is the sample (or empirical) covariance matrix (SCM) of the random variable $\boldsymbol{\mathsf{x}}$ and

$$\hat{\boldsymbol{r}}_{\boldsymbol{\mathsf{x}}\mathsf{y}}(N) = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)y^{*}(n) \tag{2.9b}$$

is the sample cross-correlation between the random variables $\boldsymbol{\mathsf{x}}$ and y that form the input and output of the system. Note that (2.9a) and (2.9b) are themselves the ML estimates of the covariance matrix of $\boldsymbol{\mathsf{x}}$ (denoted $\boldsymbol{R}_{\boldsymbol{\mathsf{x}}}$) and the cross correlation between $\boldsymbol{\mathsf{x}}$ and y (denoted $\boldsymbol{r}_{\boldsymbol{\mathsf{x}}\mathsf{y}}$), respectively.

In the rest of this work, unless otherwise stated, it is assumed that $(\boldsymbol{x}(n), y(n))$ correspond to independent realizations of the probabilistic model of (2.5), and the estimation problem is treated as the problem of identifying the parameters of a distribution.

2.1.3 The Sample Limited Regime

Maximum-likelihood estimation is a highly intuitively appealing concept, which had led to the persistent belief among some statisticians that an ML estimator is an innately good one. Such a belief was shattered by the results of James and Stein [56], [89], [158] that one could always, on average, improve the MSE of the estimator of the mean of a multivariate Gaussian distribution by "shrinking" the estimator towards any fixed point. The results of James and Stein have been extended to linear system learning problems [151]—shrinking the LS solution towards a fixed point can improve the MSE on average.

Of particular interest to the work of this thesis is that, for finite sample sizes, nothing can be said about how tightly the distribution of the ML solution concentrates about the true parameter in comparison to other estimators [136]. In other words, the ML solution may not be close to the true parameter under a given distance metric for a finite amount of data N. The consistency of the ML estimate of (2.7) is a strictly asymptotic result and can not be extended in general.

Specializing to the case of the linear parameter identification problem under consideration, the performance of the LS solution suffers when $N \sim M$. In highly sample limited regimes, the LS solution does not typically result in a good estimate. An extreme case of this is easily observed in the so-called *rank-deficient* regime [176], where $N < M$. In this case, there is no unique LS solution. Rather, every solution to the set of linear equations

$$\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\hat{\boldsymbol{h}} = \hat{\boldsymbol{r}}_{\boldsymbol{x}\boldsymbol{y}}(N) \tag{2.10}$$

is an LS solution. It can be verified that every such solution has the exact same likelihood, i.e., that $\prod_{n=1}^{N} p_{\boldsymbol{x},y}(\boldsymbol{x}(n), y(n)\ ;\ \hat{\boldsymbol{h}})$ is identical for every $\hat{\boldsymbol{h}}$ that satisfies (2.10). As $\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)$ has rank N and $\boldsymbol{h}$ is M-dimensional, the space of solutions is an $N - M$ dimensional hyperplane. Evidently, some of the solutions (points in the hyperplane) will be closer to the true vector $\boldsymbol{h}_0$ than others (in a Euclidean sense, say).

Even when $N \geq M$ and the LS solution is unique, it is not generally good until $N \gg M$. To make this more concrete, define

$$\alpha = \frac{M}{N}\,. \tag{2.11}$$

When $\alpha \approx 0$, unlimited data is available and the estimators are operating in the asymptotic regime. As the number of observations decreases (and/or the system dimension increases), α increases towards 1, and the regime defined in this work as the *sample limited* regime is entered.[1]$\alpha > 1$ is termed the *snapshot deficient* regime.

While it has been long understood that modifications to the LS solution can improve performance with limited data (see Section 2.1.4), a theoretical characterization of sample limited adaptive processing generally involves the use of Random Matrix Theory methods [43] and is relatively recent (e.g., [2], [120], [121]).

In particular, the work of Pajovic and Preisig [129]–[131] shows mathematically that the

[1] The conventional term in array processing is snapshot limited [25], [139]; but sample limited or realization limited seems more appropriate in the context of this work, as "snapshot" is a term most commonly applied to arrays.

performance of an LS estimator is not expected to be good in the sample limited regime. Specifically, it is shown that, when $\alpha = M/N \leq 1$, i.e., when $M \leq N$,[2]

$$\begin{aligned} \boldsymbol{R}_{\hat{\boldsymbol{h}}_{\mathrm{ls}}} &= \mathbb{E}\left[(\boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\mathrm{ls}})(\boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\mathrm{ls}})^\dagger\right] \\ &= \frac{\sigma^2}{N(1-\alpha)} \boldsymbol{J}_{\boldsymbol{x}} \,, \end{aligned} \tag{2.12a}$$

where $\boldsymbol{J}_{\boldsymbol{x}} = \boldsymbol{R}_{\boldsymbol{x}}^{-1}$ is the *inverse covariance matrix* of $\boldsymbol{x}$. From this, it can be seen that, in this regime, the MSE of the LS solution is given by

$$\mathbb{E}\left[\left\|\boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\mathrm{ls}}\right\|^2\right] = \frac{\sigma^2}{N(1-\alpha)} \mathrm{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\} \,, \tag{2.12b}$$

which is to say it increases without bound as N approaches M.

As an aside, a related result in the context of array processing is the classical Reed-Mallett-Brennan result [143] that the number of snapshots required for adaptive array processing is at least 2 to 3 times the number of array elements. This result also implies (albeit in a somewhat different context) that the data dimension should be significantly larger than the problem dimension for good performance in adaptive estimation.

However, modern signal processing is full of examples of massive linear systems that need to be estimated from relatively small amounts of data. For instance, adaptive processing of data from arrays with thousands of elements [150] or estimation of long channels such as Ultra-Wide-Band channels [117] and wireless acoustic communication (AComms) channels [79] are problems that can be cast into this framework.

Various methods can be employed to handle limited sample availability, including the use of application-specific physical constraints, e.g., [97], [132]. However, the most widespread general purpose methods are *regularization*-based. The concept of regularization is now discussed.

[2] For $N < M$, of course, LS does not have a unique solution, so it is not possible to define a specific error

2.1.4 Regularization to Handle Limited Samples

In general, the concept of *regularization* is the process of adding some constraining information to a cost criterion in the form of a regularization function. For the LS cost criterion of (2.2), the solution then takes the form

$$\hat{\boldsymbol{h}}_{\mathrm{reg}} = \underset{\boldsymbol{h}\in\mathbb{C}^M}{\arg\min} \sum_{n=1}^{N} \left|y(n) - \boldsymbol{h}^{\dagger}\boldsymbol{x}(n)\right|^2 + \delta_r r(\boldsymbol{h})\,. \tag{2.13}$$

In the above, $\delta_r \geq 0$ controls the importance of the regularization function $r(\boldsymbol{h})$ [21]. The larger the value of δ_r, the more penalty is applied to values of $\boldsymbol{h}$ that make $r(\boldsymbol{h})$ large.

Typically, for linear parameter identification problems, the regularization function $r(\boldsymbol{h})$ takes the form of a norm of $\boldsymbol{h}$, i.e., $r(\boldsymbol{h}) = \|\boldsymbol{h}\|_p^p$ (the norm is raised to the pth power for dimensional consistency). This means that the resulting algorithm looks for solutions that have the smallest p-norm. Not every value of p results in a useful algorithm, however. Of particular interest are $p = 2$ and $p = 1$.

Tikhonov Regularization

A classical method for handling ill-posed problems, Tikhonov regularization [84], [171] penalizes the squared ℓ_2-norm of the solution. In other words, the Tikhonov-regularized least squares solution is given by

$$\hat{\boldsymbol{h}}_{\mathrm{dl}} = \underset{\boldsymbol{h}\in\mathbb{C}^M}{\arg\min} \sum_{n=1}^{N} \left|y(n) - \boldsymbol{h}^{\dagger}\boldsymbol{x}(n)\right|^2 + \delta_2 \left\|\boldsymbol{h}\right\|_2^2\,. \tag{2.14}$$

In the above the subscript dl stands for diagonal loading—the reason for the nomenclature will be evident momentarily.

A closed form solution to (2.14) exists and is given by

$$\hat{\boldsymbol{h}}_{\mathrm{dl}} = \left(\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) + \delta_2 \boldsymbol{I}_M\right)^{-1} \hat{\boldsymbol{r}}_{\boldsymbol{x}\boldsymbol{y}}(N)\,. \tag{2.15}$$

This is simply the LS solution, but with the addition of a scaled identity matrix to the sample covariance matrix of $\boldsymbol{x}$ that makes the matrix being inverted a full-rank matrix. In other words, the diagonal of the SCM is “loaded” with $\delta \boldsymbol{I}_M$, which explains the nomenclature above. A minor note is that Tikhonov regularization, in general, has a penalization factor given by $r(\boldsymbol{h}) = \|\boldsymbol{\Gamma}\boldsymbol{h}\|_2^2$ for some Tikhonov matrix $\boldsymbol{\Gamma}$, which is conventionally chosen to be $\boldsymbol{\Gamma} = \delta \boldsymbol{I}_M$.

One question at this juncture may be to understand why the ℓ_2-norm penalization works. One way to understand this is through the previously introduced notion of shrinkage [41]. By penalizing the ℓ_2-norm, diagonal loading implicitly shrinks the solution towards $\boldsymbol{0}$ in a spherically symmetric manner; by which it is meant that the diagonal loading penalizes all vectors at equal distances from the origin by the same amount—a property that is easily seen from the form of the regularizer; thus reducing the variance of the estimate. The reductions in variance for the finite sample case have been computed using Random Matrix Theory [129], [130].

Viewed a different way, the ℓ_2-norm of the system estimate is equivalent to its so-called white noise enhancement [47], i.e., a measure of how much it amplifies white noise. This is easy to see, because when white noise of unit-power is the input to the linear system $\hat{\boldsymbol{h}}$, the output is white noise with power $\|\hat{\boldsymbol{h}}\|^2$. Historically, this is a measure of the robustness of an estimator. It will be recognized that this is the time-series analogue of the white noise gain of an array [176].

The larger the ℓ_2-norm of a system, therefore, the more it amplifies white noise, and as a consequence, the less robust it is (as mismatches manifest as noise). By penalizing the ℓ_2-norm of the solution, diagonally loaded LS results in a highly robust estimator of the linear system.

Another interpretation of diagonal loading is that if $\boldsymbol{h}_0$ is treated as a realization of a random variable $\boldsymbol{h}_0$, then diagonally loaded LS is the MAP estimate of $\boldsymbol{h}_0$ assuming a Gaussian prior,$\boldsymbol{h}_0 \sim \mathcal{CN}\left(\boldsymbol{0}\,,\,\delta_2^{-1}\boldsymbol{I}_M\right)$. Thus, in effect, diagonal loading is expressing an a-priori belief that the complex vector of interest is a zero-mean complex Gaussian with spherical symmetry about the origin.

Subset Selection and LASSO Regression

As an alternative to the ℓ_2 regularization, a different strategy is to penalize the ℓ_1-norm of the estimate, a strategy that leads to the least absolute shrinkage and selection operator (LASSO) regression. The solution to the (penalized form) of the LASSO regression cost criterion is written as

$$\hat{\boldsymbol{h}}_{\text{lasso}} = \underset{\boldsymbol{h}\in\mathbb{C}^M}{\arg\min} \sum_{n=1}^{N} \left| y(n) - \boldsymbol{h}^\dagger \boldsymbol{x}(n) \right|^2 + \delta_1 \left\| \boldsymbol{h} \right\|_1 . \tag{2.16}$$

LASSO regression is often credited to Chen or Tibshirani [34], [35], [169], although the technique likely predates these works.

The ℓ_1-penalty is often motivated as a convex relaxation of so-called *subset selection*, which is, in essence, an ℓ_0-penalty on the basic LS cost criterion, i.e.,

$$\hat{\boldsymbol{h}}_{\text{subset-selection}} = \underset{\boldsymbol{h}\in\mathbb{C}^M}{\arg\min} \sum_{n=1}^{N} \left| y(n) - \boldsymbol{h}^\dagger \boldsymbol{x}(n) \right|^2 + \delta_0 \left\| \boldsymbol{h} \right\|_0 . \tag{2.17}$$

The ℓ_0-norm is the number of non-zero elements in the vector. In other words, ℓ_0-norm penalized LS corresponds to simultaneous variable selection and estimation—it finds solutions which have the fewest non-zero entries. While the theoretical performance of subset selection is known to be good in terms of risk inflation [64], the problem of (2.17) is non-convex and NP-hard, making it impractical to solve.

The LASSO regression of (2.16), on the other hand, is convex. Much like ℓ_2 regularization, the ℓ_1 regularization causes shrinkage, thereby reducing the variance of the estimate. However, it shrinks the solution in such a way as to encourage sparsity,[3] similar to the ℓ_0-solution [28], [76]; additionally, because the problem is convex, a unique solution exists.

Moreover, a variety of convergence properties and other useful properties of the LASSO regression can be proved, including consistency and compatibility results with some conditions on the ℓ_1 norm of the true vector $\boldsymbol{h}_0$ ([26], [170] and references therein). Thus, in applications where the underlying parameter vector is sparse, the ℓ_1-regularized LS has been

[3] The usual geometric interpretation is that while diagonal loading causes shrinkage towards spheres centered at the origin, LASSO causes shrinkage along hypercubes, which tends to lead to sparse solutions.

shown to be a powerful technique. It is therefore widespread as a technique for reconstruction in compressed sensing and sampling [29], [162].

A Bayesian interpretation is also possible for ℓ_0 and ℓ_1 regularization. LASSO can be interpreted as Bayesian regression with a Laplace prior on the unknown vector [55], [169], whereas ℓ_0 regularization is approximately equivalent to assuming a Cauchy prior on the unknown system [93].

In recent years, the field has seen a flurry of research involving the use of sparse solution techniques in fields as diverse as underwater communication and networks [18], [116], [153], room acoustics [187], sparse inverse covariance matrix estimation [27], [66], [114], and source localization [105].

The wide applicability notwithstanding, computing ℓ_1-regularized solutions continues to be a challenge. A closed form solution to (2.16) is not possible, and most techniques available to solve the problem are iterative, or approximate, or both [16], [17], [42], [82], [85], [124], [173], [186]. These techniques come from different fields and have various similarities and differences [110]. However, most of the available methods are still, in real terms, considerably slower than either solving (2.8) or (2.15), which is only to be expected. Attempts have been made to use statistical structure to speed up the computation in a compressed sensing framework [32], [53], [118], which does not precisely correspond to the parameter estimation challenge posed herein.

No attempt is made in this work to compare the quality of different possible regularization schemes in general, except in the particular simulations and applications for which results are presented. Both ℓ_2 and ℓ_1 regularization are in widespread use, and both can be applied to sample limited linear least squares problems; therefore they are both used for comparison against the strategy of this work, which is to reduce problem dimension.

The choice of which kind of regularization is more meaningful and powerful may be highly application-domain dependent—for instance, it has been suggested that in array processing, mismatch is a more important issue than sparsity, so that ℓ_2-regularization is more important; whereas, in image processing, compressed sensing has long been the tool of choice. Addressing the philosophical differences between the frameworks is outside the scope of this work.

2.1.5 The Need for Alternate Approaches

Powerful though regularization may be, there are some reasons why it may be beneficial to consider other approaches to handle sample deficiency.

The first is complexity. As stated above, particularly with ℓ_1-regularized methods, the amount of computation required to compute a regularized solution can be quite large. While it is often argued that computational resources are not limiting factors today, this is not always the case—especially when computations need to be carried out repeatedly or in real time.

Additionally, regularized solutions are not always appropriate for cases where data is being received on line. For the case of ℓ_1 and ℓ_0 regularization, there does not appear to exist a mechanism for incorporating new data into an existing solution. Because ℓ_2 regularization is equivalent to a Gaussian prior, the Kalman filter (see Section 2.2) is a suitable recursive method to incorporate new data; however, the Kalman filter requires $\mathcal{O}\left(M^3\right)$ operations per time-step, which is quite computationally intensive. On the other hand, the existence of an efficient ($\mathcal{O}\left(M^2\right)$ per time-step) recursive update for least squares (the recursive least squares algorithm [78]) is one of the most useful features of that algorithm.

Moreover, regularization suffers somewhat from sensitivity to the choice of the regularization parameter (the δ's in (2.13)—(2.17)). Formal proofs for the case of diagonal loading were provided by Pajovic [129, Chapter 3], wherein the optimum diagonal loading coefficient δ_2 is predicted using Random Matrix Theory and the sensitivity of the optimum diagonal loading to the physics of the problem is analyzed for array processing. Similarly, it has been shown [62] that when δ_1 of (2.16) is chosen in a data-dependent manner, the performance of the LASSO estimator can rapidly deteriorate in practice. Indeed, experience also bears this out, as a poor choice of parameter can cause regularized solution to perform worse than expected. Unfortunately, there is often no domain knowledge available in practical applications to fine-tune the parameter choice.

Perhaps most fundamental, however, is the nature of regularization itself. Regularization is equivalent to imposing a-priori beliefs on the parameter, as described in the preceding discussion. Thus, while conventional LS is a true "black-box" estimator in that it only depends upon the data, regularized LS is a “gray-box” estimator, where the penalty implicitly causes the estimator to assume knowledge about the unknown system.

In other words, while the assumptions that lead to a regularized solution are indeed useful in that they allow for a solution to be computed with less data than would otherwise be possible, the very fact that those assumptions have been made automatically skews the solution, which may not always be appropriate. As the underlying system is unknown, it may be argued that the safest assumption is to leave the solution unconstrained.

If constraining the estimate is not allowed, how can sample limited regimes be handled? One useful insight that arises from the previous discussions is that smaller linear learning problems require less data than larger ones (for obvious reasons), require less computation, and, as a consequence of (2.12b), can be computed with greater accuracy. If the overall large system identification problem could be separated into smaller subproblems, in theory, each of those subproblems could be solved more accurately than the overall problem. Note that this does not immediately imply that the overall solution can also be improved, but it does provide an initial indication that such improvement is, at least, possible.

This work exploits this insight by systematically reducing the problem dimension by exploiting structure on the input (for convenience, this will henceforth sometimes be called “input structure”). As the input is observed, an estimator that exploits input structure exploits only the data and its properties; it assumes nothing about the unknown system and thus treats it as a “black-box.”

The key ideas that allow input structure to be exploited are explained in Section 2.3. First, however, an extension of the problem of (2.1) to the case where the true system is varying with time is described, so that the benefits of reducing the problem dimension in both cases can be simultaneously considered.

2.2 Time-Varying Systems

An important generalization of (2.1) is obtained when the underlying system is allowed to vary with time [78], [107]. In other words, the relationship between $\boldsymbol{x}(n)$ and $y(n)$ is now given by

$$y(n) = \boldsymbol{h}_0^\dagger(n)\boldsymbol{x}(n) + v(n)\,. \tag{2.18}$$

The challenge in this case is to *track* the value of $\boldsymbol{h}_0(n)$, which is a dimension M vector, over time. The term "track" has been carefully chosen because it has a very specific connotation, as will now be explained.

2.2.1 The Tracking Problem

The problem of estimating $\boldsymbol{h}_0(n)$ from the inputs and outputs of the system needs to be considered carefully. To begin with, if absolutely no assumptions can be made on how $\boldsymbol{h}_0(n)$ is generated, then the problem at each time n simply reduces to time-invariant system identification with a single observation. It would be no surprise to find that such an estimate is likely to be terrible, even with regularization.

The more meaningful and realistic framework is to assume that the system varies "smoothly," i.e., that $\boldsymbol{h}_0(n)$ is derived from the previous values $\boldsymbol{h}_0(1), \boldsymbol{h}_0(2), \ldots$ in some meaningful way. In that case, the idea is that by combining the past estimates with the new data (which comes from the evolved system), the variations of the system can be *tracked* over time. Most commonly, the estimate of the system at time $n-1$ is combined with the data at time n to obtain the system at time n, in which case we say the estimate is computed *recursively.* Thus, tracking implies a recursive solution, which immediately distinguishes it from the processing framework for the time-invariant system identification problem that has been considered thus far.

The tracking problem as presented here is applicable to many situations. For instance, wireless underwater communication channels [38] and vehicular communication channels [165] are time-varying, so that channel identification and equalization on such channels require tracking. The system model of (2.18) has also been applied to tracking financial markets [46], [58], [106], to mobile robotics [144], [147] and many other problems in signal processing and communication [15].

2.2.2 Solving the Tracking Problem

Perhaps the most fundamental way to solve the tracking problem is the Kalman filter [94]. The idea behind the Kalman filter is to model the unknown system as a Gauss-Markov random process, i.e., the unknown system is assumed to be a linear dynamical system. Under this assumption, and under a Gaussian assumption for the noise, the Kalman filter is known to be an optimal filter in a statistical sense, i.e., its state estimates are the maximum a-posteriori estimates of the states given the data. Note that the Kalman filter does not make a statistical assumption on the input $\boldsymbol{x}(n)$—it merely treats the input as an observation vector through which the unknown system is observed.

However, the Kalman filter makes fairly strong assumptions about the underlying dynamical model for the system. Unmodeled system dynamics can seriously degrade the performance of the Kalman filter [149], which can become important in practice. The Kalman filter strategy also defeats the purpose of this work, as the objective is to make as few assumptions about $\boldsymbol{h}_0(n)$.

The recursive least squares (RLS) algorithm and the least mean squares (LMS) algorithm are computationally less complex than the Kalman filter and widely used. Both the RLS and LMS algorithms may be viewed as special cases of the Kalman filter [78, Section 10.8].

While it is sometimes assumed that RLS and LMS algorithms are used because a Kalman filter is computationally complex, neither the RLS nor the LMS algorithm require explicit models for the evolution of $\boldsymbol{h}_0(n)$. Thus, they are often more robust in real world situations. Moreover, the parameters of the dynamical model assumed by the Kalman filter are oftentimes unknown or inadequately known, in which case, the algorithm needs to simultaneously update its estimate of the parameters of the assumed dynamical model, and estimate the system itself. The requirement of learning so many parameters can sometimes impede its performance. Such an algorithm is termed an extended Kalman filter.

To the best of the author's knowledge, there is no theoretical performance comparison between the algorithms. Some simulation based results exist (e.g., [57], [102], [103]), but the answer of which one is the best appears to depend upon the setting and regime, which are themselves often hard to characterize. Oftentimes, therefore, the choice of algorithm is domain dependent and experience driven.

In this work, the RLS and LMS algorithms are used as baselines for comparison primarily because, for the applications considered for tracking in this thesis, such as adaptive equalization in wireless underwater communication, making assumptions on the dynamical model of the time-varying system is inappropriate. Computational complexity is also a factor that rules out the Kalman filter for these applications. Detailed derivations of the LMS and RLS algorithms may be found in the books by Haykin [78] or Sayed [148]. These works also contain limited theoretical and simulation based performance analysis of the algorithms.[4]

Recursive Least Squares

The exponentially weighted RLS algorithm is derived from the LS algorithm of 2.1.1 in two steps:

1. Modifying the cost criterion of (2.2) with an exponential *forgetting factor* to incorporate the effect of time-variance. This leads to the exponentially weighted least squares solution, which is obtained as the solution to the exponentially weighted cost criterion as follows
$$\hat{\boldsymbol{h}}_{\text{ew-ls}}(n) = \underset{\boldsymbol{h}\in\mathbb{C}^M}{\arg\min} \sum_{p=1}^{n} \lambda^{n-p} \left| y(n) - \boldsymbol{h}^\dagger \boldsymbol{x}(n) \right|^2 \tag{2.19}$$

2. Using the Sherman-Morrison matrix inversion identity [154] to obtain a recursive version of LS that can be applied one data point at a time.

Define by $\hat{\boldsymbol{R}}_{\boldsymbol{x}\lambda}(n)$ the exponentially weighted sample covariance matrix at time n, i.e.,

$$\hat{\boldsymbol{R}}_{\boldsymbol{x}\lambda}(n) = \sum_{p=1}^{n} \lambda^{n-p} \boldsymbol{x}(p)\boldsymbol{x}^\dagger(p)\,. \tag{2.20a}$$

Defining in a similar way the exponentially weighted sample cross correlation,

$$\hat{\boldsymbol{r}}_{\boldsymbol{x}y\lambda}(n) = \sum_{p=1}^{n} \lambda^{n-p} \boldsymbol{x}(p) y^*(p)\,, \tag{2.20b}$$

[4] Section 5.4 introduces and analyzes the RAGS-RLS with assumptions similar to those made in the analysis of Haykin [78].

Initialization	$\hat{\boldsymbol{J}}_{\boldsymbol{x}\lambda}(0) = \delta^{-1}\boldsymbol{I}, \hat{\boldsymbol{h}}_{\text{rls}}(0) = \boldsymbol{0}$
Kalman Gain Update	$\boldsymbol{k}(n) = \dfrac{\hat{\boldsymbol{J}}_{\boldsymbol{x}\lambda}(n-1)\boldsymbol{x}(n)}{\lambda + \boldsymbol{x}^{\dagger}(n)\hat{\boldsymbol{J}}_{\boldsymbol{x}\lambda}(n-1)\boldsymbol{x}(n)}$
Output Estimate	$\hat{y}_{\text{RLS}}(n) = \hat{\boldsymbol{h}}_{\text{rls}}^{\dagger}(n-1)\boldsymbol{x}(n)$
Coefficient Update	$\hat{\boldsymbol{h}}_{\text{rls}}(n) = \hat{\boldsymbol{h}}_{\text{rls}}(n-1) + \boldsymbol{k}(n)\left(y(n) - \hat{y}_{\text{RLS}}(n)\right)^{*}$
Inverse Covariance Matrix Update	$\hat{\boldsymbol{J}}_{\boldsymbol{x}\lambda}(n) = \dfrac{\hat{\boldsymbol{J}}_{\boldsymbol{x}\lambda}(n-1) - \boldsymbol{k}(n)\boldsymbol{x}^{\dagger}(n)\hat{\boldsymbol{J}}_{\boldsymbol{x}\lambda}(n-1)}{\lambda}$

Figure 2-1: Steps of the exponentially weighted RLS algorithm.

the exponentially weighted LS solution is given by

$$\hat{\boldsymbol{h}}_{\text{ew-ls}}(n) = \hat{\boldsymbol{R}}_{\boldsymbol{x}\lambda}^{-1}(n)\hat{\boldsymbol{r}}_{\boldsymbol{x}y\lambda}(n)\,. \tag{2.21}$$

If $\lambda = 1$, and N observations are being processed, $\hat{\boldsymbol{R}}_{\boldsymbol{x}\lambda}(N) = \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)$, which was defined in (2.9a), $\hat{\boldsymbol{r}}_{\boldsymbol{x}y\lambda}(N) = \hat{\boldsymbol{r}}_{\boldsymbol{x}y}(N)$ as defined in (2.9b), and $\hat{\boldsymbol{h}}_{\text{ew-ls}}(N) = \hat{\boldsymbol{h}}_{\text{ls}}(N)$. Thus, exponentially weighted LS is a generalization of LS. Additionally, the closer λ is to 1,the slower the maximum rate of time variability exponentially weighted LS can track but the better the noise rejection properties.

Recursive least squares (RLS) is simply a very efficient formulation of (2.21) for data received online, as it typically is with time-varying systems. Defining $\hat{\boldsymbol{J}}_{\boldsymbol{x}\lambda}(n) = \hat{\boldsymbol{R}}_{\boldsymbol{x}\lambda}^{-1}(n)$,[5] the steps of the RLS algorithm are defined in Figure 2-1.

The RLS algorithm has a complexity of $\mathcal{O}\left(M^2\right)$ per observation, in contrast to the Kalman filter, whose complexity is $\mathcal{O}\left(M^3\right)$ per observation. The RLS algorithm and its derivatives continue to be widely popular in the study of time-varying online systems in a variety of fields [86], [91], [156].

[5] Throughout this thesis, $\boldsymbol{J}$ will represent the inverse covariance matrix. This notation will be introduced later.

Least Mean Squares and Normalized Least Mean Squares

Least mean squares (LMS) is a highly efficient method to track the unknown system $\boldsymbol{h}_0(n)$. The LMS solution is extremely fast (with complexity $\mathcal{O}(M)$ per data point) and its performance is fairly good—it can even outperform the more complex RLS algorithm in some situations.

The update of the LMS algorithm is particularly simple:

$$\hat{\boldsymbol{h}}_{\text{lms}}(n) = \hat{\boldsymbol{h}}_{\text{lms}}(n-1) + \mu_{\text{lms}}\boldsymbol{x}(n)\left(y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\text{lms}}(n-1)\boldsymbol{x}(n)\right)^{*}, \tag{2.22}$$

where μ_{lms} is the step-size of the LMS algorithm.

To improve stability, the principle of minimum disturbance, i.e., the philosophy that the filter should be moved as little as possible, is often adopted into the LMS algorithm, leading to the so called Normalized LMS (NLMS) adaptation algorithm. One form of the N-LMS adaptation equation is

$$\hat{\boldsymbol{h}}_{\text{nlms}}(n) = \hat{\boldsymbol{h}}_{\text{nlms}}(n-1) + \frac{\mu_{\text{nlms}}}{\|\boldsymbol{x}(n)\|^2}\boldsymbol{x}(n)\left(y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\text{nlms}}(n-1)\boldsymbol{x}(n)\right)^{*}, \tag{2.23}$$

where now the step size is μ_{nlms}.

Both the N-LMS and LMS can be interpreted as optimal in an H^{∞} sense, which is a notion of robustness to modeling errors [75]. This partially explains the robustness of LMS to modeling errors—it is often considered even more robust than RLS.

2.3 Input Structure to Reduce Problem Dimension

Having examined the problems that are under consideration in this thesis and a variety of existing solutions to them, the approach taken in this thesis is considered.

The idea behind this thesis is to systematically reduce problem dimension of the estimation and tracking problems by exploiting structure on the input to the estimation problem. Recall that the dimension of $\boldsymbol{h}_0$ is M. Rather than trying to directly solve estimation and tracking problems of dimension M, we attempt to split the problem into a set of subproblems each of smaller dimension, solve those and then recombine the solutions to solve the full problem.

2.3.1 Goals of Reducing Problem Dimension

Before going any further, it is worth noting what benefits may be anticipated by reducing the problem. The answer is slightly different depending on whether the system in question is time-invariant or time-varying.

For time-invariant systems, it is evident that smaller dimensional system identification problems require less data. The performance of LS for smaller dimensional LS problems is also better, as indicated by (2.12b). Moreover, the computational complexity of a system identification problem also increases with its dimensionality. Thus, if the problem is appropriately split into smaller pieces, a reasonable expectation might be to improve either performance or complexity; and to reduce the data requirements for the identification problem. Note that the computation and performance gains are not obvious, because although each individual problem is smaller, multiple such problems need to be solved. However, it is at least conceivable that the such gains could exist.

For time-varying systems, there is no precise notion in a recursive framework of a specific requirement on the amount of data. However, it is certainly reasonable to expect a set of smaller problems to be faster than a single larger one, so complexity gain may be expected from splitting the problem. Performance gains may be possible, but as has been noted, how well a tracking algorithm performs appears to depends upon the regime of operation of the algorithm. Thus, whether performance gains are possible by intelligently splitting the problem and if so, how much, are both questions on which it is difficult to get insight before the algorithms are actually developed.

At a high level, there appear to be good reasons to investigate reducing dimension by splitting the problem in a meaningful way. For reasons such as those of Section 2.1.5, the approach chosen in this thesis is to do so by exploiting input structure. However, this is not trivial to do, as explained next.

2.3.2 The Challenges of Exploiting Input Structure

The idea of using input structure may appear at an intuitive level as an obvious strategy to reduce problem dimension. Indeed, it is not novel in signal processing to exploit the structure of an observed signal in this manner. For instance, the entire field of source coding [44] is predicated on exploiting input structure. In frequency domain equalization [119], it is sometimes possible to reduce equalization complexity by processing different frequencies independently.[6] Subarray processing [1], [127] utilizes a similar principle wherein a large array is separated into smaller subarrays for processing to reduce complexity. Exploiting input structure is a fairly common strategy in the computation of covariance matrices in array processing, e.g., [71], [100], [128], [134], [137], [160], [180].

In the context of the adaptive signal identification problem, however, constraining input structure is considerably more difficult. The reasons may not appear obvious at first. Indeed, it has been suggested that several kinds of structure may be imposed that can improve the estimate of the inverse covariance matrix, such as assuming a diagonal covariance structure (for obvious reasons), a Toeplitz covariance structure [109] or sparsity [66], [92]; and such an improved estimate of the input signal statistics can be used to improve the system estimate.

The difficulty arises because the output of the system is a mix of all the input components, so it is not clear how input structure might manifest in the output. Put a different way, the system identification problem is characterized by both the input and the output of the system of interest—constraining one without suitably modifying the other is meaningless. To truly improve the estimate of the system, any part of the estimate of $\boldsymbol{h}_0$ that depends upon the output of the system $y(n)$ would have to be computed in a manner consistent with how a structured input would generate the output. In general, it has been found that correctly enforcing these constraints is fairly challenging.

For instance, it was observed by Blair [22, Chapter 3] that simply trying to directly exploit Toeplitz structure by constraining the inverse covariance matrix of the data does not lead to the performance gains that may be expected while estimating adaptive equalizer

[6] There are some caveats to this. In fact, the work of this thesis shows that one can not naively process different frequencies independently, as the observed output is usually a mixture of the outputs at different frequencies. One has to first estimate the outputs corresponding to different frequencies and then use those outputs to estimate the equalizer coefficients—see Chapter 4.

coefficients. In the context of this work as well, it is discussed in Section 5.3.4 that enforcing graphical model constraints on the input to the system without suitably constraining the output leads to a deterioration in performance.

2.3.3 Graphical Model Input Structure

The question then becomes: in the context of the general linear system identification and tracking problem, what kind of structure can meaningfully be used to reduce the dimensionality of estimation problems? The notion of *probabilistic graphical models* will allow the system identification and tracking problem to be split into smaller problems in a statistically meaningful way.

The structure that is meaningful is the system identification problem is that different subvectors of the input are statistically independent of one another (or, more precisely, statistically related only through low-dimensional subsets). The idea is that these statistically independent subvectors can form sub-problems whose linear combination gives back the complete parameter identification problem.

The reason that this kind of structure is exploitable in linear parameter identification whereas, say, Toeplitz structure is not, is that statistically independent inputs lead to statistically independent parts of the output, and the overall output can then be modeled as a linear combination of independent random variables. It is possible to discriminate between a linear combination of statistically independent random variables and statistically dependent ones; which makes this a non-trivial method of capturing structure on the input, and having it meaningfully affect the output.

To formalize this notion, start with the assumptions made on the input in Section 2.1.2. Specifically, the inputs $\boldsymbol{x}(n)$ are treated as independent realizations of a zero-mean Gaussian random variable x whose covariance matrix is $\boldsymbol{R}_{\mathsf{x}}$. We define the *inverse covariance matrix* of x by $\boldsymbol{J}_{\mathsf{x}} = \boldsymbol{R}_{\mathsf{x}}^{-1}$.

By statistical structure on the input, structure on the distribution $p_{\mathsf{x}}(\boldsymbol{x})$ is meant. While there are many ways to capture such structure, in recent times, probabilistic graphical models [98] have emerged as a powerful way to represent structure in distributions. By allowing complex distributions to be factored into smaller components, they allow for particularly efficient statistical inference and parameter learning. A comprehensive discussion of graphical models and algorithms for inference and learning on such models may be found in the book by Koller [96].

The graph indicates that some parts of the input are conditionally independent of the others, conditioned upon some specific subset of the input (see Section 2.4). This then leads to algorithms where independent parts of the input each serve as the input to separate identification problems which produce independent outputs. The overall output $y(n)$ can then be modeled as a combination of those independent outputs. Evidently, the outputs are also then related to the inputs and to one another via conditional independence relationships, so that the entire problem can be captured within one consistent framework. Such an approach will be shown to have several of the advantages that were envisioned in Section 2.3.1.

In the rest of this chapter, a brief overview of the concept of graphical models is provided. In Chapter 3, some motivating examples are provided of real-world system identification problems where the input is characterized by a graphical model. The algorithms to exploit the model in system identification and tracking will be introduced in Chapters 4 and 5.

2.4 Probabilistic Graphical Models

A probabilistic graphical model is a way of graphically representing structure in multivariate distributions using a graph. A graph $\mathcal{G}$ is a set of M vertices or nodes $\mathcal{V}$ connected by edges $\mathcal{E}$. The variables of the multivariate distribution are placed in one-to-one correspondence with the vertices of the graph.

The pattern of edges of the graph represent the structure in the distribution. In general, graphical models work by representing a set of "local" relationships among variables. More precisely, they represent conditional independence relationships between subsets of the variables. Depending upon the set of relationships that are being represented, there are two kinds of graphical models that could be used: *undirected graphical models*, also called Markov Random Fields (MRFs) and *directed graphical models*, also called Bayesian networks. These names are descriptive of the kind of relationships that are represented by the graphs.

2.4.1 Undirected Graphical Models

In an undirected graph, as the name suggests, the edges have no directionality associated with them, i.e., an edge between the nodes i, j is the same as an edge between j, i. Let $\mathcal{G}$ be an undirected graph with M nodes.

A graph is said to be *fully connected* if there is an edge between every pair of nodes in the graph. A *clique* is a fully connected subgraph of a graph. A clique c of a graph $\mathcal{G}$ is said to be *maximal* if there does not exist a node $v \in \mathcal{V}$ such that $v \notin c$ and $v \cup c$ is a clique.

Let $\mathcal{C}$ be the set of maximal (possibly overlapping) cliques of $\mathcal{G}$. For $c \in \mathcal{C}$, define a *potential function* $\psi_c(\boldsymbol{x}_c)$ as a non-negative function over all realizations of the variables of clique c. Then, the distribution of random variable x is said to *factor* according to $\mathcal{G}$, or to be characterized by $\mathcal{G}$, if,[7] for some potential functions $\psi_c(\cdot)$

$$p_{\mathsf{x}}(\boldsymbol{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{x}_c) \,. \tag{2.24}$$

In the above Z is a normalization constant given by[8]

$$p_{\mathsf{x}}(\boldsymbol{x}) = \int_{\boldsymbol{x} \in \mathbb{C}^M} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{x}_c) \, \mathrm{d}\boldsymbol{x} \,. \tag{2.25}$$

The factorization of (2.24) is shown in Figure 2-2 for two undirected graphs.

It can be shown that the factorization of (2.24) is equivalent to a set of conditional independence relationships [111]. If $\mathcal{E}$ is the set of edges of $\mathcal{G}$,

$$(k, m) \notin \mathcal{E} \iff \mathsf{x}_k \perp\!\!\!\perp \mathsf{x}_m \mid \mathsf{x}_{\setminus k,m} \,, \tag{2.26}$$

where $\mathsf{x}_{\setminus k,m}$ means all components of the random vector x *except* for x_k and x_m. This is termed the *pairwise Markov* property of the undirected graph.

A stronger *global Markov* property can also be stated for the graph. Let $(\mathcal{A}, \mathcal{B}, \mathcal{D})$ be a

[7] There is no distinction in this work between the terms "the random variable x is characterized by the graph $\mathcal{G}$" and "the distribution $p_{\mathsf{x}}(\boldsymbol{x})$ is characterized by the graph $\mathcal{G}$.

[8] If the domain of x is not $\mathbb{C}^M$, the integration (or sum) is over the domain in general.

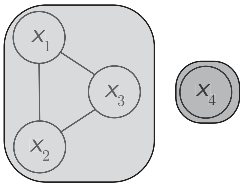
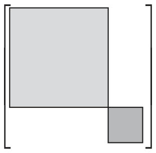
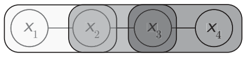
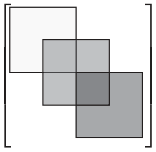
Graph $\mathcal{G}$	Factorization of PDF	Inverse Covariance Matrix $\boldsymbol{J_x}$ (Gaussian)
	$p_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{Z}\psi_1(x_1, x_2, x_3)\psi_2(x_4)$	
	$p_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{Z}\psi_1(x_1, x_2)\psi_2(x_2, x_3)\psi_3(x_3, x_4)$	

Figure 2-2: Decomposition of a probability density function for undirected graphical models and the corresponding inverse covariance matrix structure for Gaussian random vectors. Shaded boxes enclose cliques. Elements of the inverse covariance matrix outside the shaded boxes are 0.

set partition of $\mathcal{V}$ such that $\mathcal{D}$ separates $\mathcal{A}$ and $\mathcal{B}$ on the graph. Then, it can be shown that

$$\mathbf{x}_{\mathcal{A}} \perp\!\!\!\perp \mathbf{x}_{\mathcal{B}} \mid \mathbf{x}_{\mathcal{D}} \,. \tag{2.27}$$

The celebrated Hammersley-Clifford Theorem [74] states that, for strictly positive distributions, the global Markov property is equivalent to the distribution factorization of (2.24) which in turn is equivalent to the pairwise Markov property, i.e., a strictly positive multivariate distribution factors according to $\mathcal{G}$ if and only if the random variables satisfy the global Markov property over $\mathcal{G}$, which in turn happens if and only if they satisfy the pairwise Markov property over $\mathcal{G}$.

Gaussian Graphical Models

Now, in addition to being characterized by $\mathcal{G}$, suppose that $\mathbf{x} \sim \mathcal{CN}(\mathbf{0}\,,\, \boldsymbol{R}_{\mathbf{x}})$, and let $\boldsymbol{J}_{\mathbf{x}} = \boldsymbol{R}_{\mathbf{x}}^{-1}$, i.e., $\mathbf{x}$ is a multivariate (complex) normal vector with covariance matrix $\boldsymbol{R}_{\mathbf{x}}$ and inverse covariance matrix $\boldsymbol{J}_{\boldsymbol{x}}$. Then it can be shown that [96, Theorem 7.2][9]

$$(k, m) \notin \mathcal{E} \iff J_{\boldsymbol{x}_{k,m}} = 0\,. \tag{2.28}$$

The inverse covariance matrix structure is illustrated in Figure 2-2 for two different undirected graphs.

The structure of the inverse covariance matrix for Gaussian random variables is thus related to the Markov properties (as Gaussian distributions are strictly positive, the pairwise and global Markov properties are equivalent).

Equation (2.28) provides the first mathematical hint that efficiencies may be possible by exploiting graphical model input structure in adaptive estimation. Looking at the LS solution of (2.8), one of the terms of the solution is the sample inverse covariance matrix of $\boldsymbol{x}$, which is an estimate of the inverse covariance matrix $\boldsymbol{J}_{\boldsymbol{x}}$. If $\boldsymbol{x}$ is characterized by a graph $\mathcal{G}$, then $\boldsymbol{J}_{\boldsymbol{x}}$ has a number of zeros, as shown by (2.28), which can be exploited to simplify its estimation. Indeed, an estimator that exploits this property to improve the inverse covariance matrix estimate has been developed [181], [182]. The problem under consideration is not inverse covariance estimation but linear system identification, which requires a more indirect approach, but this provides some initial insight.

Decomposable Graphical Models

One more notion that is needed for this thesis is that of *decomposability.*

A set partition $(\mathcal{A}, \mathcal{B}, \mathcal{D})$ of $\mathcal{V}$ is said to form a *decomposition* of the graph $\mathcal{G}$ if

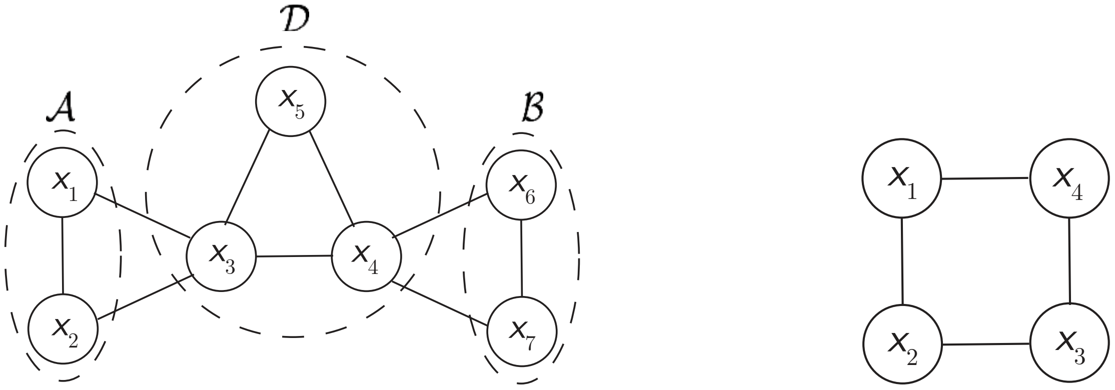
1. $\mathcal{D}$ separates $\mathcal{A}$ and $\mathcal{B}$ in $\mathcal{G}$, and,
2. $\mathcal{D}$ is *complete* or fully connected, meaning that every pair of nodes in $\mathcal{D}$ is connected by an edge.

The decomposition is said to be *proper* if $\mathcal{A} \neq \emptyset$ and $\mathcal{B} \neq \emptyset$.

An undirected graph $\mathcal{G}$ is said to be decomposable if

1. It is complete, or

[9] Note that *neither* direction of (2.28) is implied if $\boldsymbol{x}$ is not Gaussian.



(a) Decomposable graph, with $\mathcal{A}, \mathcal{B}$ and $\mathcal{D}$

(b) Non-decomposable graph

Figure 2-3: Examples of decomposable and non-decomposable graphs

2. It admits a proper decomposition into decomposable subgraphs.

The definition is thus recursive. Figure 2-3a shows an example of a decomposable graph with one possible choice for the $\mathcal{A}, \mathcal{B}, \mathcal{D}$ decomposition, and Figure 2-3b shows an example of a graph that is *not* decomposable.

An important property of decomposable graphs is that there is an ordering of cliques, $c_1, c_2, \ldots$, which are connected by *separators*, $s_1, s_2, \ldots$, where

$$s_k = \left(\bigcup_{i=1}^{k} c_i \right) \cap c_{k+1} \, , \tag{2.29}$$

so that, for every k, there exists $j \leq k$ so that $s_k \subseteq c_j$. This is essentially the so-called "junction tree" property [12]. Note that in this definition, s_k may be an empty set for some k. However, in this work, only non-empty separators are of interest.

In this work, the set of non-empty separators of a decomposable graph $\mathcal{G}$ is denoted using $\mathcal{S}$. Importantly, the set $\mathcal{S}$ is permitted to have repetitions (in other words, a separator may appear more than once), but it satisfies the property that if the node $v \in \mathcal{V}$ appears in C_v cliques, then it appears in $C_v - 1$ separators. While this definition may appear unnecessarily complicated, the set is defined as such so that the estimators in the rest of the work can be defined in a consistent manner with minimal additional notation required.

Some examples of the sets $\mathcal{C}$ and $\mathcal{S}$ as used in this thesis for several decomposable graphs are shown in Figure 2-4, which shows when the set $\mathcal{S}$ is treated as empty and when it

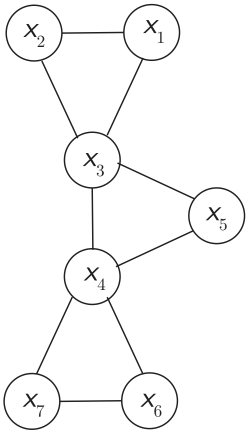
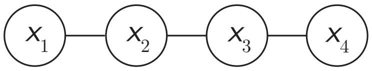
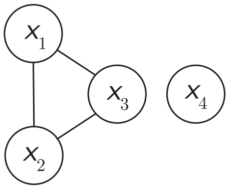
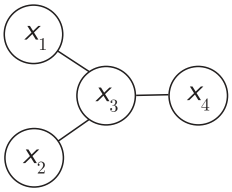
Graph $\mathcal{G}$	$\mathcal{C}$	$\mathcal{S}$
	$\{\{x_1, x_2, x_3\}, \{x_3, x_4, x_5\}, \{x_4, x_6, x_7\}\}$	$\{\{x_3\}, \{x_4\}\}$
	$\{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}\}$	$\{\{x_2\}, \{x_3\}\}$
	$\{\{x_1, x_2, x_3\}, \{x_4\}\}$	$\emptyset$
	$\{\{x_1, x_3\}, \{x_2, x_3\}, \{x_3, x_4\}\}$	$\{\{x_3\}, \{x_3\}\}$

Figure 2-4: Examples of the sets of maximal cliques $\mathcal{C}$ and separators $\mathcal{S}$. The separator set, in particular, is defined so that it only contains non-empty separators, and additionally, it may contain repetitions; but if a node v appears in C_v cliques, then it appears in $C_v - 1$ separators.

contains repetitions. Denote by $C = |\mathcal{C}|$ and $S = |\mathcal{S}|$ as the number of maximal cliques and non-empty separators in the graph (where the set is defined as in the previous paragraph), respectively.

One of the main reasons for the importance of decomposable graphical models is that, for such models, the distribution $p_{\mathbf{x}}(\boldsymbol{x})$ can be factored as

$$p_{\mathbf{x}}(\boldsymbol{x}) = \frac{\prod_{c \in \mathcal{C}} p_{\mathbf{x}_c}(\boldsymbol{x}_c)}{\prod_{s \in \mathcal{S}} p_{\mathbf{x}_s}(\boldsymbol{x}_s)} . \tag{2.30}$$

2.4.2 Directed Acyclic Graphical Models

A directed graphical model represents a different probability distribution factorization than does a undirected model. The directed graphs of interest do not have cycles, so they are called directed acyclic graphs (DAGs).

Every node in a DAG $\mathcal{G}_{\mathrm{d}}$ has a set of *parent* nodes—the set of nodes that have edges ending in that node (the parent nodes of a particular node can be the empty set). Let the parents of node m be denoted π_m. Then, the distribution of a multivariate random variable $\mathbf{x}$ that factors according to $\mathcal{G}_{\mathrm{d}}$ can be written as:

$$p_{\mathbf{x}}(\boldsymbol{x}) = \prod_{m=1}^{M} p_{\mathsf{x}_m | \mathbf{x}_{\pi_m}}(x_m \mid \boldsymbol{x}_{\pi_m}) . \tag{2.31}$$

Some examples of such directed models and the factorization of their corresponding distributions are shown in Figure 2-5. Evidently, the distribution structure being represented is Bayesian factorization—the graph represents structure in conditional distributions. This is the reason that DAG models are termed Bayesian networks.

From the form of the distributions, it should be clear that the directed model represents the set of conditional independence statements:

$$\mathsf{x}_m \perp\!\!\!\perp \mathsf{x}_{\nu_m} \mid \mathsf{x}_{\pi_m} , \tag{2.32}$$

where ν_m are the set of vertices that appear earlier than m in the ordering of the vertices, not including its parents.

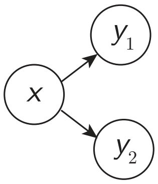
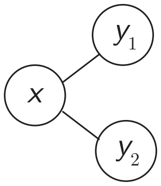
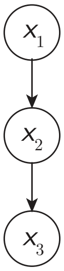
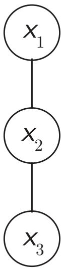
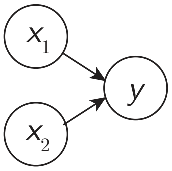
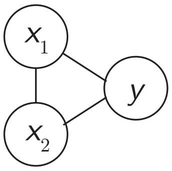
Directed Graph $\mathcal{G}$	Factorization of PDF	Moral Graph
	$p_{\mathsf{x}}(x)p_{\mathsf{y}_1\vert\mathsf{x}}(y_1 \mid x)p_{\mathsf{y}_2\vert\mathsf{x}}(y_2 \mid x)$	
	$p_{\mathsf{x}_1}(x_1)p_{\mathsf{x}_2\vert\mathsf{x}_1}(x_2 \mid x_1)p_{\mathsf{x}_3\vert\mathsf{x}_2}(x_3 \mid x_2)$	
	$p_{\mathsf{x}_1}(x_1)p_{\mathsf{x}_2}(x_2)p_{\mathsf{y}\vert\mathsf{x}_1,\mathsf{x}_2}(y \mid x_1, x_2)$	

Figure 2-5: Decomposition of a probability density function for a directed acyclic graph and the corresponding (moral) undirected model.

Moralization

The process of transforming a DAG into an undirected graph is termed *moralization* [45, Section 3.2.1]. The resulting undirected graph is called a moral graph. The moral graph of a DAG is obtained by adding an edge between every pair of nodes that have a common child, if one does not exist, and then making all edges undirected. While we do not go into detail here, moralization is necessary to ensuring that the resulting undirected graph correctly captures only the conditional independence statements implied by the DAG (it may not capture all of the DAG conditional independence statements).

Figure 2-5 shows the moral graph, i.e., for each of the directed graphical models in the figure.

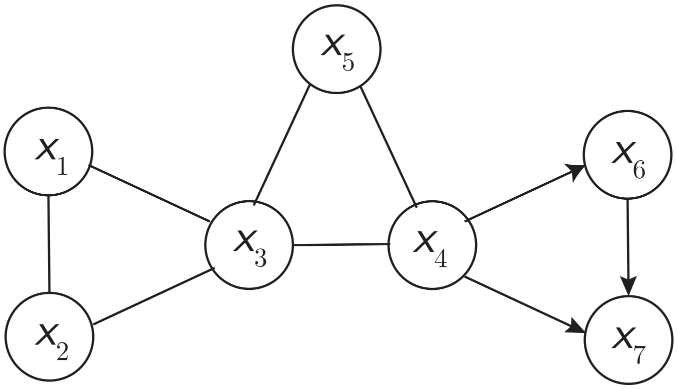


Figure 2-6: Example of a partially directed acyclic graph (PDAG). The factorization of the distribution for this graph is shown in (2.33).

2.4.3 Partially Directed Acyclic Graphs

Hybrids of directed and undirected graphs can be defined, and are termed partially directed acyclic graphs (PDAGs), or mixed graphs. In a PDAG, the joint distribution has a factorization that is partly represented by an MRF and partly by Bayesian factorization.

An example of a PDAG is shown in Figure 2-6. For the PDAG of the figure, the distribution factors as

$$p_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{Z}\psi_1(x_1, x_2, x_3)\psi_2(x_3, x_4, x_5)p_{\mathsf{x}_6|\mathsf{x}_4}(x_6 \mid x_4)p_{\mathsf{x}_7|\mathsf{x}_4,\mathsf{x}_6}(x_7 \mid x_4, x_6)\,. \tag{2.33}$$

Note that, for the parts of the graph that are represented by undirected edges, the distribution factors into functions (in the example, $\psi_1(x_1, x_2, x_3)$ and $\psi_2(x_3, x_4, x_5)$) that do not necessarily correspond to conditional distributions, whereas the parts of the graph represented by directed edges lead to a Bayesian factorization in the distribution.

PDAGs thus offer the flexibility of being able to represent both types of factorization in a single joint distribution. This property will be useful in the study of the system identification problem, because the output of the system is related to its input using a Bayesian relationship, whereas it is more natural (and more general) to assume that the input has decomposable undirected Gaussian graphical model structure. Section 4.1 shows how the representation is done for the problem of interest.

2.4.4 Graphical Model Structure for Input Data

The structure exploited in this thesis is primarily undirected graphical model structure. It is assumed that the inputs to the system identification problem $\boldsymbol{x}(n)$ are independent realizations of a multivariate Gaussian random variable x, for reasons explained in Section 2.1.2.

For this thesis, it is assumed that $p_{\mathsf{x}}(\boldsymbol{x})$ factorizes according to a decomposable Gaussian graphical model. In other words, the inverse covariance matrix of the input $\boldsymbol{J}_{\mathsf{x}}$ has some form of structured sparsity.

A large part of the thesis is devoted to developing algorithms that exploit such structure. First, however, it becomes important to understand in what real world situations such an assumption may be valid—in which system identification applications might we expect the input to be characterized by a graphical model? This is considered in the next chapter.

Unless otherwise stated, all algorithms and methods in this thesis will be assumed to have access to the correct graphical model. In particular, most simulations and analysis make this assumption (except while specifically considering the effect of the wrong model on the GEM-LS algorithm in Section 4.3.3. The models of Chapter 3 are a result of the physical properties of data in the respective application. There is extensive work on learning graphical models using a variety of methods— e.g., [23], [39], [114]—but neither the issue of graph learning nor the effect of an imperfectly learned graph on the algorithms presented will be considered in any depth.

Note that there is extensive literature on graphical models for time series. For instance, there is work on dynamically varying graphs to describe time series [20], [31], [189], models that attempt to describe relationships between entire time series [8], and of course the ubiquitous models that use directed graphs to specify the evolution of variables with time, such as HMMs [11]. The approach taken in this work is different in that it uses graphs to express the relationships between different variables within a vector, assuming that the time-evolution of the vector is trivial. For this work, it is assumed that the realizations of the vector are independent from time index to time index.[10]

It is worth noting that while graphical models do often capture useful and intuitive

[10] In some restricted ways, the independence of realizations can be relaxed.

structure in the form of conditional independences, they are by no means the only kind of structure that can be represented. Often, exploiting other kinds of distribution structure is key to the development of useful algorithms.

2.5 Some Comments and Looking Ahead

This chapter has considered in detail the system identification problem of interest and a survey of existing literature in the field has been considered. The prevalent strategy in the literature thus far has been that of constraining the estimate, and a variety of reasons for considering the alternative method of exploiting input structure were detailed. Finally, it was discussed that graphical models provide a way to exploit input structure by allowing the problem to be sub-divided, and relevant concepts of graphical models were introduced.

In the following chapter, several system identification problems will be shown to have graphical model input structure. For the applications considered in this thesis, the results of Section 2.4.4, and in particular (2.28), will be sufficient to show graphical model structure. These applications will also be used to test the performance of algorithms in practice in future chapters.

On an aside, the system identification problem, as has been indicated, has a rich and long history, and makes an appearance in several different fields. The applications considered in Chapter 3 have likewise been widely explored. Each of the application fields has given rise to several techniques and somewhat diverse language and priorities for dealing with its corresponding set of problems.

This thesis sits on the boundary of several of these fields: signal processing and inference, stochastic analysis, physical acoustics (due, in part, to the applications considered, such as underwater acoustics which traditionally strongly relies on acoustic properties). As a consequence, it sometimes presents a challenge to reconcile the various perspectives from a viewpoint that allows for useful development in the context of the work of this thesis. Nonetheless, it has become clear during these researches that viewing the problem through the lens of the statistical properties of the problem allows for the development of interesting algorithms and performance improvements.

Chapter 3

Select Applications with Graph-Structured Inputs

While developing algorithms, the key question is always, “Where will it be useful?” Likewise, making an assumption about input structure is only meaningful if there are real-world problems that satisfy the assumptions.

In Section 2.4, it was explained that decomposable undirected graphical model structure on the input is used to reduce the problem dimension in this thesis. In this chapter, this strategy is motivated by showing some examples of real-world system identification problems in which the input has graphical model structure.

First, a particularly simple example—that of independent Gaussian input processes—is considered and applied to acoustic echo cancellation. This is then extended to the case of processes whose covariance matrices are block-diagonal, and applied to the problem of channel identification in a fractionally spaced channel.

Then a more powerful result is shown regarding the popular class of wide-sense cyclostationary processes. While graphical model structure is not immediately evident in such processes, it is shown that, in the frequency domain, cyclostationary processes exhibit conditional independence structure that can be represented using a graphical model.

It is then shown how structure that is manifest in a finite dataset differs from the structure that would be observed if infinite data were available for a process with a diagonal covariance matrix. The analysis presents initial results into how theoretical graphical model structure can be modified to handle practical constraints.

Finally, the results on cyclostationary processes are combined with insights from the finite dataset analysis to analyze the structure of acoustic communication data from the SPACE08 experiment. The results of this section are useful in the problem of adaptive equalization of an underwater communication channel, which will be considered in Section 5.5.2.

Apart from motivating the structure, this chapter also serves to introduce the real-world applications that will be used to test the performance of the algorithms.

3.1 The Basic Methodology

It is worth beginning by laying out the principle of how the graph is obtained in the applications considered in this thesis. The tools for understanding the graphical model structure in the applications have already been introduced—all that needs to be done is show how they work together to realize graphical model structure.

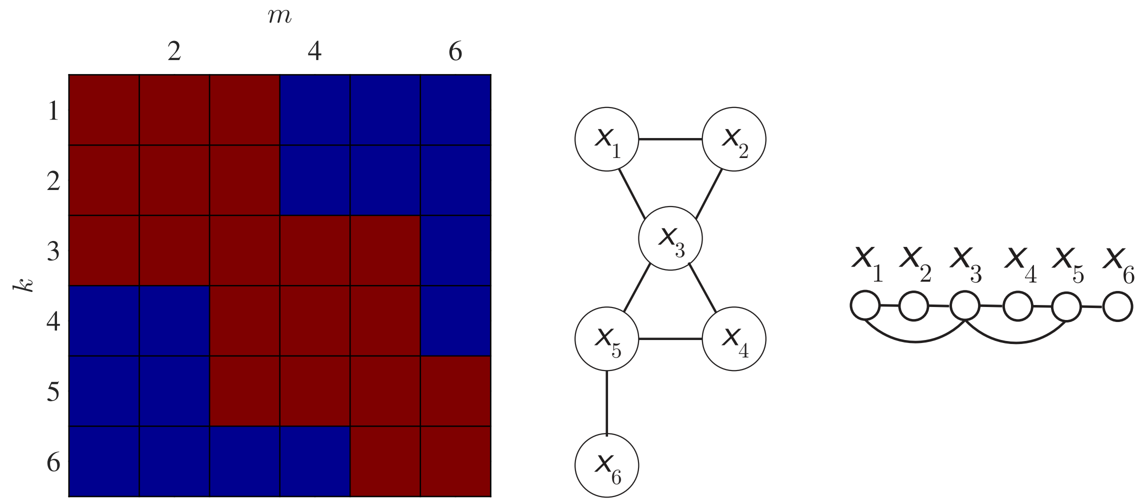
The basic idea is the following: in the linear system identification problem of interest, the inputs $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(N)$ can be assumed to be realizations of a multivariate Gaussian random vector x. (see Section 2.1.2). Making that assumption, therefore, it can be shown that (cf. (2.28))

$$(k, m) \notin \mathcal{E} \iff J_{\mathsf{x}_{k,m}} = 0\,, \tag{3.1}$$

which implies that a graphical model description on x is equivalent to structured sparsity on the inverse covariance matrix.

This is useful, as the inverse covariance matrix $\boldsymbol{J}_{\mathsf{x}}$ is a second order property of x, and can easily be analyzed with simple signal processing techniques. Thus, the strategy employed is that, for the application of interest, the inverse covariance matrix of the input to the problem is analyzed for zeros. Then the graphical model representing x can be drawn by placing an edge between nodes (k, m) whenever $J_{\mathsf{x}_{k,m}} \neq 0$.

In practice, it is often useful to draw the graph by placing an edge between nodes (k, m) whenever $J_{\mathsf{x}_{k,m}} > \epsilon$ for some judiciously chosen ϵ. In other words, the definition of conditional independence is relaxed, and the graph represents only "sufficiently large" conditional dependences.



(a) Inverse Covariance Matrix

(b) Gaussian graphical model

(c) Graph, concise representation

Figure 3-1: Demonstrating how the inverse covariance matrix structure is converted to a graphical model structure. The red squares in Figure 3-1a indicate elements of a representative inverse covariance matrix whose magnitudes are larger than the chosen threshold ϵ and the blue squares elements that are smaller (i.e., blue squares are pairs of elements that are being modeled as conditionally independent in the graph). Figure 3-1b shows the corresponding graphical model. In Figure 3-1c, a concise representation of the same graph is shown.

The method of translating inverse covariance matrix structure to a graphical model is demonstrated in Figure 3-1. In Figure 3-1a, the relative magnitudes of the elements of an inverse covariance matrix are shown on a dB-scale, i.e., the checkerboard plot shows $10\log_{10}$ of the magnitudes of the elements of the inverse covariance matrix. The corresponding graph is shown in Figure 3-1b, which has an edge between every pair of elements for which the corresponding inverse covariance matrix entry is non-zero (or, more generally, sufficiently large).

Figure 3-1c a different and more concise representation of the same graph is shown. It should be understood that Figure 3-1b and 3-1c are the same graph. The condensed representation in Figure 3-1c is a more convenient representation for relatively large graphs. As the applications of this thesis all result in graphs with a few tens to a few hundreds of nodes, this is the representation of choice throughout this chapter. However, the representation of Figure 3-1b is easier to visualize and is used in Chapter 4 and beyond. It is emphasized that the graphs are the same no matter which representation is chosen. Additionally, it should be understood that the graphs are always Gaussian graphical models—if Gaussianity is not assumed, then the inverse covariance structure is not equivalent to a graphical model.

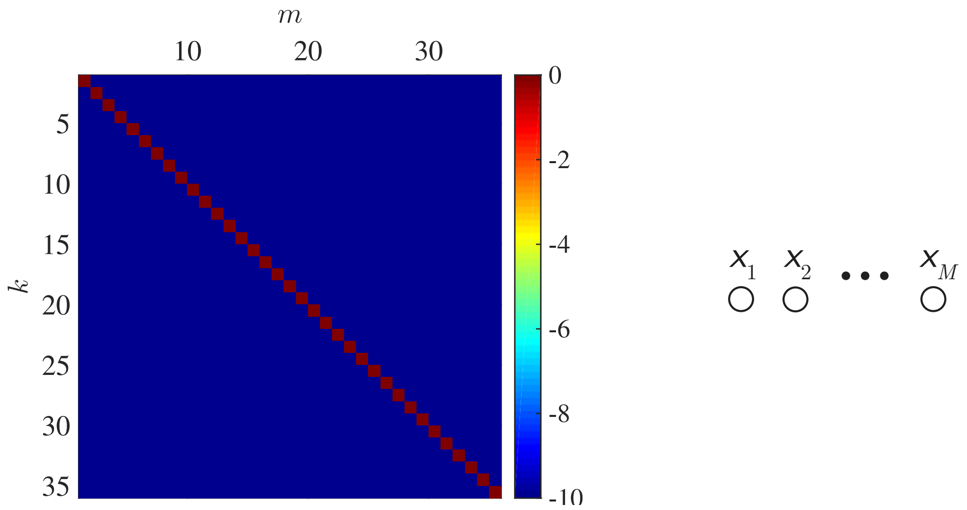
It may appear that, as only the inverse covariance structure is relevant, the graph does not really need to be drawn. However, the graph is an invaluable tool for visualizing the structure in the problem and understanding how to exploit it. Indeed, as it be seen in Section 4.1, the graph is a key tool in algorithm development without which the algorithms developed herein may not have become apparent.

There are different ways to investigate inverse covariance structure. One simple method is an off-line empirical study of the properties of the inverse covariance matrix or graph. While that may suffice for some applications, in this work, the ensemble inverse covariance of the input in various applications is shown theoretically to have sparse structure using the statistical properties of the signal. For two of the three applications considered—acoustic echo cancellation and channel identification (see Sections 3.2 and 3.3)—the development is especially simple. For adaptive equalization (Section 3.4), the statistical property that leads to graph structure is cyclostationarity, and a little more work needs to be done to show that cyclostationarity is equivalent to inverse covariance structure in the frequency domain.

It is emphasized that the *statistical*, as opposed to the *physical* properties of the data, are being considered. The physics of the various applications certainly affect the statistics of the data; however, in all the cases here, relatively mild assumptions regarding the statistics of the signals and environment are made that are minimally affected by the particular physical properties of a dataset, and these assumptions are sufficient to show the existence of inverse covariance/graph structure.

3.2 Input Processes with Diagonal Covariance Matrices

Let $\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(N)$ be independent realizations of a zero-mean Gaussian vector x where $\mathsf{x} \sim \mathcal{CN}\left(\mathbf{0}\,,\, \sigma_{\mathsf{x}}^2 \boldsymbol{I}_M\right)$. It is obvious then that the vector x factorizes according to a trivial graph with all independent nodes, i.e., a graph with no edges. Of course, the covariance matrix need not specifically be a scaled identity matrix—it need only be diagonal. Thus, the result of this section holds more generally for processes where the components of x are independent.



(a) Inverse covariance matrix $\boldsymbol{J_x}$

(b) Graphical model

Figure 3-2: Diagonal inverse covariance matrix and corresponding graphical model (a graph with M disjoint nodes). For the matrix, relative values of the elements are plotted on a dB-scale.

The inverse covariance structure and the corresponding graphical model are shown in Figure 3-2.

A widely occurring case in practice is that $\boldsymbol{x}$ contains successive samples of an independent time-series. Successive vectors in this case are often generated by a tapped delay line. In that case, $\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots$ are not independent realizations of $\boldsymbol{x}$ (as the same time-series sample appears in multiple realizations). In the development of algorithms and description of the processes, it is assumed that the realizations are independent, although in the implementations for real-world applications of Sections 4.5 and 5.5, the inputs are generated using a tapped delay line, as they would conventionally be.

This model is simply saying that the different components of the system $\boldsymbol{h}_0$ operate on independent inputs. Although the corresponding graph is very simple, exploiting the graph structure is not trivial, and the precise mechanics of how the graph should be exploited will be introduced in Chapters 4 and 5.

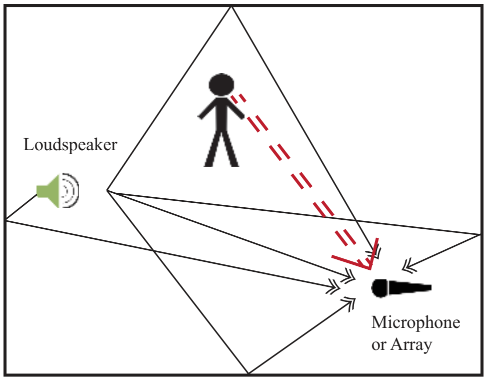


Figure 3-3: Echoes in a room environment

3.2.1 Room Impulse Response Estimation for Acoustic Echo Cancellation

The simple white input process model is applicable in a variety of fields. One important application is *room impulse response* estimation and tracking [72], which is relevant in many problems including sound source localization [52], [145], sound field reproduction [19] and perception of 3d audio [157]. In this work, the application of initializing acoustic echo cancellation (AEC) filters [24] is considered.

The general problem of echo in a room environment is illustrated in Figure 3-3. A loudspeaker plays a *known* signal that is received by a microphone through the room impulse response. The goal is to subtract the room impulse response so that the microphone is not capturing the loudspeaker signal, but instead is able to capture a clean rendition of a different *desired* signal in the room (in this case, shown by the red arrows from the person in the room).

While many modern applications such as gaming systems controlled by voice like the Microsoft XBox [152] or in-car audio [36] use a microphone array rather than a single microphone, and can then exploit array geometry to obtain accurate estimates of the room impulse response [61], [187], the size and cost of an array sometimes make it necessary to use a single microphone [164].

For a single microphone and loudspeaker, a block diagram of the AEC problem is shown in Figure 3-4. The unknown acoustic impulse response between the loudspeaker and the microphone is the system $\boldsymbol{h}_0$ that we wish to identify. Its length is given by $M = T_{\text{rev}} f_s$, where T_{rev} is the reverberation time of the room and f_s the sampling time. This is usually

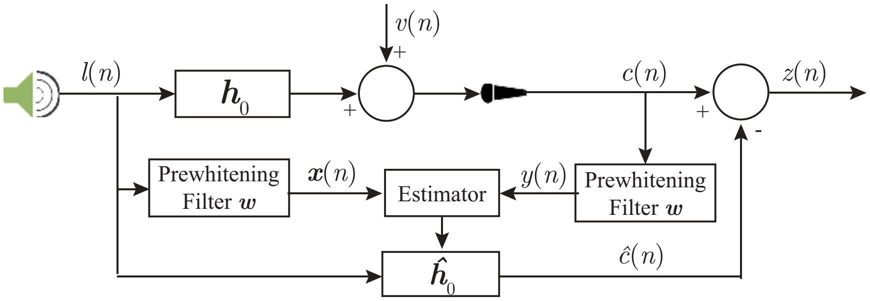


Figure 3-4: Block diagram of the echo cancellation system

quite large—a few hundred to a few thousand taps is typical.

To identify the system, a known calibration tone $l(n)$ is sent from the loudspeaker, and is received at the microphone as $c(n)$. Typically, the classical prewhitened *matched filter* [6], [175] is used to compute the initial estimate of the AEC filter, which is the MMSE estimator of the room impulse response. The prewhitening filter $\boldsymbol{w}$ gives the estimation (and tracking) algorithms access to a signal $x(n)$ whose statistics are approximately white. For consistency the received signal $c(n)$ is also filtered by the whitening filter coefficients $\boldsymbol{w}$ to give the output signal $y(n)$ to the estimation algorithm. Then, $\boldsymbol{x}(n)$ and $y(n)$ obey the linear system model of (2.1), where each $\boldsymbol{x}(n), n = 1, 2, \ldots, N$ contains M values of the signal $x(n)$ (the vectorize block collects M samples into a vector).

Thus, using the same prewhitening filter, and replacing the matched filter estimator with a graph structured estimator using a graph with M independent nodes, it is shown that it is possible to improve the quality of the initial AEC filter. The measure of quality in this application is the Echo Return Loss Enhancement (ERLE), measured by

$$\text{ERLE} = 10\log_{10}\left(\frac{\frac{1}{L}\sum_{n=1}^{L}|c(n)|^2 - \sigma^2}{\frac{1}{L}\sum_{n=1}^{L}|z(n)|^2 - \sigma^2}\right), \tag{3.2}$$

where L is the total length of the received signal $y(n)$, and $z(n) = c(n) - \hat{c}(n)$ is the received signal with the echo removed. L depends on N, but the exact depends on how the signal is vectorized. The noise power σ^2 is estimated in practice from a noise-only segment. The ERLE is measured after the filter weights have been estimated and one is attempting echo cancellation.

The ERLE is a measure of how much echo is remaining in the system after canceling out the best available estimate of the reverberant signal. Larger values of ERLE mean that more echo has been removed from the received signal. It is shown that ERLE improvements are possible by exploiting the fact that the input signal has graph-structure, especially for very short calibration signals, which would allow the length of the calibration period to be reduced.

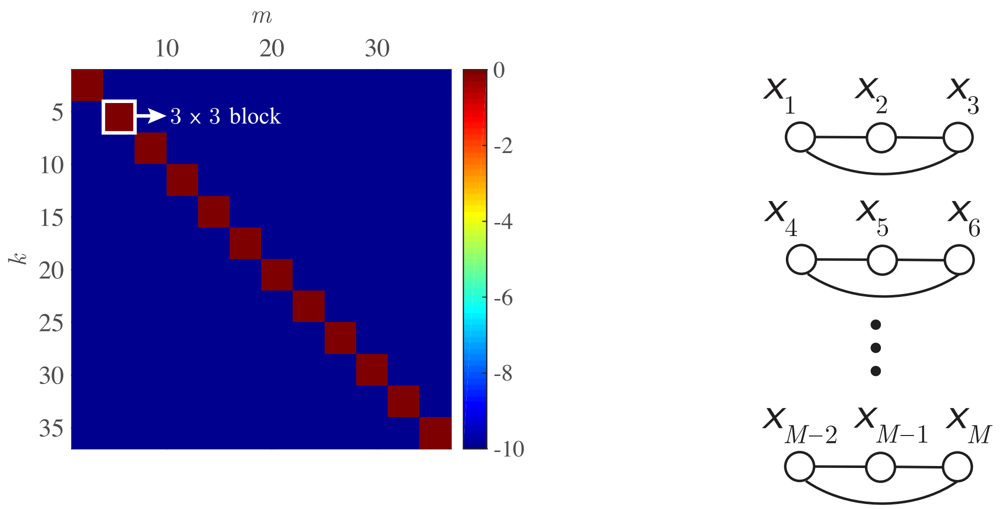
Generally, the impulse response varies over time due to the movement of objects in the room and changing environmental conditions, so the impulse response needs to be tracked over time. Due to the size of the filters involved, this is generally done with some variant of the NLMS algorithm [14], [54]. As the problem then fits into the tracking framework of (2.18), the RAGS-RLS tracking algorithm (see Section 5.4) can be used to track the impulse response using the same trivial graphical model of M independent nodes. Section 5.5 shows that, for this simple graph, RAGS-RLS can match or outperform NLMS (depending on the regime) for tracking AEC filters, and is slightly faster.

3.3 Input Processes with Block-Diagonal Covariance Matrices

Consider a system identification problem (2.1) wherein the inputs $\boldsymbol{x}(n)$ are independent realizations of $\mathsf{x} \sim \mathcal{CN}(\mathbf{0}\,,\,\boldsymbol{R}_{\mathsf{x}})$, where $\boldsymbol{R}_{\mathsf{x}}$ is now *block-diagonal*, rather than diagonal as in Section 3.2, where the size of each block is T. Then, the inverse covariance matrix is also block diagonal, and the corresponding graphical model characterizing x is a model with a set of unconnected cliques, each corresponding to one of the blocks of the inverse covariance matrix. There are M/T disjoint cliques of size T if M is a multiple of T (otherwise, either the first or last clique is smaller depending upon the system). The structure of the inverse covariance matrix and corresponding graphical model for $T = 3$ is shown in Figure 3-5.

3.3.1 Fractionally-Spaced Channel Identification

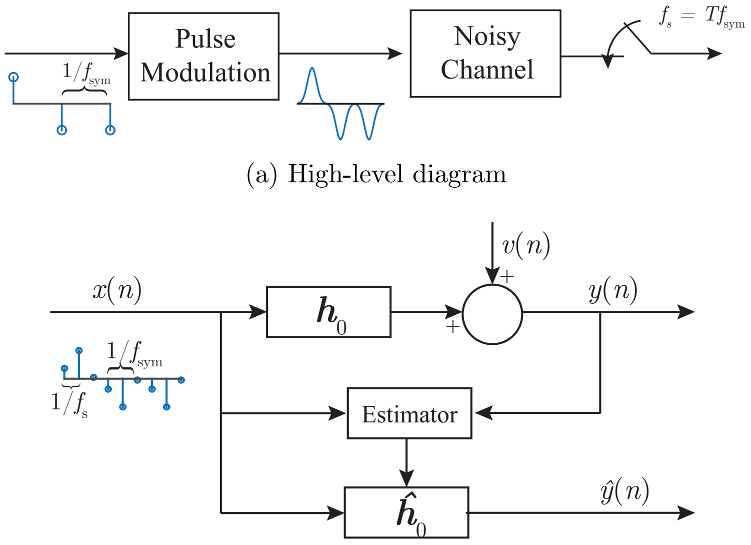
To see where such a model could be useful, consider the problem of *fractionally-spaced channel identification*, a fundamental problem in communication systems [3], [51], [142].



(a) Inverse covariance matrix $\boldsymbol{J_x}$

(b) Graphical model

Figure 3-5: Block-diagonal inverse covariance matrix and corresponding graphical model for the case when each block has size $T = 3$. The graph has M/T disjoint cliques of size T. For the matrix, relative values of the elements are plotted on a dB-scale.



(a) High-level diagram

(b) Equivalent discrete-time block diagram of estimator

Figure 3-6: Conceptual diagrams of the fractionally-spaced channel identification system. In this case, $T = 2$. Successive samples of $x(n)$ are generated using a known upsampling filter with an impulse response of length T.

A high-level illustration of this problem is shown in Figure 3-6a. For the purpose of channel identification, a pilot sequence of independently generated complex Gaussian distributed symbols is modulated and transmitted across an unknown channel. To introduce time-diversity, the sampling rate at the receiver f_s is usually an integer-multiple of the symbol rate f_{sym}, i.e., $f_s = T f_{\mathrm{sym}}$.

The equivalent discrete-time system at a sampling frequency f_s is shown in Figure 3-6b. The objective of the estimator is to estimate the unknown channel $\boldsymbol{h}_0$, whose time-domain taps are separated by $1/f_s$. The length of the unknown channel is M, where for simplicity M is assumed to be an integer multiple of T.

The channel output $y(n)$ is given by (2.1), where $\boldsymbol{x}(n)$, for $n = 1, 2, \ldots, N$ consists of the input $x(n)$ of Figure 3-6b arranged into a block of length M. To the unknown system, therefore, each input vector $\boldsymbol{x}(n), n = 1, 2, \ldots, N$ to the unknown channel is a realization of a complex Gaussian random vector $\mathbf{x}$ whose covariance matrix is block-diagonal with $T \times T$ blocks. This is because of the assumption that symbols are independent, and there are T samples corresponding to each symbol which are mutually correlated. These T elements of the vector are uncorrelated with any other element of the vector.

Thus, a graph-structured system identification algorithm could be used with a graph consisting of M/T unconnected cliques, where each clique has T nodes. Such a method is shown in Section 4.5.1 to have good performance for very short pilot sequences, allowing a reduction in the necessary pilot sequence length, in a similar manner to the reduction in calibration time that was discussed for the echo cancellation problem.

In the context of this time-domain channel identification problem, it was assumed that symbols are Gaussian distributed and mutually independent. When pilot sequences are being chosen for the sole purpose of obtaining a channel estimate, it is possible to choose the training symbols to meet these requirements. However, following that period, symbols are information bearing and whether they are mutually independent or not may not always be known. Additionally, they may not always be jointly Gaussian.

Typically, it is reasonable to assume that the sequence of information bearing symbols is a realization of a wide-sense stationary process. In that case, the sequence of transmitted symbols in the equivalent discrete-time model at the higher sampling frequency (as shown in Figure 3-6b) is *cyclostationary*.

In the next section, it is shown that wide-sense cyclostationary processes can be approximated by a Gaussian graphical model in the *frequency domain*. Thus, with a little additional signal processing, it will still be possible to exploit graphical model structure.

3.4 Cyclostationary Input Processes

A zero-mean complex-valued random process $\mathsf{x}_p, p \in \mathbb{Z}$, is defined to be wide-sense cyclostationary with period T if, for all $l \in \mathbb{Z}$,

$$R_{\mathsf{x}}(p,q) \triangleq \mathbb{E}\left[\mathsf{x}_p \mathsf{x}_q^*\right] = \mathbb{E}\left[\mathsf{x}_{p+lT} \mathsf{x}_{q+lT}^*\right] \, . \tag{3.3}$$

It should be noted that the class of wide-sense cyclostationary random processes encompasses wide-sense stationary processes as well, which can be verified by substituting $T = 1$ into (3.3).

Since the pioneering works of Hurd and Gardner in the '60s and '70s [67], [70], [87], cyclostationary processes have found application in fields ranging from communication systems, acoustics and circuits to econometrics and biology. The comprehensive review work by Gardner, et al., [68] lists a number of references in various application areas of these signals. They have thus been a mainstay of signal processing in a variety of domains for over half a century.

For cyclostationary distributions, the joint distribution $p_{\ldots,\mathsf{x}_1,\mathsf{x}_2,\ldots}(\ldots, x_1, x_2, \ldots)$ is constrained by (3.3). These constraints imply that the distribution is structured in some way. However, conditional independence relationships are not immediately evident from (3.3). It is shown here that in the frequency domain, all cyclostationary processes with period T can be represented by the same undirected graphical model. For the case of wide-sense stationary processes, it will be seen that the graph reduces to a set of independent nodes.

The existence of such a graphical model has some useful implications, as has already been indicated in Section 3.3.1. For instance, in array processing and communication systems, frequency-domain processing of cyclostationary signals is quite common [59], [81], [188]. In such applications, when the underlying problem is system identification, the results of this work can be readily applied.

3.4.1 Frequency-Domain Covariance Structure

Let $\mathsf{x}_p, p \in \mathbb{Z}$ be a cyclostationary random process. Take the Discrete-Time Fourier Transform (DTFT) of the sequence, i.e., define

$$\mathsf{X}_\omega = \sum_{p \in \mathbb{Z}} \mathsf{x}_p e^{-j\omega p}, \ -\pi \leq \omega < \pi \,, \tag{3.4}$$

where the Fourier transform is assumed to exist, at least in the sense of distributions [80], for almost all sample paths of x_p. Then, it can be shown [68], [88] that

$$\begin{aligned} H(\omega, \nu) &= \mathbb{E}\left[\mathsf{X}_\omega \mathsf{X}_\nu^*\right] && \text{(3.5a)} \\ &= \sum_{t=-(T-1)}^{T-1} S_{\mathsf{x}}^t(\omega) \delta\left(\omega - \nu + \frac{2\pi}{T} t\right), \ -\pi \leq \omega, \nu < \pi \,, && \text{(3.5b)} \end{aligned}$$

where the functions $S_{\mathsf{x}}^t(\omega)$ are the so called "cyclic spectra" of the process. When $T = 1$, the process is wide-sense stationary, and $S_{\mathsf{x}}^0(\omega) = S_{\mathsf{x}}(\omega)$ is the power spectral density (PSD) of the wide-sense stationary process. In this case, (3.5b) is the familiar result that, for wide-sense stationary processes, frequency coefficients at different frequencies are uncorrelated. More generally, it says that for processes that are cyclostationary with period T, frequency coefficients are only correlated at frequency lags of integer multiples of $2\pi/T$. In other words, $H(\omega, \nu)$ has the structure shown in Figure 3-7, where it is zero everywhere except the lines $\omega - \nu = 2\pi t/T$. The function $H(\omega, \nu) = \mathbb{E}\left[\mathsf{X}_\omega \mathsf{X}_\nu^*\right]$ is sometimes called the *Loève bispectrum*[1] and has been extensively used in the study of cyclostationary signals and their generalizations [4], [122], [123], [179].

To begin with, suppose the Fourier transform is evaluated at M_f frequencies belonging to the set

$$\mathcal{F} = \left\{-\pi(M_f - 2)/M_f, -\pi(M_f - 4)/M_f, \ldots, \pi(M_f - 2)/M_f, \pi\right\} \,, \tag{3.6}$$

where M_f is assumed to be a multiple of T. In a slight abuse of notation, let X_k be the

[1] Its full name is generally used in order to avoid confusion with the "bispectra" of wide-sense stationary processes, which refers to the 3rd moment of those processes—these are not considered further here.

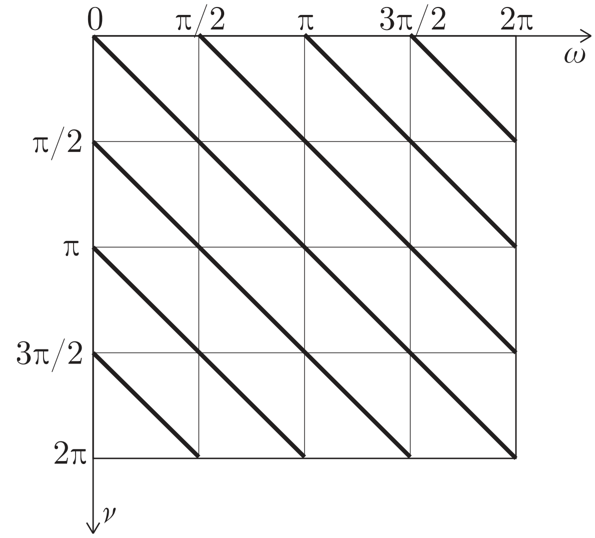


Figure 3-7: Support lines of $H(\omega, \nu) = \mathbb{E}[X_\omega X_\nu^*]$ on the (ω, ν) plane for a cyclostationary process with $T = 4$. The axes are chosen as shown to maintain consistency with covariance matrices in which the first element is on the top left. $H(\omega, \nu)$ is non-zero on the thick lines.

frequency coefficient for the frequency $\omega_k = \pi(2k - M_f)/M_f$, for $k = 1, 2, \ldots, M_f$. Then, the random vector

$$\boldsymbol{X} = [X_1, X_2, \cdots, X_{M_f}] \tag{3.7}$$

has a covariance matrix $\boldsymbol{R_X} = \mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\dagger]$ which comprises of the function $H(\omega, \nu)$, sampled at M_f^2 equally spaced points in the (ω, ν) plane. Thus, provided that M_f is chosen to be a multiple of T, the structure of Figure 3-7 is also manifest in the covariance matrix of $\boldsymbol{X}$.[2] The matrix $\boldsymbol{R_X}$ is termed the *frequency-domain covariance matrix* of the process. Element (i, k) of the frequency-domain covariance matrix for a cyclostationary process is zero whenever $i - k \neq pM_f/T$ for $p = -(T-1), \ldots, (T-1)$.

Figure 3-8 shows the frequency-domain covariance matrix of a cyclostationary signal with $T = 4, M_f = 64$ (specifically, a PAM signal with 4 samples/symbol, which is cyclostationary [67]). Clearly, the covariance matrix elements are zero, except for the entries $R_{\boldsymbol{X}_{i,k}}$ where $i - k$ is an integer multiple of M_f/T.

[2] When M_f is not a multiple of T, it can be verified that the frequency-domain covariance matrix is obtained by convolving $H(\omega, \nu)$ by a 2-d discrete-time sinc function—this is due to sampling with a "fractional lag" in the frequency domain.

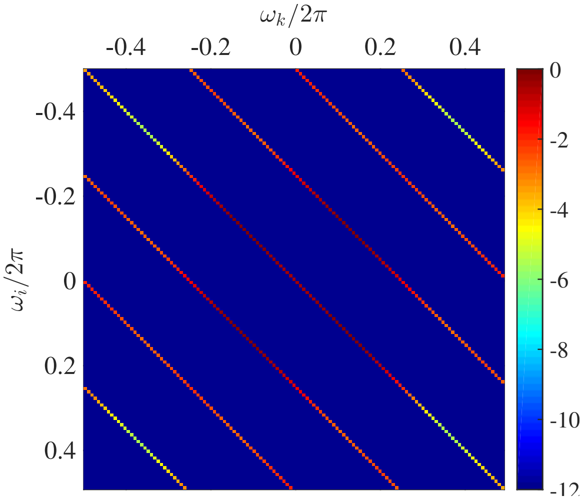


Figure 3-8: Frequency-domain covariance matrix for a cyclostationary signal. The figure shows the relative magnitudes of the elements (normalized by the largest matrix element) on a dB-scale. $R_{\boldsymbol{X}_{i,k}} = 0$ when $i - k \neq pM_f/T$, where $p = -(T-1), \dots, (T-1)$. In this case, $M = M_f = 64$ and $T = 4$.

3.4.2 Dropping Frequencies

Thus far, it has been assumed that the vector $\boldsymbol{X}$ contains the frequency coefficients for every frequency of $\mathcal{F}$, where $\mathcal{F}$ was defined in (3.6). However, the structured sparsity of the covariance matrix remains even if a consecutive subset of the frequencies is chosen. Thus, the vector $\boldsymbol{X}$ can contain frequency coefficients for some consecutive subset of $\mathcal{F}$. This is useful in cases where not all frequencies carry useful information and some subset of frequencies is sufficient for processing.

It is assumed henceforth that M refers to the number of retained frequencies. Note that M does not have to be a multiple of T, but M_f does. The vector $\boldsymbol{X}$ is now defined as the M-vector of retained frequencies. It is important to note that in this case, X_k is the Fourier transform of the process at the frequency $\omega_k = \omega_{\text{first}} + 2\pi(k-1)/M_f$, where $\omega_{\text{first}} \in \mathcal{F}$ is the first retained frequency.

3.4.3 Frequency-Domain Inverse Covariance Matrix Structure

The covariance matrix of the random vector $\boldsymbol{X}$ comprising M equally spaced frequency coefficients of a cyclostationary process has been shown thus far to have a structured sparsity. In the following, the *inverse* of this covariance matrix is shown to exhibit the same structured sparsity.

Theorem 3.4.1. *Let $\Delta \geq 2$ be a positive integer.*[3] *Let $\mathcal{M}_\Delta^M$ be the set of invertible $M \times M$ matrices such that if $\boldsymbol{A} \in \mathcal{M}_\Delta^M$, then $A(k,m) = 0 \, \forall \, k,m$ such that $k - m \neq r\Delta, r \in \mathbb{Z}$.*

Then $\mathcal{M}_\Delta^M$ is closed under matrix inversion.

Proof. Let $\boldsymbol{A} \in \mathcal{M}_\Delta^M$ and $s \in \{1, 2, \dots, M\}$ be a row index. It is simple to show from the second condition above that only non-zero elements of $\boldsymbol{A}$ in row s are $A(s, s + w\Delta)$, where

$$w \in \left\{ \left\lceil \frac{1-s}{\Delta} \right\rceil, \dots, \left\lfloor \frac{M-s}{\Delta} \right\rfloor \right\} .$$

Fix some w_0 in the range above. Then, for any row $s' \neq s$, $A(s', s + w_0\Delta) \neq 0 \implies s' - (s + w_0\Delta) = w'\Delta$, which simply means that $s' - s$ is an integer multiple of Δ.

Hence, divide $\{1, 2, \dots, M\}$ into Δ partitions $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_\Delta$, where

$$\mathcal{P}_i = \left\{ i, i + \Delta, i + 2\Delta, \dots, i + \left\lfloor \frac{M-i}{\Delta} \right\rfloor \Delta \right\} .$$

By the argument above, only rows in $\mathcal{P}_i$ possibly have non-zero elements in the same columns.

Now consider using Gauss-Jordon elimination to invert this matrix (as $\mathcal{M}_\Delta^M$ has *invertible* matrices, this inverse matrix exists). The augmented matrix has a form that looks like:

$$\begin{array}{c|cccccc|cccccc|}
 & 1 & 2 & \cdots & \Delta+1 & \Delta+2 & \cdots & 1 & 2 & & \cdots & & M \\
\hline
1 & \mathtt{x} & 0 & \cdots & \mathtt{x} & 0 & \cdots & 1 & 0 & & \cdots & & 0 \\
2 & 0 & \mathtt{x} & \cdots & 0 & \mathtt{x} & \cdots & 0 & 1 & & \cdots & & 0 \\
\vdots & & & & \vdots & & & & & \vdots & & & \\
\Delta+1 & \mathtt{x} & 0 & \cdots & \mathtt{x} & 0 & \cdots & 0 & 0 & \cdots & 1 & \cdots & 0 \\
\vdots & & & & \vdots & & & & & \vdots & & & \\
M & 0 & & \cdots & & 0 & \mathtt{x} & 0 & 0 & & \cdots & & 1 \\
\hline
\end{array}$$

where $\mathtt{x}$ refers to matrix elements that may be non-zero.

It should be clear that to reduce a matrix with the above structure to the row-echelon

[3] $\Delta = 1$ gives the trivial case of a fully populated matrix

form, elementary row operations need only be carried out on the rows within each partition $\mathcal{P}_i$. For instance, rows 1 and $\Delta + 1$ both belong to $\mathcal{P}_1$, and might be added together while reducing the matrix to row-echelon form. However, we would never add a multiple of row 2, which belongs to $\mathcal{P}_2$, to either of these rows (or swap row 2 with any of rows from $\mathcal{P}_1$), as such an operation could not help us reduce the matrix on the left to the identity matrix.

Now, looking at the identity matrix on the right, if all row operations are to be carried out within a particular partition, then by reversing the logic laid out above, any row in the resultant right side matrix (call it $\boldsymbol{B} = \boldsymbol{A}^{-1}$) also satisfies the condition that the only non-zero elements in row t are $B(t, t + w\Delta)$, i.e., $\boldsymbol{B} \in \mathcal{M}_{\Delta}^{M}$. □

It is simple to see that the sparse structure derived in Section 3.4.1 is equivalent to the statement $\boldsymbol{R}_{\boldsymbol{X}} \in \mathcal{M}_{Mf/T}^{M}$. Then Theorem 3.4.1 shows that the frequency-domain inverse covariance matrix also has exactly the same kind of sparse structure, i.e., $\boldsymbol{J}_{\boldsymbol{X}} \in \mathcal{M}_{Mf/T}^{M}$, where $\boldsymbol{J}_{\boldsymbol{X}} = \boldsymbol{R}_{\boldsymbol{X}}^{-1}$.

The conclusion that the inverse covariance matrix of a cyclostationary process shares the same structure as the covariance matrix is verified in Figure 3-9, which represents the inverse covariance matrix for the cyclostationary process of Figure 3-8. It is clear from the simulation that the inverse covariance matrix has the same structure as the covariance matrix in the frequency domain. This result seems quite counter-intuitive, as it is not usually the case that matrix inversion preserves structure—however, in this case, it does.

3.4.4 Joint Gaussianity of Frequency Coefficients

In the diagonal (white) and block-diagonal (fractional spaced sampling) covariance processes of Sections 3.2 and 3.3, it was assumed that the processes were all Gaussian. Such an assumption has not been made thus far for cyclostationary processes in this section. The inverse covariance matrix of the frequency coefficients of wide-sense cyclostationary processes has been shown to have structure regardless of the joint distribution of the random variables x_p in (3.3).

If it could also be argued that the frequency coefficients are jointly Gaussian, then Section 2.4.1, and in particular (2.28), would imply that the frequency coefficients are characterized

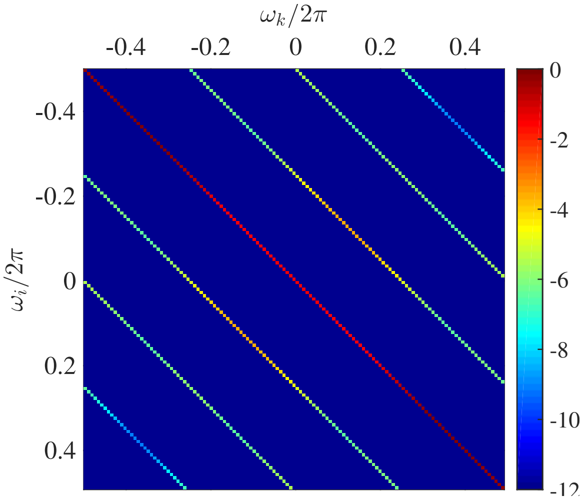


Figure 3-9: Frequency-domain inverse covariance matrix for the cyclostationary signal of Figure 3-8. The figure shows the relative magnitudes of the elements (normalized by the largest matrix element) on a dB-scale. The matrix has the same structure as the covariance matrix of Figure 3-8. As in Figure 3-8, $M = M_f = 64, T = 4$.

by a graphical model.

For stationary processes, joint Gaussianity of frequency coefficients has been established via a central limit type argument [135], [184], [185]. The proofs of these works are not reproduced here, but the intuition behind them is that the frequency coefficients are obtained by summing a relatively large number of random variables multiplied by orthogonal complex exponential functions, which leads to a central limit theorem for wide-sense stationary processes.

Using this result, it is relatively simple to see that frequency coefficients for a cyclostationary process are asymptotically *marginally* Gaussian, because

$$\begin{aligned} \mathsf{X}_k &= \sum_{p\in\mathbb{Z}} \mathsf{x}_p e^{-j\omega_k t} \\ &= \sum_{t=-(T-1)}^{T-1} e^{-j\omega_k t} \sum_{q\in\mathbb{Z}} \mathsf{x}_q^t e^{-jT\omega_k q}\,, \end{aligned} \tag{3.8}$$

where x_q^t is a subsequence of x_p, given by $\mathsf{x}_q^t = \mathsf{x}_{t+qT}$. It is simple to verify that the subsequences are all stationary, and thus each has an asymptotically Gaussian Fourier transform. As X_k is a linear combination of Gaussian random variables, it is also asymptotically marginally Gaussian. It may be possible to generalize this result to show that small groups of frequency coefficients are asymptotically jointly Gaussian, although a formal proof has not been found in the literature.

However, this does not immediately imply that the random vector $\boldsymbol{X}$ is multivariate Gaussian. A central limit theorem that encompasses cyclostationary processes does not appear to have been proven, and it is not clear that one exists.

Nonetheless, it is assumed herein that the frequency coefficient vector $\boldsymbol{X}$ is a multivariate Gaussian vector. The rationale for doing so is that in this work, the vector $\boldsymbol{X}$ serves as the input to a linear system identification problem; as discussed in Section 2.1.2, a statistical view of this problem justifies Gaussianity as a reasonable assumption for the input to the channel. This makes the graphical model representation approximate.

Note, of course, that if the time-domain process $\boldsymbol{x}_p$ is a Gaussian process, then the frequency coefficients are jointly Gaussian without any approximations or asymptotic considerations.

3.4.5 Gaussian Graphical Model for Cyclostationary Processes

Assuming that the random vector $\boldsymbol{X}$ is indeed multivariate complex normal, then the structured sparsity of the inverse covariance matrix established in 3.4.3 implies that a graphical model characterizes the M frequency domain random variables $X_1, X_2, \ldots, X_M$. The cliques of the graph can be inferred from the sparsity pattern of the inverse covariance matrix, as explained in Section 3.1 and previously exemplified in Figure 3-2 and 3-5.

To concisely denote which nodes are connected in this case, it is helpful to define $\Delta = M_f/T$ and $m(t) = \lfloor (M-t)/\Delta \rfloor$. Then the graph has $\Delta = M/T$ disjoint cliques each of size $\mathcal{O}(T)$, where the exact size of each clique depends on which frequencies are retained. For $M = M_f$, i.e., when all the frequencies are retained, the form of the graph is shown in Figure 3-10.

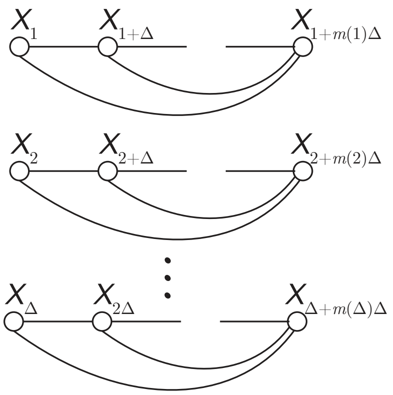


Figure 3-10: Gaussian graphical model for cyclostationary process in the frequency domain. $\Delta = M_f/T$ and $m(t) = \lfloor (M-t)/\Delta \rfloor$, and for this example, $M = M_f$.

3.4.6 Multichannel Adaptive Frequency-Domain Equalization

To see how the graphical model derived in this section can be used in a system identification application, the problem of adaptive multichannel frequency-domain equalization in communication systems is considered. Frequency-domain equalization has been particularly useful in the context of wireless underwater acoustic communication channels [141], [188], although the set up of this section applies to wireless communication systems in general. This section also serves as a blueprint for the processing required to exploit the frequency domain graphical model structure in other applications.

The block diagram of a multichannel frequency-domain adaptive equalizer is shown in Figure 3-11. The modulated signal, denoted $y(n)$, is transmitted across noisy, time-varying channels that cause intersymbol interference. The signal is received at R spatially diverse receivers. Note that the equivalent discrete-time transmitted signal $y(n)$ is generated by a fractionally sampled system similar to that of Figure 3-6a. However, in Section 3.3.1 it was assumed that the information sequence was independent and Gaussian. Both those restrictions are relaxed here—all that is required is that the information sequence is wide-sense stationary (the information sequence is not shown in Figure 3-11—$y(n)$ is the equivalent discrete-time transmitted signal obtained after modulation of the information sequence).

The tracking algorithm in Figure 3-11 attempts to estimate and track the equalizer coefficients that filter the combined frequency-domain channel output (this will be defined momentarily) to estimate the transmitted signal. This is an inverse problem, so the input

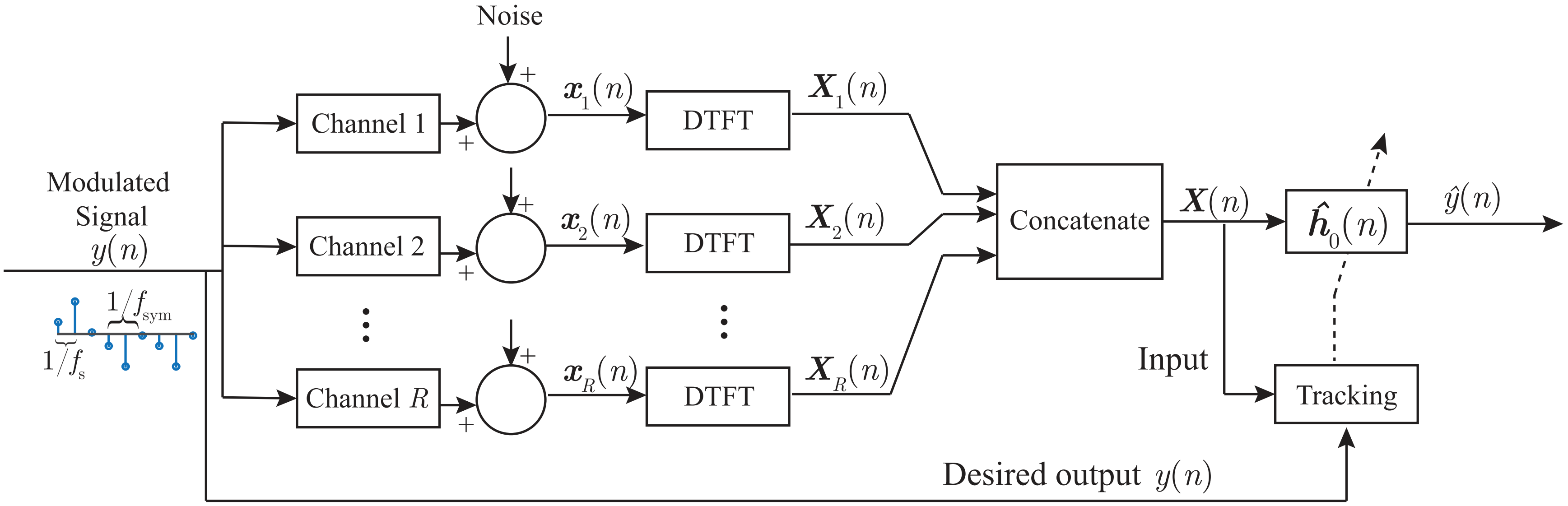


Figure 3-11: Block Diagram of Frequency-Domain Equalizer. This is an example of an inverse problem, so the input and (desired) output of the tracking problem are explicitly marked to avoid confusion. $\boldsymbol{x}_r(n)$ contains samples of the received signal from channel r for $r = 1, 2, \ldots, R$. $\boldsymbol{X}_r(n)$ contains M_0 frequency coefficients separated by $2\pi/M_f$, where in this case $M_f = M_t$. $\boldsymbol{X}(n)$ contains the $M = RM_0$ frequency coefficients for all the channels as defined in (3.17).

and (desired) output of the system to be tracked are explicitly noted on the figure.

If the information sequence is wide-sense stationary, then the transmitted signal $y(n)$ is wide-sense cyclostationary. If additionally it can be assumed that the channel variations are wide-sense stationary, which is usually a reasonable assumption over the typical time-scales of interest for communication for both time-varying wireless communication channels [112] and underwater acoustic communication channels [172], then the received signal is cyclostationary, as shown below.

Theorem 3.4.2. *When a cyclostationary signal with period T is passed through a time-varying channel with wide-sense stationary channel variations, the received signal is cyclostationary with period T.*

Proof. Suppose a Wide-Sense stationary signal $\mathsf{s}(n)$ is passed through a Randomly Time-Variant Linear (RTVL) channel $\mathsf{h}(n,p)$. Thus, the received signal $\mathsf{x}(n)$ is given by

$$\mathsf{x}(n) = \sum_{p} \mathsf{h}^*(n,p)\mathsf{s}(n-p) + \mathsf{v}(n)\,, \tag{3.9}$$

where $\mathsf{h}(n,p)$ represents the time-varying complex baseband impulse response of the channel, and $v(n)$ is assumed to be zero-mean Wide-Sense Stationary noise, with $R_{\mathsf{v}}(p) = \mathbb{E}\left[v(n+p)v^*(n)\right]$. Additionally, we assume that the channel variations are independent of the signal.

Define the channel correlation function

$$R_{\mathsf{h}}(n,m;p,q) = \mathbb{E}\left[\mathsf{h}(n,p)\mathsf{h}^*(m,q)\right]\,. \tag{3.10}$$

Recall that, for the particular case that the channel is Wide-Sense Stationary (WSS), the channel correlation function is invariant under a translation in time, i.e.,

$$R_{\mathsf{h}}(n,m;p,q) = R_{\mathsf{h}}(n-m;p,q)\,. \tag{3.11}$$

There is no assumption that it is necessarily invariant under a lag.

The covariance function of the received signal is given by

$$\begin{aligned}
R_{\mathsf{x}}(n,m) &= \mathbb{E}\left[\mathsf{x}(n)\mathsf{x}^*(m)\right] \\
&= \mathbb{E}\left[\left(\sum_p \mathsf{h}^*(n,p)\mathsf{s}(n-p)+\mathsf{v}(n)\right)\left(\sum_q \mathsf{h}^*(m,q)\mathsf{s}(m-q)+\mathsf{v}(m)\right)^*\right] \\
&= \sum_p \sum_q \mathbb{E}\left[\mathsf{h}(n,p)\mathsf{h}^*(m,q)\right]^* \mathbb{E}\left[\mathsf{s}(n-p)\mathsf{s}^*(m-q)\right] + \mathbb{E}\left[\mathsf{v}(n)\mathsf{v}^*(m)\right] && \text{(3.12a)} \\
&= \sum_p \sum_q R_{\mathsf{h}}^*(n,m;p,q) R_{\mathsf{s}}(n-p,m-q) + R_{\mathsf{v}}(n-m)\,, && \text{(3.12b)}
\end{aligned}$$

where (3.12a) is due to the assumed independence between the transmitted signal variations and channel variations; and (3.12b) exploits the fact that the noise is WSS.

Now, when the channel is Wide-Sense Stationary [13, IV-B]

$$R_{\mathsf{x}}(n,m) = \sum_p \sum_q R_{\mathsf{h}}^*(n-m;p,q) R_{\mathsf{s}}(n-p,m-q) + R_{\mathsf{v}}(n-m)\,. \tag{3.13}$$

It follows from (3.13) that, if the transmitted signal is cyclostationary with period T, then the received signal is also cyclostationary with period T, since

$$\begin{aligned}
R_{\mathsf{x}}(n+T,m+T) &= \sum_p \sum_q R_{\mathsf{h}}^*(n+T-m-T;p,q) R_{\mathsf{s}}(n+T-p,m+T-q) + R_{\mathsf{v}}(n+T-m-T) \\
&= \sum_p \sum_q R_{\mathsf{h}}^*(n-m;p,q) R_{\mathsf{s}}(n-p,m-q) + R_{\mathsf{v}}(n-m) && \text{(3.14)} \\
&= R_{\mathsf{x}}(n,m)\,. && \text{(3.15)}
\end{aligned}$$

Thus, the received signal is also cyclostationary with period T. □

The output of each receiver of Figure 3-11 is converted into blocks[4] $\boldsymbol{x}_r(n)$, $r = 1,2,\ldots,R$, at each time $n = 1,2,\ldots,N$. Due to Theorem 3.4.2, each block is a realization of a cyclostationary process whose period is the number of samples per symbol of the transmitted signal, i.e., $T = f_s/f_{\text{sym}}$.

[4] For now, each block is assumed to be infinitely long; the effect of finite block length is considered in Section 3.5.2.

The Fourier transform of each block $\boldsymbol{x}_r(n)$ is taken to produce the corresponding frequency-domain realization $\boldsymbol{X}_r(n)$. The Fourier transform is evaluated at frequencies separated by $2\pi/M_f$. M_0 consecutive frequencies are retained.[5] These M_0-length vectors $\boldsymbol{X}_r(n)$ are then concatenated into an overall vector $\boldsymbol{X}(n)$ of length $M = RM_0$, where

$$\boldsymbol{X}(n) = [X_{11}(n), X_{21}(n), \cdots, X_{R1}(n),\ X_{12}(n), X_{22}(n), \cdots, X_{R2}(n), \cdots, \tag{3.16}$$

$$X_{1M_0}(n), X_{2M_0}(n), \cdots, X_{RM_0}(n)]^T, \tag{3.17}$$

where X_{rm} refers to the mth retained frequency of channel r and the superscript ^{T} refers to matrix transpose.

Having defined these quantities, the problem is now simple to state. The objective is to track the unknown coefficients of a linear equalizer which, taking the frequency-domain channel output as an input, produces the transmitted signal $y(n)$ at the output. This is an *inverse* problem, as illustrated in Figure 1-1b. The "true" set of equalizer coefficients are unknown, but the ideal set of equalizer coefficients $\boldsymbol{h}_0(n)$ are those that relate $\boldsymbol{X}(n)$ to the transmitted symbol $y(n)$ through (2.18).

Typically, an RLS algorithm would be used with $\boldsymbol{X}(n)$ as the input and $y(n)$ as the output (especially in the context of wireless underwater communication systems). However, as $\boldsymbol{x}_r(n)$ is a realization of a cyclostationary process, the frequency domain vector $\boldsymbol{X}_r(n)$ can be thought of as a realization of a random variable $\mathsf{\boldsymbol{X}}_r$ which is characterized by a graphical model such as that of Figure 3-10. Thus, the multichannel vectors $\boldsymbol{X}(n)$, $n = 1, 2, \ldots, N$, can in turn be viewed as realizations of a graph-structured random vector $\mathsf{\boldsymbol{X}}$.

The graph for the multichannel vector $\mathsf{\boldsymbol{X}}$ is obtained simply by connecting the cliques for a particular frequency and its $2\pi/T$ offsets for all the channels. Defining $\Delta = M_f/T$ and $m(t) = \lfloor (M_0 - t)/\Delta \rfloor$ (recall that the frequency spacing is $2\pi/M_f$ and the number of retained frequencies per channel is M_0), the graph for $R = 2$ is shown in Figure 3-12. The thick lines in Figure3-12 are just a concise way of representing an edge between every pair

[5] Note that, in Section 3.4.2, the number of retained frequencies was taken to be M, whereas now, M_0 frequencies are retained and $M = RM_0$ is the total length of the multichannel vector $\boldsymbol{X}(n)$. The reason for this change in notation is that in this case, $\boldsymbol{X}(n)$ is the input to the adaptive algorithm and throughout this work, this is assumed to have length M.

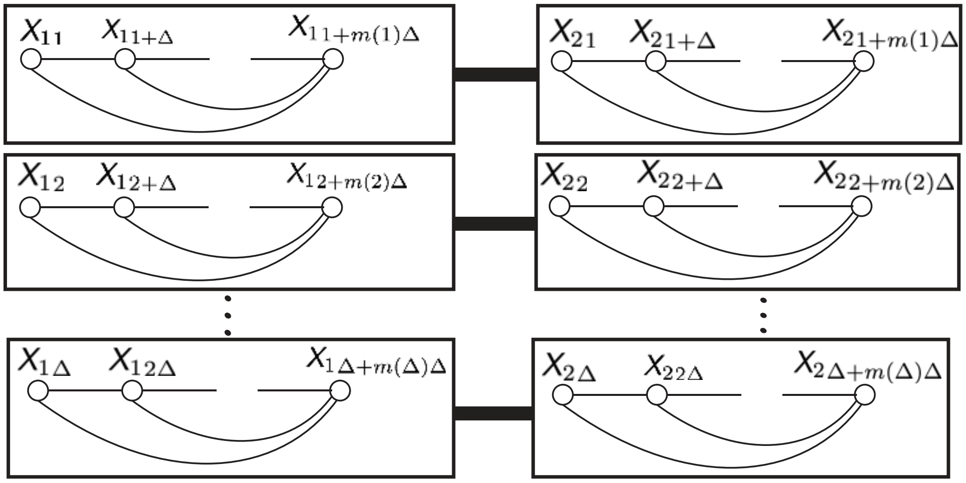


Figure 3-12: Frequency domain graphical model for multichannel cyclostationary processes. $\Delta = M_f/T$, where the frequency spacing is $2\pi/M_f$, and $m(t) = \lfloor (M_0 - T)/\Delta \rfloor$. The variable X_{rm} refers to the mth retained frequency for the rth channel, where $m = 1, 2, \ldots, M_0$, $r = 1, 2, \ldots R$ and $M = RM_0$ is the total number of nodes in the graph. For this graph, $R = 2$. The graph is simply obtained by connecting the cliques corresponding to the same frequencies for all the channels. The thick black lines in the diagram mean that all the nodes inside the boxes joined by the lines are connected to one another.

of nodes of the cliques connected by the line. This is done to avoid having to draw each of the edges, which would make the graph unreadable.

By using this graphical model with the RAGS-RLS algorithm, excellent results have been obtained for adaptive equalization in wireless underwater communication, which are presented in Section 5.5.2.

A few details have been omitted in the description above. For instance, nothing has been said about how long the time-domain blocks $\boldsymbol{x}_r(n)$ should be; indeed, throughout this section, it has been assumed that the Fourier transform has infinite length. This is, of course, unrealistic. Additionally, the graphical model structure described in all the above applications is ensemble structure, and the question may reasonably arise—does the structure in a finite dataset match the ensemble structure? These issues are explored in the next section.

3.5 The Effect of Limited Data on Structure

A number of applications have been introduced in Sections 3.2—3.4 in which a system identification problem has inputs that are characterized by a Gaussian graphical model. In all of the applications, the graphical model stemmed from a sparse structured inverse covariance matrix, as explained in Section 2.4.1. The sparsity pattern of the inverse covariance matrix of a random vector is indicative of conditional independences relationships amongst the components of the random vector.

In this section, a related but different problem is considered—are the marginal independence relationships represented by a covariance matrix and the conditional independence relationships represented by an inverse covariance matrix truly observed in a dataset with limited data?

This may appear to be a slight detour from the main purpose of this thesis, which is to exploit graphical model structure in system identification. However, in this chapter, various datasets that are characterized by structured sparse inverse covariance matrices have been considered. It is instructive to understand what happens to the structure in those datasets when the datasets are small, as they typically are in the applications of interest. If the structure in those datasets deviates from the expected structure, it may be worth attempting to capture the expected limited sample structure rather than persisting with the ensemble structure, because, in any given dataset the limited sample structure is what might truly be observed.

There is a long history of analysis of the sample covariance and inverse covariance matrices with finite data, e.g., [120], [130], [155], [174]. Most of it is focused on the eigenvalue distribution of the matrices, which does not yield information about graphical model structure. In contrast, as is explained below, the expected absolute value of the elements of the sample covariance and inverse inverse covariance matrices reflects the correlation and their conditional correlation, respectively, of any pair of elements in a finite dataset and thus characterizes the graphical model structure of the data.

3.5.1 General Finite-Realization Results

Apparent Correlation and Conditional Correlation

As has been previously assumed, let $\boldsymbol{x}(n)$ be a realization of $\mathsf{\mathbf{x}}$, whose covariance matrix is $\boldsymbol{R}_{\mathsf{\mathbf{x}}}$. As defined in (2.9a), denote the sample covariance matrix given N samples to be $\hat{\boldsymbol{R}}_{\mathsf{\mathbf{x}}}(N)$, and note that[6] $\mathbb{E}\left[\hat{\boldsymbol{\mathsf{R}}}_{\mathsf{\mathbf{x}}}(N)\right] = \boldsymbol{R}_{\mathsf{\mathbf{x}}}$.

Suppose that, for some i, k, $R_{\mathsf{\mathbf{x}}_{i,k}} = 0$, so that x_i and x_k are uncorrelated in an ensemble sense. However, suppose for instance that $\mathbb{E}\left[\left|\hat{\mathsf{R}}_{\mathsf{\mathbf{x}}_{i,k}}(N)\right|\right] = 0.5$. Then, given N realizations of the process, the sample cross-correlation between x_i and x_k is likely to be close to ±0.5—in other words, x_i and x_k are quite likely to *appear* correlated given any N realizations of $\mathsf{\mathbf{x}}$.

Similarly, define the inverse sample covariance matrix as $\hat{\boldsymbol{J}}_{\mathsf{\mathbf{x}}}(N) = \hat{\boldsymbol{R}}_{\mathsf{\mathbf{x}}}^{-1}(N)$. If the expected absolute value of an element of the inverse sample covariance matrix $\mathbb{E}\left[\left|\hat{\mathsf{J}}_{\mathsf{\mathbf{x}}_{i,k}}(N)\right|\right]$ is large for some i, k for finite N, then given N realizations of the process, x_i and x_k may not appear to be conditionally independent (for a Gaussian process) even if $J_{\mathsf{\mathbf{x}}_{i,k}} = 0$.

From a system design point of view, the "apparent" independence and conditional independence statements are relevant. If only the ensemble conditional independences were modeled, an algorithm exploiting the model ignores dependences that may appear to be strong in any given realization of the process. Hence, a more inclusive approach is taken, namely to model apparent conditional dependences as though they were true conditional dependences.

The approach taken thus emerges. The expected absolute values of the elements of the sample covariance and inverse covariance matrix are computed, and an edge is placed in the graph for all pairs (i, k) for which the expected absolute value is larger than a chosen threshold, so that all conditional dependences that are *potentially* large in any realization are modeled while exploiting structure.

[6] Note that the sample correlation matrix $\hat{\boldsymbol{\mathsf{R}}}_{\mathsf{\mathbf{x}}}(N)$ is a random matrix as each input is a realization of a random variable.

Expected Absolute Sample Covariance

To compute $\mathbb{E}\left[\left|\hat{R}_{\mathbf{x}_{i,k}}(N)\right|\right]$, start by considering

$$\hat{R}_{\mathbf{x}_{i,k}}(N) = \frac{1}{N}\sum_{n=1}^{N} x_i(n)x_k^*(n)\,, \tag{3.18}$$

where $\mathbf{x}(n) \overset{\text{iid}}{\sim} \mathbf{x}$. The sequence $x_i(n)x_k^*(n)$ is a sequence of independent random variables; so provided N is fairly large, the Central Limit Theorem applies and $\hat{R}_{\mathbf{x}_{i,k}}(N)$ is a scalar circularly symmetric complex Gaussian random variable. Additionally, the assumption is made that the real and imaginary parts are independent[7] and have the same variance, which is denoted by $\rho_{i,k}^2(N)$.

As it has been assumed at the beginning of this section that $\mathbf{x}(n)$ is circularly symmetric complex Gaussian, the matrix $\hat{\mathbf{R}}_{\mathbf{x}}(N)$ is complex Wishart distributed, and the moments of its elements have been computed [104]. In particular, when $i \neq k$,

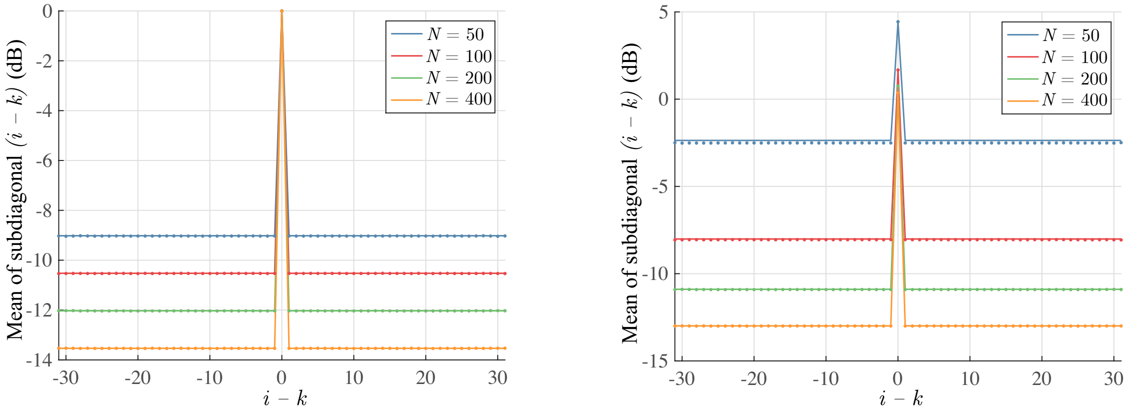
$$\mathbb{E}\left[\hat{R}_{\mathbf{x}_{i,k}}(N)\right] = R_{\mathbf{x}_{i,k}} \tag{3.19a}$$

$$\rho_{i,k}^2(N) = \frac{1}{2N} R_{\mathbf{x}_{i,i}} R_{\mathbf{x}_{k,k}}\,. \tag{3.19b}$$

When $i = k$, $\hat{R}_{\mathbf{x}_{i,k}}(N)$ is real and positive. When $i \neq k$, $\hat{R}_{\mathbf{x}_{i,k}}(N)$ is complex normal, so its absolute value is a Rician random variable [146]. Thus, the expected value of $\left|\hat{R}_{\mathbf{x}_{i,k}}\right|$ is given by

$$\begin{aligned}
\mathbb{E}\left[\left|\hat{R}_{\mathbf{x}_{i,k}}(N)\right|\right] &= \begin{cases} \mathbb{E}\left[\hat{R}_{\mathbf{x}_{i,k}}(N)\right], & \text{when } i = k \\ \rho_{i,k}(N)\sqrt{\dfrac{\pi}{2}}L_{1/2}\left(-\dfrac{\left|\mathbb{E}\left[\hat{R}_{\mathbf{x}_{i,k}}(N)\right]\right|^2}{2\rho_{i,k}^2(N)}\right), & \text{when } i \neq k \end{cases} \\
&= \begin{cases} R_{\mathbf{x}_{i,i}}, & \text{when } i = k \\ \sqrt{\dfrac{\pi R_{\mathbf{x}_{i,i}} R_{\mathbf{x}_{k,k}}}{4N}}L_{1/2}\left(-\dfrac{2N\left|R_{\mathbf{x}_{i,k}}\right|^2}{R_{\mathbf{x}_{i,i}} R_{\mathbf{x}_{k,k}}}\right), & \text{when } i \neq k \end{cases}.
\end{aligned} \tag{3.20}$$

[7] Note that this does not hold if $i = k$ as $x_i(n) = x_k(n)$—this case will be considered separately.



(a) Covariance matrix

(b) Inverse covariance matrix

Figure 3-13: Means along each sub-diagonal of the absolute value of the sample covariance and inverse covariance matrices for a white process for different numbers of snapshots N. These verify (3.20) and (3.23), and indicate that elements that are uncorrelated or conditionally uncorrelated may not appear so. The solid lines are predictions from the equations and the points are obtained from simulation.

In the above, $L_{1/2}(x)$ represents a Laguerre polynomial, given by

$$L_{1/2}(x) = e^{x/2}\left[(1-x)I_0\left(-\frac{x}{2}\right) - xI_1\left(-\frac{x}{2}\right)\right] , \tag{3.21}$$

where $I_n(x)$ is the modified Bessel function of the first kind. It is useful to note that $L_{1/2}(0) = 1$.

If $\mathsf{x}_i(n)$ and $\mathsf{x}_k(n)$ are uncorrelated, $R_{\boldsymbol{\mathsf{x}}_{i,k}} = 0$. In that case (3.20) is indicating that, for finite N, the apparent correlation between them is proportional to $1/\sqrt{N}$. For small N, x_i and x_k may thus appear to be quite strongly correlated.

This is easily verified when $\boldsymbol{\mathsf{x}}(n)$ is white so that $\boldsymbol{R}_{\boldsymbol{\mathsf{x}}} = \boldsymbol{I}_M$. Then, all the elements off the diagonal of the covariance matrix should be zero. In Figure 3-13a, the mean along every subdiagonal of the matrix $\mathbb{E}\left[\left|\hat{\boldsymbol{R}}_{\boldsymbol{\mathsf{x}}}(N)\right|\right]$ is plotted for a white process. The solid lines are the predictions from (3.20), while the points are obtained from simulation.

As the figure indicates, any pair of uncorrelated elements may be apparently correlated in a realization limited dataset. For example, if elements whose correlation is larger than 0.1 are considered "correlated," then, when $N = 50$, any pair of elements $\mathsf{x}_i(n)$ and $\mathsf{x}_k(n)$ can appear fairly strongly correlated.

Expected Absolute Sample Conditional Covariance

For the sample inverse covariance matrix, very similar arguments may be made, with modifications to account for inversion. First, it is again assumed that the elements of the sample inverse covariance matrix $\hat{\boldsymbol{J}}_{\boldsymbol{x}}(N)$ are circularly symmetric complex Gaussian whose real and imaginary parts are independent and have variance $\tilde{\rho}^2_{i,k}(N)$. The moments of the elements of the inverse covariance matrix are [104, Section 5]

$$\mathbb{E}\left[\hat{J}_{\boldsymbol{x}_{i,k}}(N)\right] = \frac{J_{\boldsymbol{x}_{i,k}}}{1-\alpha} \tag{3.22a}$$

$$\tilde{\rho}^2_{i,k}(N) = \frac{1}{2N(1-\alpha)^3} J_{\boldsymbol{x}_{i,i}} J_{\boldsymbol{x}_{k,k}} + o\left(\frac{1}{N^2(1-\alpha)^4}\right) . \tag{3.22b}$$

Because of the complex Gaussianity assumptions on the elements of the sample inverse covariance matrix, their magnitudes are Rician distributed, and the following is obtained

$$\mathbb{E}\left[\left|\hat{J}_{\boldsymbol{x}_{i,k}}(N)\right|\right] = \begin{cases} \dfrac{J_{\boldsymbol{x}_{i,i}}}{1-\alpha}, & \text{when } i = k \\ \sqrt{\dfrac{\pi J_{\boldsymbol{x}_{i,i}} J_{\boldsymbol{x}_{k,k}}}{4N(1-\alpha)^3}} L_{1/2}\left(-\dfrac{2N(1-\alpha)\left|J_{\boldsymbol{x}_{i,k}}\right|^2}{J_{\boldsymbol{x}_{i,i}} J_{\boldsymbol{x}_{k,k}}}\right), & \text{when } i \neq k \end{cases} , \tag{3.23}$$

where $\alpha = M/N$ was defined in (2.11) and it is assumed that $0 < \alpha < 1$.

When elements i, k are independent conditioned upon the other elements of the vector $\boldsymbol{x}$, then $J_{\boldsymbol{x}_{i,k}} = 0$; but $\mathbb{E}\left[\left|\hat{J}_{\boldsymbol{x}_{i,k}}(N)\right|\right]$ is inversely proportional to $1/\sqrt{N(1-\alpha)^3}$, which can be quite large if N is close to M, as indicated in Figure 3-13b for a white process. The figure shows the mean along every subdiagonal of the matrix $\mathbb{E}\left[\left|\hat{\boldsymbol{J}}_{\boldsymbol{x}}(N)\right|\right]$, so it would be expected that, except along the main diagonal when $i - k = 0$, the rest of the values are all relatively small. However, this is not the case in reality. Both the prediction of (3.23) (the solid lines) and simulated results (the points) indicate that every pair of elements may appear strongly correlated conditioned upon the other elements.

In terms of graphical models, given an M-vector with a diagonal covariance matrix, such as that of Section 3.2, it may appear tempting to model it using a graph of M unconnected nodes. However, the results of (3.23) indicate that, if only a finite number N realizations of that process were available, such a model may assume a number of conditional independences that are not really manifest in the dataset. Similar assertions can be made for the block diagonal structure of Section 3.3 also. Thus, before exploiting the graphical model framework in practice, it is necessary to evaluate whether there is sufficient data for the modeled conditional independence statements to become observable. At the least, therefore, this section adds a cautionary note about the possibility of mismatch in these applications.

Finally, note that aside from providing insight into structure, (3.20) and (3.23) can also be used to predict the performance of the sample covariance and inverse covariance matrices as estimators of the covariance and inverse covariance matrices. Specifically, it is simple to show that

$$\mathbb{E}\left[\left|\hat{R}_{\mathbf{x}_{i,k}}(N) - R_{\mathbf{x}_{i,k}}\right|\right] = \sqrt{\frac{\pi R_{\mathbf{x}_{i,i}} R_{\mathbf{x}_{k,k}}}{4N}} \tag{3.24a}$$

$$\mathbb{E}\left[\left|\hat{J}_{\mathbf{x}_{i,k}}(N) - J_{\mathbf{x}_{i,k}}\right|\right] = \sqrt{\frac{\pi J_{\mathbf{x}_{i,i}} J_{\mathbf{x}_{k,k}}}{4N(1-\alpha)^3}}\,. \tag{3.24b}$$

Unsurprisingly, the less data that is available, the worse the performance of the SCM at estimating the covariance matrix; and this limited data effect is even more pronounced for the inverse covariance matrix. While not the objective of this work, this further speaks to the need for improved estimators for the limited data regime for the problems of covariance and inverse covariance matrix estimation—a problem that has received much attention in the literature, e.g. [33], [49], [66], [92], [140], [182].

3.5.2 Frequency-Domain Structure for Cyclostationary Signals with Finite Data

In general, for any known input covariance matrix, and in particular for the diagonal and block diagonal structures of Sections 3.2 and 3.3, the finite-data structure (or lack thereof) can be inferred from the predictions made by (3.20) and (3.23). As shown in Section 3.4, however, graphical model structure for wide-sense cyclostationary processes (and, by specialization, wide-sense stationary processes) is only manifest in the frequency domain.

With finite data, going from the time-domain to the frequency domain presents an additional layer of complexity due to the fact that Fourier transform lengths are now finite. When cascaded with the finite snapshot effect, the resulting effect of finite data on the covariance and inverse covariance structure are interesting. It will be shown that some restrictions need to be placed on the Fourier transform length and frequency spacing so that the structure remains observable.

Let $\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(N)$ be independent realizations of a cyclostationary random process x of length M_t (the subscript t here is to denote the length of the "time-domain" process), whose cyclostationary period is T. The Fourier transform of the realizations are denoted $\boldsymbol{X}(1), \boldsymbol{X}(2), \ldots, \boldsymbol{X}(N)$, each of which are realizations of a frequency-domain random process X, which contains M frequency coefficients of the cyclostationary process at frequencies separated by $2\pi/M_f$. In this case, the frequency coefficients are computed using a finite Fourier transform, and M refers to the number of retained frequencies as defined in Section 3.4.2.

We begin by understanding the ensemble statistics of X, and then subsequently consider the effects of the finite number of realizations.

Covariance and Inverse Covariance Matrices for Cyclostationary Signals with Finite Fourier Transform Length

Define

$$\mathsf{X}_{\omega}^{(M_t)} = \frac{1}{\sqrt{M_t}} \sum_{p=0}^{M_t-1} \mathsf{x}_p e^{-j\omega p}\,, \tag{3.25}$$

where the normalization by $\sqrt{M_t}$ is to ensure the transform is unitary. The cross-correlation function between the frequency coefficients at ω and ν, which is denoted $H^{(M_t)}(\omega, \nu)$, akin to (3.5a), is given by

$$\begin{aligned} H^{(M_t)}(\omega, \nu) &= \mathbb{E}\left[X_{\omega}^{(M_t)} \left(X_{\nu}^{(M_t)}\right)^*\right] && \text{(3.26a)} \\ &= H(\omega, \nu) \star D^{(M_t)}(\omega, \nu)\,, && \text{(3.26b)} \end{aligned}$$

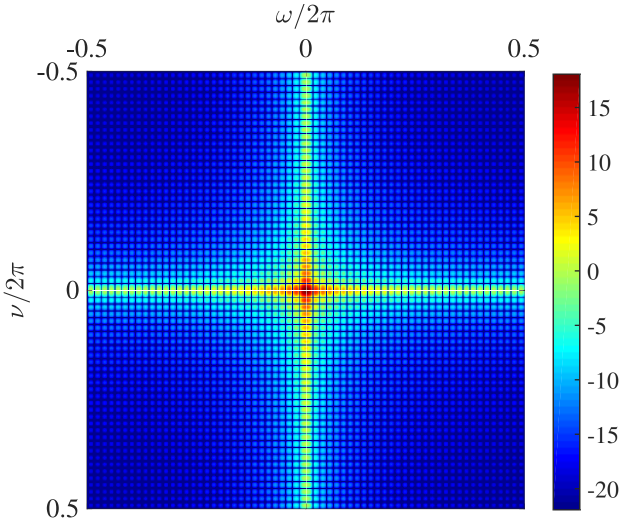


Figure 3-14: 2-d periodic sinc function of (3.28), when $M_t = 64$. The absolute value of the function is plotted on a dB scale.

where $\star$ represents 2-d convolution in the frequency domain,

$$f(\omega,\nu) \star g(\omega,\nu) = \frac{1}{4\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} f(\theta,\phi) g^*(\omega-\theta,\nu-\phi)\,\mathrm{d}\theta\,\mathrm{d}\phi\,, \tag{3.27}$$

and

$$D^{(M_t)}(\omega,\nu) = \frac{\sin(M_t\omega/2)\sin(M_t\nu/2)}{M_t\sin(\omega/2)\sin(\nu/2)} e^{-j\frac{M_t-1}{2}(\omega-\nu)}\,. \tag{3.28}$$

Thus, the cross-correlation function for finite M_t is obtained by a 2-dimensional convolution of the cross-correlation for infinite M_t with a 2-dimensional periodic sinc function (sometimes called a Dirichlet sinc function), which is plotted on a dB scale in Figure 3-14 for $M_t = 64$.

To compute $H^{(M_t)}(\omega,\nu)$, consider convolving $H(\omega,\nu)$, which has support as shown in Figure 3-7, with $D^{(M_t)}(\omega,\nu)$ as shown in Figure 3-14. It is well known that the main lobe of $D^{(M_t)}(\omega,\nu)$ has a diameter of $4\pi/M_t$. With the first-order approximation that the function is zero outside of the main lobe region, the convolution of $H(\omega,\nu)$ and $D^{(M_t)}(\omega,\nu)$ gives rise to a function that has the structure shown in Figure 3-15. There is a non-zero band of width at most the width of the main lobe of the sinc function, i.e., of width at most $4\pi/M_t$, where the bands are spaced by $2\pi/T$. Thus, in order to ensure that the bands are far apart, it is necessary to choose $M_t \gg 2T$.

For the elements of $H^{(M_t)}(\omega,\nu)$ outside the bands of Figure 3-15, it has been observed

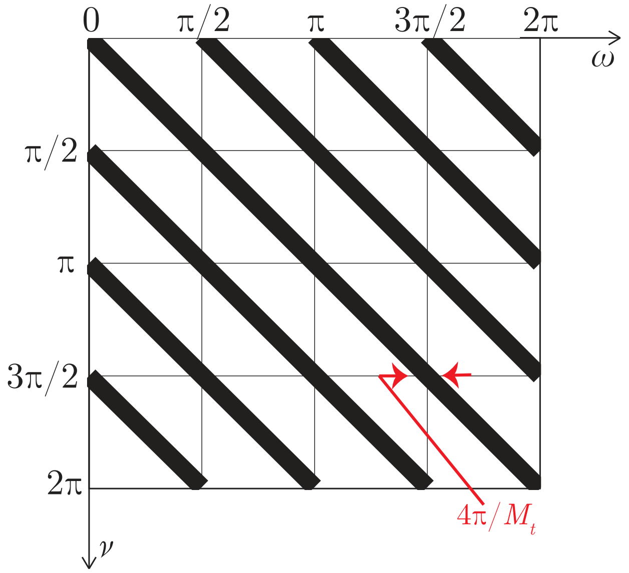


Figure 3-15: Structure of $H^{(M_t)}(\omega,\nu)$, for the case when $T = 4$ and $M_t = 64$. It consists of a band of width at most $4\pi/M_t$ for each non-zero support line of $H(\omega,\nu)$.

empirically that, for large M_t,

$$H^{(M_t)}(\omega,\nu) \approx \frac{\gamma(\omega,\nu)}{M_t}, \text{ if } \omega - \nu \neq 2\pi\frac{t}{T},\ t = -(T-1),\ldots,(T-1)\,, \tag{3.29}$$

where $\gamma(\omega,\nu)$ is a proportionality constant that depends upon ω,ν, but does not depend upon M_t. Attempts to prove mathematically that $\gamma(\omega,\nu)$ does not depend upon M_t have proven unsuccessful.[8] However, it is possible to verify (3.29). For instance, Figure 3-16 shows $\left|H^{(M_t)}(-\pi,-\pi/2)\right|^2$ for a wide-sense stationary process (for which $T = 1$), which falls off as $1/M_t^2$ for $M_t \geq 32$.

The covariance matrix of $\boldsymbol{X}$ is obtained by sampling $H^{(M_t)}(\omega,\nu)$ at M^2 points, where the points are separated in frequency by $2\pi/M_f$. As in Section 3.4.1, assume that M_f is a multiple of T to avoid issues with fractional sampling. Moreover, in order that only one frequency bin overlaps with each band of Figure 3-15, it is needed that the frequency spacing between bins is at least half the main lobe width, i.e., $2\pi/M_f \geq 2\pi/M_t$, or $M_f \leq M_t$. However, in order to avoid undersampling in the frequency domain, $M_f \not< M_t$, and so M_f is restricted to be equal to M_t. Thus, it is required that $M_f = M_t \gg 2T$ and M_f is a multiple

[8] It has been suggested that the variation of (3.29) is a manifestation of the Riemann-Lebesgue lemma.

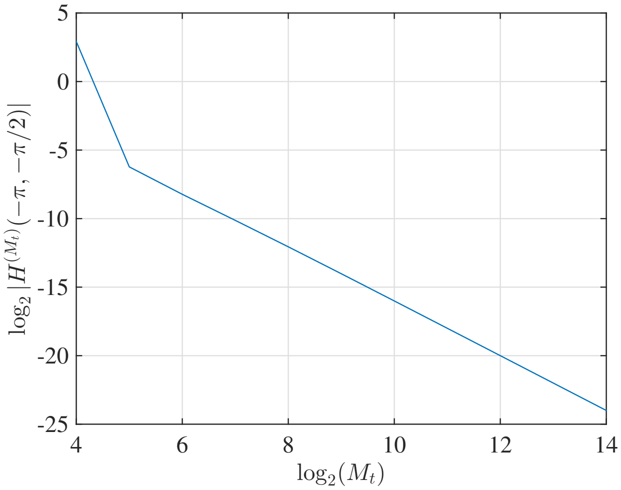


Figure 3-16: Example of the $1/M_t$ variation of $H^{(M_t)}(\omega, \nu)$ off the cyclostationary diagonals. $\left|H^{(M_t)}(-\pi, -\pi/2)\right|$ is plotted as a function of M_t on a log-log scale for a wide-sense stationary process. The straight line corresponds to a $1/M_t$ falloff for $\left|H^{(M_t)}(\omega, \nu)\right|$.

of T, which places restrictions on the time-domain block length and frequency spacing.

Assuming that these conditions are met, then the covariance matrix of $\boldsymbol{X}$ is given by $R^{(M_t)}_{\boldsymbol{X}_{i,k}}$, where

$$R^{(M_t)}_{\boldsymbol{X}_{i,k}} \approx \begin{cases} H\left(\omega_i, \omega_k\right), & \text{when } i - k = p\dfrac{M_f}{T},\ p = -(T-1), \ldots, (T-1) \\ \dfrac{\gamma(\omega_i, \omega_k)}{M_t}, & \text{otherwise} \end{cases}, \tag{3.30}$$

where the case for $i - k = pM_f/T$ follows from the fact that for large M_t the periodic sinc function has the sifting property [63]

$$\lim_{M_t \to \infty} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} D^{(M_t)}(\theta, \phi) F^*(\omega - \theta, \nu - \phi)\, \mathrm{d}\theta\, \mathrm{d}\phi = F(0,0)\,. \tag{3.31}$$

To obtain a result similar to that of (3.30) for inverse covariance matrices, note that $\boldsymbol{R}^{(M_t)}_{\boldsymbol{X}} = \boldsymbol{R}^{(\infty)}_{\boldsymbol{X}} + \boldsymbol{B}$, where $\boldsymbol{R}^{(\infty)}_{\boldsymbol{X}}$ has the structure shown in Section 3.4.1, and $\boldsymbol{B} = \boldsymbol{R}^{(M_t)}_{\boldsymbol{X}} - \boldsymbol{R}^{(\infty)}_{\boldsymbol{X}}$ contains the $\gamma(\omega_i, \omega_k)/M_t$ terms of the covariance matrix.

Let $\boldsymbol{J}^{(\infty)}_{\boldsymbol{X}} = \left(\boldsymbol{R}^{(\infty)}_{\boldsymbol{X}}\right)^{-1}$, which has the structure described in Section 3.4.3. Then, assuming that the eigenvalues of the matrix $\boldsymbol{J}^{(\infty)}_{\boldsymbol{X}} \boldsymbol{B}$ are all less than 1, the matrix $\boldsymbol{J}^{(M_t)}_{\boldsymbol{X}} =$

$\left(\boldsymbol{R}_{\boldsymbol{X}}^{(M_t)}\right)^{-1}$ can be expanded in a convergent power series, so that

$$\boldsymbol{J}_{\boldsymbol{X}}^{(M_t)} = \boldsymbol{J}_{\boldsymbol{X}}^{(\infty)} + \boldsymbol{J}_{\boldsymbol{X}}^{(\infty)} \boldsymbol{B} \boldsymbol{J}_{\boldsymbol{X}}^{(\infty)} + \dots . \tag{3.32}$$

Combining this with (3.30), it can be seen that

$$J_{\boldsymbol{X}_{i,k}}^{(M_t)} = \begin{cases} J_{\boldsymbol{X}}{}_{i,k}^{(\infty)} + o\left(\dfrac{1}{M_t}\right), & \text{when } i-k = p\dfrac{M_f}{T},\ p = -(T-1), \dots, (T-1) \\ \dfrac{\tilde{\gamma}(\omega_i, \omega_k)}{M_t} + o\left(\dfrac{1}{M_t^2}\right), & \text{otherwise} \end{cases}, \tag{3.33}$$

where $\tilde{\gamma}(\omega_i, \omega_k)$ is independent (or, more accurately, weakly dependent) of M_t, just like $\gamma(\omega_i, \omega_k)$ in (3.30).

Equation (3.33) shows that, while the main structure predicted by Section 3.4.3 is maintained in the inverse covariance matrix, the elements $J_{\boldsymbol{X}_{i,k}}^{(M_t)}$, $i-k \neq pM_f/T$ are not exactly zero when the Fourier transform length is finite, and more precisely, they fall off at a rate proportional to the inverse of the Fourier transform length.

Effect of Finite Numbers of Realizations

Combining the inverse covariance matrix of the frequency domain vector $\boldsymbol{X}$ computed in (3.33) with the finite realization result of (3.23), the following result is obtained for the entries of the frequency-domain inverse covariance matrix computed with finite realizations and a finite Fourier transform length:

$$\mathbb{E}\left[\left|\hat{J}_{\boldsymbol{X}_{i,k}}(N)\right|\right] = \sqrt{\frac{\pi J_{\boldsymbol{X}_{i,i}} J_{\boldsymbol{X}_{k,k}}}{4N(1-\alpha)^3}} L_{1/2}\left(-\frac{2N(1-\alpha)\left|\tilde{\gamma}(\omega_i, \omega_k)\right|^2}{M_t^2 J_{\boldsymbol{X}_{i,i}} J_{\boldsymbol{X}_{k,k}}}\right). \tag{3.34}$$

In the above, $\alpha = M/N$ where M is the number of retained frequencies and N the number of realizations; and $\tilde{\gamma}(\omega_i, \omega_k)$ is the proportionality constant appearing in (3.33) and $\boldsymbol{J}_{\boldsymbol{X}} = \boldsymbol{J}_{\boldsymbol{X}}^{(\infty)}$ is the inverse covariance matrix in the frequency domain for the cyclostationary process assuming an infinite Fourier transform length (i.e., it is the matrix defined in Section 3.4.3). The superscripts have been dropped to reduce the complexity of the notation.

It is simplest to gain insight by considering the special case when the time-domain process

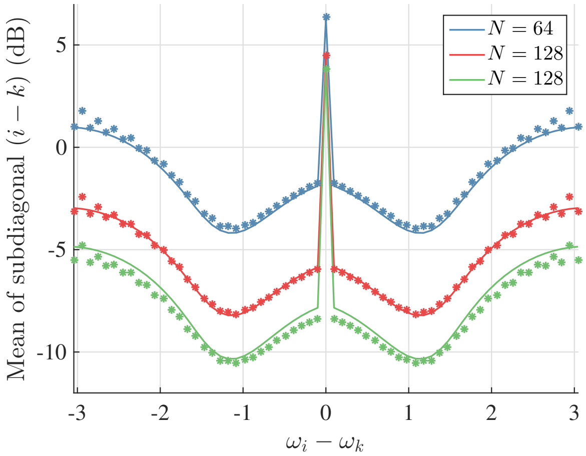
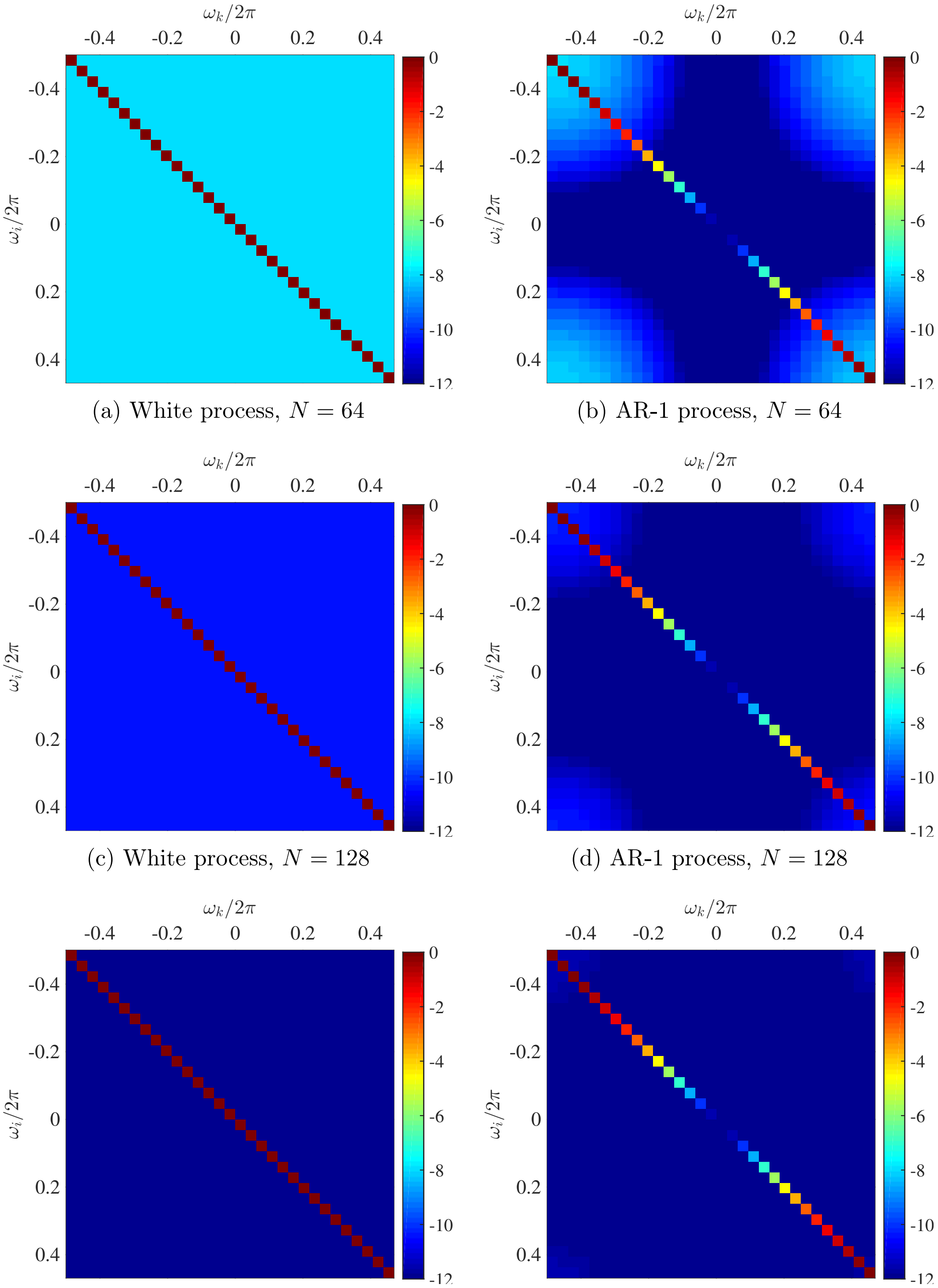


Figure 3-17: Mean along each subdiagonal of the absolute value of the sample inverse covariance matrix in the frequency domain for an AR-1 process on a dB-scale. The x-axis represents $\omega_i - \omega_k$ along each diagonal. The solid lines are the predictions from (3.34) and the dot markers are from simulation.

is wide-sense stationary (rather than wide-sense cyclostationary). In this case the ensemble covariance matrix with an infinite Fourier transform length is diagonal and contains the power spectral density along its diagonal. Thus $J_{\boldsymbol{X}_{i,i}} = 1/S_{\mathsf{x}}(\omega_i)$.

Using this, some predictions and inferences can be made for the special case of wide-sense stationary processes. To begin with, Figure 3-17 plots the mean along each subdiagonal of the matrix $\mathbb{E}\left[\left|\hat{\boldsymbol{J}}_{\boldsymbol{X}}(N)\right|\right]$ for the particular case when the process is wide-sense stationary (more precisely, it is an AR-1 process) to verify the accuracy of the predictions made.. Solid lines are the values predicted by (3.34) and points are obtained by simulation. This plot is similar to the plot of Figure 3-13, except that it is for an AR-1 process in the frequency domain and the subdiagonals are now indexed by a frequency difference. The analysis clearly does not capture every aspect of the absolute inverse covariance matrix but the predictions are reasonably accurate.

In order to see how the predictions of (3.34) can be used to modify the graphical model of the frequency-domain cyclostationary processes to handle the finite data results described in this section, consider Figure 3-18, which plots the predicted absolute sample inverse covariance matrix for 2 different wide-sense stationary processes—a white process and an AR-1 process—for different values of N. As a reminder, if the absolute inverse covariance between two elements is high, the elements may appear conditionally dependent in any given



(a) White process, $N = 64$

(b) AR-1 process, $N = 64$

(c) White process, $N = 128$

(d) AR-1 process, $N = 128$

(e) White process, $N = 256$

(f) AR-1 process, $N = 256$

Figure 3-18: Expected absolute values of the sample inverse covariance matrix for a white process and an AR-1 process in the frequency domain for $M_t = M_f = 32$ (as required), no frequencies dropped ($M = 32$) and different values of N. The relative magnitudes of the elements are shown on a dB-scale.

realization of N snapshots, and so an edge is introduced between those two elements.

For large N, as expected, the model simply corresponds to the model predicted by ensemble statistics, which, in this case, is M independent nodes as the processes in question are wide-sense stationary. The general intuitive trend is that the smaller the value of N, the more the number of edges that are needed in the graph. The specific pattern of edges depends on the power spectral density of the process. For a white process, for instance, if a 12dB relative absolute condition cross-correlation limit were assumed, for $N = 64$, the graphical model would have to be a fully connected graph. On the other hand, with an AR-1 process, every frequency is connected to a few adjacent frequencies (modulo 2π), where the number depends on the AR-1 parameter (i.e., the correlation between adjacent time-series samples) and the number of snapshots N.

Thus, in general it can be said the graphical model may need to be modified in order to account for the finite data size by introducing additional edges into the assumed graph. The question of which specific frequencies need to be connected depends on the power spectral density of the process for wide-sense stationary processes, and more generally, on the Loéve bispectrum for cyclostationary processes. The potential gains from such an approach are demonstrated in Section 5.5.2 for adaptive equalization in underwater communication systems.

3.6 Structure of Received Signal in Underwater Acoustic Communication

As discussed in Section 3.4.6, multichannel received data in underwater acoustic communication systems is expected to be characterized by inverse covariance structure in the frequency domain. Data from the Surface Processes And Communication Experiment 2008 (SPACE08) is now used to show the expected cyclostationary structure. This section pulls together a variety of results from Sections 3.4 and 3.5.

SPACE08 was conducted off the coast of Martha's Vineyard, MA, from October 14 to November 1, 2008. For the purposes of this work, m-sequence data from this experiment is sufficient. The experiment comprised 2-hour intervals called "epochs," and a 2-hour long collection of signals was transmitted during each epoch. Different epochs can be characterized by quite widely different channel conditions due to changing environmental conditions.

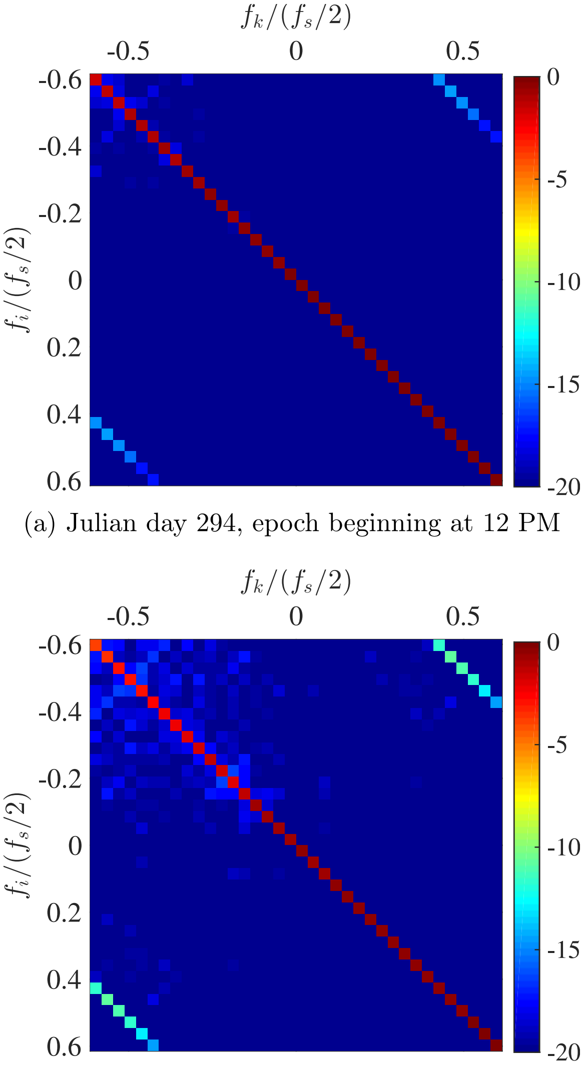
Consider first the data from a single received hydrophone (i.e., single channel data). The received signal was synchronized, converted to baseband and then processed so that the effective observed baseband signal has $T = 2$ samples per symbol. The received signal was collected into blocks of duration 4.6 ms, which corresponds to a time-domain block length of $M_t = 60$ with an effective baseband sampling rate of about 13 kHz. These blocks correspond to the blocks $\boldsymbol{x}_r(n)$ of Figure 3-11.

The metrics of performance are the $\boldsymbol{x}_r(n)$s of Figure 3-11. To comply with the requirements of Section 3.5.2, $M_f = 60$, but only $M = 37$ frequencies (corresponding to frequencies between -4kHz to 4kHz) were retained, as the effective baseband signal was restricted to this band. The data was windowed with a rectangular window of length $N = 400$. These parameters are fairly typical for medium to long range wireless underwater acoustic communication.

Figure 3-19 shows the relative magnitudes of the elements of the inverse covariance matrices computed for data collected at two different epochs. The cyclostationary frequency domain structure of Section 3.4 is clearly present. The "spillover" to adjacent frequency components (i.e., the deviation from the ideal structure of Figure 3-10) is a manifestation of practical data limitation constraints as discussed in Section 3.5.2, and this results in the presence of "bands" of non-zero elements in the inverse covariance matrix. Figure 3-19 shows that there is variation in the width of the bands among epochs. In particular, they are broader for the epoch on Julian Day 300.

The graphical model that represents the inverse covariance matrix structure of Figure 3-19 is obtained by augmenting the structure of Figure 3-10 with edges between all the nodes of d adjacent cliques, to reflect the inverse covariance structure that has the "conditional correlation spillover" effect. The precise value of d required may depend on the environmental conditions, but $d = 2$ or 3 has been found sufficient for good performance in adaptive equalization for underwater communication (see Section 5.5.2).

For $d = 2$, i.e., where each clique of Figure 3-10 is connected to its adjacent cliques, the



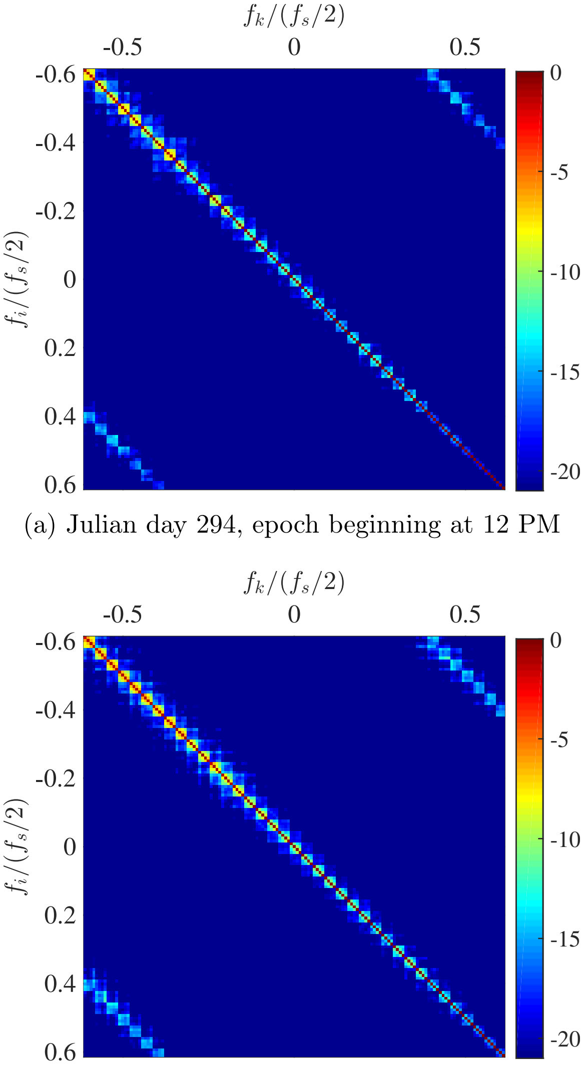
(a) Julian day 294, epoch beginning at 12 PM

(b) Julian day 300, epoch beginning at 8 AM

Figure 3-19: Relative magnitudes of elements (dB-scale) of the frequency-domain inverse covariance matrix of the received signal in the SPACE08 experiment at two different epochs. The results show the presence of the cyclostationary structure, as well as the effect of environmental conditions.

graphical model is shown in Figure 3-21a. For the SPACE08 data, $\Delta = M_f/T = 30$ and $m(t) = \lfloor (M-t)/\Delta \rfloor$, where $M = 37$ is the number of frequencies retained for processing.

Figure 3-20 shows the structure of Figure 3-19 extended to multichannel inputs, where R channels (in Figure 3-20, $R = 4$) of data are available. The structure is intuitively clear—where in the single channel cross-spectral inverse correlation matrix there was a single entry for each frequency pair, there is now a $R \times R$ submatrix representing the spatial substructure of the received signal field.



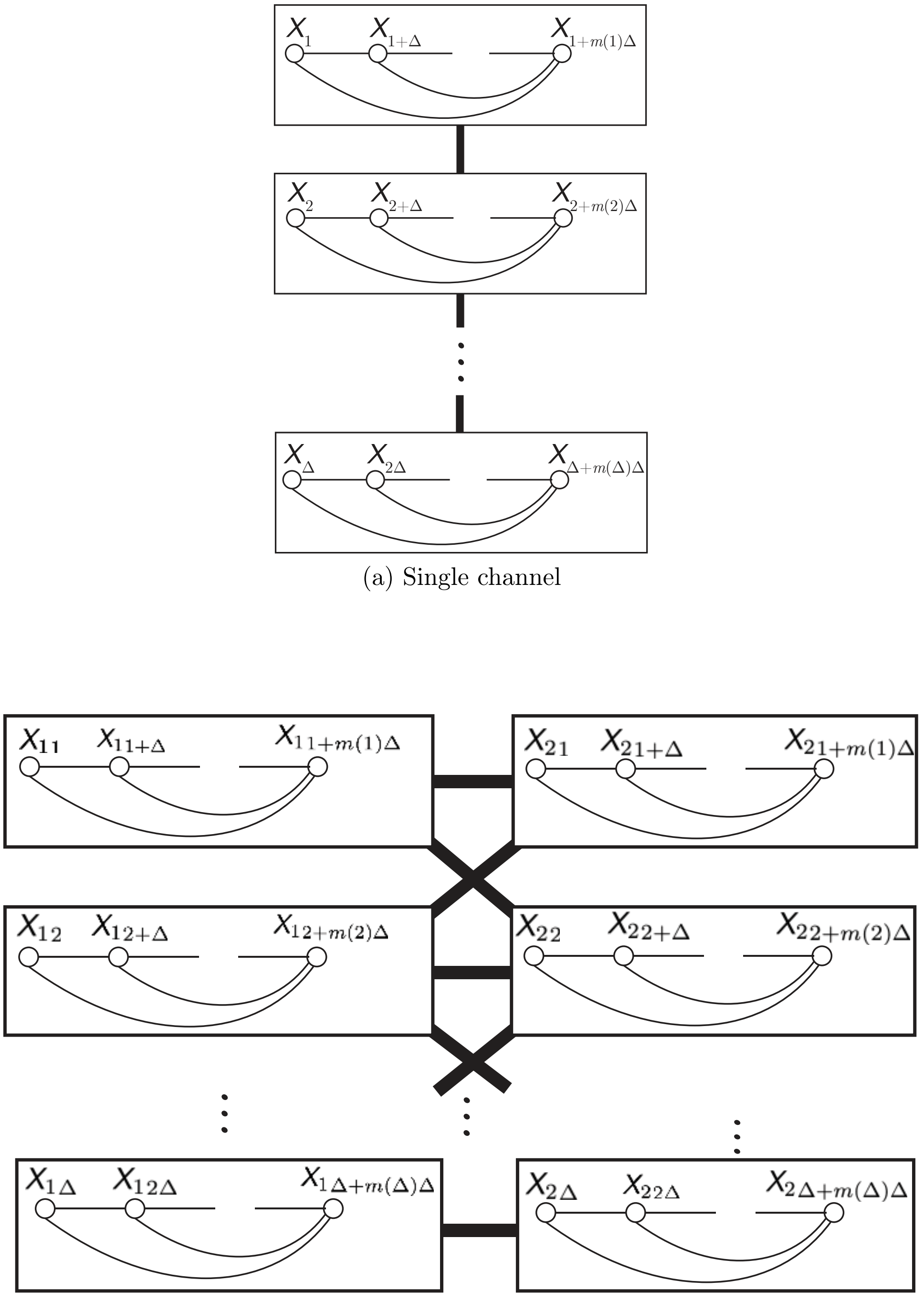
(a) Julian day 294, epoch beginning at 12 PM

(b) Julian day 300, epoch beginning at 8 AM

Figure 3-20: Multichannel inverse covariance structure from SPACE08 experiment. The relative values of the absolute inverse covariance matrix for multichannel received data with $R = 4$ on a dB-scale are shown for the same epochs as before. The figure is similar to Figure 3-19, except that each element of Figure 3-19 is replaced by a $R \times R$ block.

Finally, the graphical model for the multichannel data is shown in Figure 3-21b. This graph is a simple extension of the single channel case, for the case where $R = 2$. As in the single channel case, d represents the number of adjacent cliques that are connected to account for data limitations, and as in Figure 3-21a, $d = 2$, meaning that each clique is connected to the neighboring clique to account for the effect of finite sample sizes and finite Fourier transform length in this system.

The presence of physical acoustics in the structure above may not be immediately appar-



(a) Single channel

(b) Multichannel

Figure 3-21: Graphical model for the received signal in underwater acoustic communication, including the cyclostationary structure, and modification by connecting adjacent cliques for single channel and multichannel data. $\Delta = M_f/T = 30$ for the SPACE08 data and $m(t) = \lfloor (M-t)/\Delta \rfloor$, where $M = 37$ frequencies are retained for processing. The graph is shown for $d = 2$, where each "multichannel cyclostationary" clique is connected to the adjacent one. Figure 3-21a and 3-21b represent the structure shown in Figure 3-19 and 3-20, respectively.

ent, but it should be emphasized that it is a consequence of the physical properties of the underwater communication channel that, on the time scale of a typical block of communciation data, the channel variations are wide-sense stationary [172].

This is a crucial point in regard to modelling the channel. In general, predicting the channel variations depends to a fine scale on the important environmental processes for the particular scenario, such as range, depth, frequency, wave conditions, and so on, and would require modeling that is beyond the capabilities of a real-time adaptation or measurement system. However, by exploiting the statistical property that channel variations can be reasonably modeled as wide-sense stationary over the time-scale of interest, highly predictable statistical structure is available in the frequency domain, that is representable using a graphical model—this is a property that we can reliably take advantage of in adaptation.

3.7 Some Comments and Looking Ahead

In this chapter three applications—acoustic echo cancellation, fractionally-spaced channel identification and acoustic echo cancellation—have been introduced along with the graphical model structure expected to be observed in these applications. In all three applications, the graphical model is a consequence of the statistical properties of the time-series that serves as the input to the system. Whereas for the acoustic echo cancellation and channel identification applications, the graph is manifest in the time-domain; for the adaptive signal processing application, a frequency-domain graph is obtained. The correlation matrix structure shown, for example, in Figure 3-19 and 3-20, is used to create structured graphical models (Figure 3-21) using the approach demonstrated in Figure 3-1. In Chapters 4 and 5 the resulting structured graphical models are used to develop the two classes of estimation algorithms that form the core of this thesis.

It should be noted that the applications considered all have time-series as their inputs. Of course, if the input is not a time-series, then other statistical properties would need to be considered. For instance, if the input to the system identification processing problem comes from an array of sensors, then the spatial statistical structure would need to be considered. In this work, the structure is all based on time-series statistics—although multichannel data is considered in Section 3.6, no spatial structure has been assumed.

At this juncture, we take a step away from the application domain and begin with the development of algorithms to perform system identification with input structure characterized by graphical models. The algorithms are henceforth developed for general graphical models, without regard to any of the particular graphs developed in this chapter. Once the analysis for a general graphical model is complete, the algorithms will be used in the applications described herein to test performance in practice; at that time, the graph describing the data in the respective application will be used by the algorithm.

In the next chapter, the basic framework for exploiting input graphical model structure will be introduced and the GEM-LS algorithm will be developed.

Chapter 4

Graphical Expectation-Maximization Least Squares

To briefly review the main ideas thus far, the objective of the thesis is to estimate an unknown M-dimensional linear system $\boldsymbol{h}$ from observations of the input $\boldsymbol{x}(n)$ and output $y(n)$, where

$$y(n) = \boldsymbol{h}^{\dagger}\boldsymbol{x}(n) + v(n),\ n = 1, 2, \ldots, N\,,$$

as stated in (2.1).

It has been explained that $\boldsymbol{x}(n), n = 1, 2, \ldots, N$ are independent realizations of $\mathbf{x}$, where $p_{\mathbf{x}}(\boldsymbol{x})$ is characterized by a graphical model. A variety of applications of such a system model with graphical model input structure were introduced in Chapter 3.

We now move to the main part of this work—to understand how to solve linear parameter identification problems with graphical model structure on the input. The approach here utilizes the framework of the expectation-maximization (EM) algorithm and, because there is graphical model structure, the algorithm is called the graphical expectation maximization least squares (GEM-LS) algorithm.

4.1 Modeling the Problem

4.1.1 Overall Graph for the Problem

While the graphical model structure characterizes x, it is clear that the distribution of x is unaffected by the value of $\boldsymbol{h}$.[1] However, $\boldsymbol{h}$ evidently affects the joint distribution of (x, y). In other words, $\boldsymbol{h}$ is a parameter of the distribution $p_{\mathsf{x},\mathsf{y}}(\boldsymbol{x}, y)$. The graph of this distribution is readily obtained by observing that

$$p_{\mathsf{x},\mathsf{y}}(\boldsymbol{x}, y) = p_{\mathsf{x}}(\boldsymbol{x}) p_{\mathsf{y}|\mathsf{x}}(y \mid \boldsymbol{x}) \, ,$$

which leads to the following model:

1. x is characterized by a decomposable undirected Gaussian model as introduced in Section 2.4.1, and

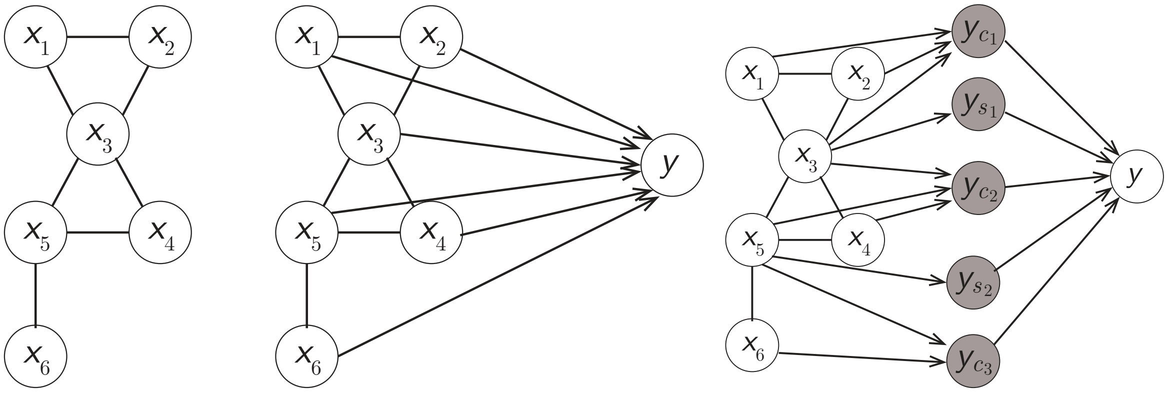
2. Each $\mathsf{x}_i, i = 1, \ldots, M$, has a *directed* link to y (Section 2.4.2)

Figure 4-1a shows an example of a decomposable undirected graph for x. The corresponding overall graph for (x, y) is shown in Figure 4-1b. Thus, the overall problem of estimating $\boldsymbol{h}$ is a parameter estimation problem in a PDAG. The most convenient way to represent this problem is to convert the entire graph into either a directed acyclic graph (DAG) or an undirected graph.

It is not possible to represent the distribution of x by a DAG without making further assumptions. The undirected equivalent of the PDAG of Figure 4-1b is obtained by moralization. The moral graph of this kind of PDAG is simply obtained by connecting all the x_i's, i.e., it is simply a fully connected graph. And indeed, it is a simple matter to verify that ML parameter estimation of $\boldsymbol{h}_0$ in the graph of Figure 4-1b is simply least squares estimation.

As the LS solution throws away all the structural information on x, directly attempting to incorporate the graphical model structure of x into the problem does not at first appear to yield any useful insight.

[1] As discussed in Section 2.1.2, this is only true for forward problems and not inverse system identification problems. Nonetheless, it is assumed to hold for both kinds of problems as otherwise, the problem models for inverse system identification problems become overly complex.



(a) Graph of $\boldsymbol{x}$ (b) Graph of $(\boldsymbol{x}, y)$ (c) Augmented Graph

Figure 4-1: Example of a decomposable graph for $\boldsymbol{x}$, the corresponding overall graph for $(\boldsymbol{x}, y)$ that characterizes the problem, and the augmented graph that leads to the EM framework. The shaded variables in the augmented graph are latent variables.

4.1.2 Augmented Graph

To incorporate the structure in a meaningful manner, the graph needs to be augmented. The idea is to introduce latent nodes that represent parts of the output corresponding to each clique and separator. Specifically, as demonstrated in Figure 4-1c, for every clique $c \in \mathcal{C}$ and separator $s \in \mathcal{S}$, where $\mathcal{C}, \mathcal{S}$ are the cliques and separators of the graph of $\boldsymbol{x}$, respectively, a clique (or separator) output y_c (y_s) is introduced, with the following models attached:

$$y_c(n) = \boldsymbol{h}_c^\dagger \boldsymbol{x}_c(n) + v_c(n),\ c \in \mathcal{C} \tag{4.1a}$$

$$y_s(n) = \boldsymbol{h}_s^\dagger \boldsymbol{x}_s(n) + v_s(n),\ s \in \mathcal{S} \tag{4.1b}$$

$$y(n) = \sum_{c\in\mathcal{C}} y_c(n) - \sum_{s\in\mathcal{S}} y_s(n) + v_0(n)\,, \tag{4.1c}$$

where v_c, v_s, v_0 are modeled as mutually independent zero-mean Gaussian noise, each having variance σ_c^2, where

$$\sigma_c^2 = \frac{\sigma^2}{C+S+1} \tag{4.2a}$$

$$= \frac{\sigma^2}{D+1}\,. \tag{4.2b}$$

$$\boldsymbol{h} = \underbrace{\begin{bmatrix} \\ \boldsymbol{h}_{c_1} \\ \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\left[\boldsymbol{h}_{c_1}\right]^{M\times 1}_{c_1\times 1}} - \begin{bmatrix} 0 \\ 0 \\ h_{s_1} \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \\ \boldsymbol{h}_{c_2} \\ \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ h_{s_2} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \\ \boldsymbol{h}_{c_3} \end{bmatrix}$$

Figure 4-2: Operation of (4.3) for the structure of Figure 4-1a. The notation of the fill-in operator for the first clique is also shown.

Recall that σ^2 is the total variance of the noise in (2.1) and C, S, respectively represent the number of maximal cliques and non-empty separators in the graph, so that this model distributes the noise across the cliques and the separators with equal power across each clique and separator. $D = C + S$ is defined for future convenience. In Section 4.3, some constraints will be placed on D to ensure convergence of the GEM-LS algorithm.

While (4.2a) is not the only way to assign the noise power to the different cliques and separators, the assumption that all the cliques and separators have equal noise power considerably simplifies the analysis.[2] Observe that the system model defined by (4.1) and (4.2a) is completely consistent with that of (2.1) with

$$\boldsymbol{h} = \sum_{c \in \mathcal{C}} \left[\boldsymbol{h}_c\right]^{M\times 1}_{c\times 1} - \sum_{s \in \mathcal{S}} \left[\boldsymbol{h}_s\right]^{M\times 1}_{s\times 1} , \tag{4.3}$$

where the fill-in operator $[\cdot]^{M\times 1}_{c\times 1}$ takes the argument defined in the brackets and places it into a matrix of dimension specified by the superscript (in this case, $M \times 1$) in the locations specified by the subscript (in this case, the elements of set c). This is operationally identical (although the notation is slightly different) to the zero fill-in operator defined in [181]. The operation of (4.3) for the cliques and separators of the graph of Figure 4-1a is shown in Figure 4-2.

Note that $y_c(n)$ and $y_s(n)$ are unknown. In other words, the random variables y_c and y_s

[2] There may be situations in which some a-priori knowledge of how noisy each clique and separator output is available, in which case that knowledge can easily be incorporated into the model of (4.1).

are *latent*, or unobserved in the graph. Latent variables are shown as shaded in the graph of Figure 4-1c.

As a final comment, the requirement of decomposability is implicitly used in the augmented graph. The clique-separator structure is a consequence of decomposability, and without this property, it is not clear how the "clique outputs" could be consistently defined.

4.1.3 Inference and Estimation in the Augmented Graph

It may now appear as if the problem has been further complicated. A number of nodes have been added and the new graph is also not moral. However, what has actually been done is to *decouple* the estimation problem from the amoral portion of the graph, thereby eliminating the need to perform ML estimation on the entire graph. In other words, the estimation problem has been split into several sub-problems, each of which operates on a completely moralized subgraph, and which are easy to recombine.

To understand how, imagine that the values of $y_c(n), y_s(n)$ corresponding to each $\boldsymbol{x}(n), y(n)$ were known, i.e., that N observations of the complete graph were available. Then, it is simple to note that $\boldsymbol{h}_c, \boldsymbol{h}_s$ could easily be estimated by solving a LS problem for each clique and separator. Each of these LS problems would have a dimension the size of the corresponding clique. Thus, the amount of data required depends on the size of the largest clique, and a solution exists provided $N \geq \max_{c \in \mathcal{C}} |c|$ (in contrast with conventional LS, which requires that $N \geq M$ to guarantee the existence of a solution).

On the other hand, given values of $\boldsymbol{x}_c(n), \boldsymbol{x}_s(n)$, which can be computed from $\boldsymbol{x}(n)$; $\boldsymbol{h}_c, \boldsymbol{h}_s$ and $y(n)$, it is then possible to infer the values of $y_c(n)$ and $y_s(n)$. As will be shown in Section 4.2, a simple closed form solution exists for ML inference in this system (under Gaussianity assumptions).

This speaks to an iterative method. Starting with some values of $\boldsymbol{h}_c, \boldsymbol{h}_s$, the proposed algorithm iterates between inferring $y_c(n), y_s(n)$ and learning a new estimate of $\boldsymbol{h}_c$ and $\boldsymbol{h}_s$. It should be evident that this is precisely parameter learning in a graph with latent variables, so the iterations correspond exactly to iterations of an expectation-maximization algorithm [50]. In the *expectation step* (or E-step) the latent variables are estimated, whereas in the *maximization step* (M-step), the parameter is estimated. This is the basic idea behind the

GEM-LS algorithm.

Note that the proposed GEM-LS algorithm will estimate $\boldsymbol{h}_c, \boldsymbol{h}_s$, rather than estimating $\boldsymbol{h}$. The estimates of $\boldsymbol{h}_c, \boldsymbol{h}_s$ are combined using (4.3) to obtain the estimate of $\boldsymbol{h}$.

4.2 Derivation of GEM-LS

The GEM-LS algorithm proceeds iteratively between estimating $\boldsymbol{h}_c, \boldsymbol{h}_s$ and estimating $y_c(n), y_s(n)$. Henceforth, let k represent the iteration index. $\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_c}$ and $\hat{y}^k_c(n), \hat{y}^k_s(n)$ represent the estimates of $\boldsymbol{h}$ and $y_c(n), y_s(n)$, respectively, in the kth iteration; and let $\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}}$ be the overall GEM-LS estimate in the kth iteration.

Prior to the first iteration, the initial estimate $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}$ is set to some fixed vector and $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}_c}, \hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}_s}$ are initialized from this by indexing. Note that the decomposition of $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}$ into $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}_c}$ and $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}_s}$ is not unique. However, the analysis of Section 4.3 and 4.4 indicate that while the choice of $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}$ has an effect on the performance of the algorithm, the choice of decomposition does not, since the overall updates made by the GEM-LS algorithm depend only upon $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}$ and not the individual clique/separator initializations.

The E-Step: Estimating the Latent Variables

The "expectation" step attempts to estimate the latent variables $\boldsymbol{y}_a(n)$ for each n, using the previously computed estimate of $\boldsymbol{h}_c, \boldsymbol{h}_s$.

To simplify the mathematics of this step, it is convenient to define

- The $D \times 1$ vector

$$\boldsymbol{s} = \begin{bmatrix} \mathbf{1}_{C\times 1} \\ -\mathbf{1}_{S\times 1} \end{bmatrix}, \tag{4.4a}$$

- The $D \times 1$ vector $\boldsymbol{y}_a(n), n = 1, \ldots, N$, where

$$y_{a_i}(n) = \begin{cases} y_{c_i}(n)\,, & i = 1, 2, \ldots C \\ y_{s_{i-C}}(n)\,, & i = C+1, \ldots D \end{cases}, \tag{4.4b}$$

$$\boldsymbol{s} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

(a) $\boldsymbol{s}$

$$\boldsymbol{y}_a(n) = \begin{bmatrix} y_{c_1}(n) \\ y_{c_2}(n) \\ y_{c_3}(n) \\ y_{s_1}(n) \\ y_{s_2}(n) \end{bmatrix}$$

(b) $\boldsymbol{y}_a(n)$

$$\boldsymbol{H} = \begin{bmatrix} & 0 & 0 & 0 & 0 \\ \boldsymbol{h}_{c_1} & 0 & 0 & 0 & 0 \\ & & 0 & h_{s_1} & 0 \\ 0 & \boldsymbol{h}_{c_2} & 0 & 0 & 0 \\ 0 & & \boldsymbol{h}_{c_3} & 0 & h_{s_2} \\ 0 & 0 & & 0 & 0 \end{bmatrix}$$

(c) $\boldsymbol{H}$

Figure 4-3: Definitions of (4.4) for the graph of Figure 4-1. For this example, $C = 3$ (the number of cliques) and $S = 2$ (separators), so that $C + S = 5$. There are $M = 6$ nodes in the graph.

- The $M \times D$ matrix $\boldsymbol{H}$, whose ith column is given by

$$\boldsymbol{H}_{:,i} = \begin{cases} [\boldsymbol{h}_{c_i}]_{c_i \times 1}^{M \times 1}\,, & i = 1, 2, \ldots C \\ \left[\boldsymbol{h}_{s_{i-C}}\right]_{s_{i-C} \times 1}^{M \times 1}\,, & i = C+1, \ldots D \end{cases}\,, \tag{4.4c}$$

  where the fill-in operator was defined in (4.3).

To illustrate these quantities, Figure 4-3 shows them for the example graph of Figure 4-1. Note that these quantities are used to simplify the notation while developing the GEM-LS algorithm, but will not feature in the final solution.

With these quantities defined, the augmented system model introduced in (4.1) can be concisely written as

$$\boldsymbol{y}_a(n) = \boldsymbol{H}^\dagger \boldsymbol{x}(n) + \boldsymbol{v}_a(n) \tag{4.5a}$$

$$y(n) = \boldsymbol{s}^\dagger \boldsymbol{y}_a(n) + v_0(n)\,, \tag{4.5b}$$

and the reconstruction operation of (4.3) is equivalent to

$$\boldsymbol{h} = \boldsymbol{H}\boldsymbol{s}\,. \tag{4.5c}$$

Consider the joint distribution of the complete set of random variables (latent and observed) $\{\boldsymbol{x}, \boldsymbol{y}_a, y\}$ of the augmented PDAG of Figure 4-1c. Since $\boldsymbol{x}$ and all the noise is

Gaussian by assumption, and $\mathbf{y}_a$ and y are obtained by linear combinations of complex Gaussian vectors, the entire set is jointly Gaussian. Additionally,

$$\log p_{\mathbf{x},\mathbf{y}_a,\mathsf{y}}(\boldsymbol{x},\boldsymbol{y}_a,y) = \log p_{\mathbf{x}}(\boldsymbol{x}) + \log p_{\mathbf{y}_a|\mathbf{x}}(\boldsymbol{y}_a \mid \boldsymbol{x}) + \log p_{\mathsf{y}|\mathbf{y}_a}(y \mid \boldsymbol{y}_a) \tag{4.6a}$$

$$= Z - \boldsymbol{x}^\dagger \boldsymbol{J}_{\mathbf{x}} \boldsymbol{x} - \frac{(\boldsymbol{y}_a - \boldsymbol{H}^\dagger \boldsymbol{x})^\dagger (\boldsymbol{y}_a - \boldsymbol{H}^\dagger \boldsymbol{x})}{\sigma_c^2} - \frac{\left|y - \boldsymbol{s}^\dagger \boldsymbol{y}_a\right|^2}{\sigma_c^2} \tag{4.6b}$$

$$= Z - \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y}_a \\ y \end{bmatrix}^\dagger \begin{bmatrix} \boldsymbol{J}_{\mathbf{x}} + \frac{\boldsymbol{H}\boldsymbol{H}^\dagger}{\sigma_c^2} & \frac{\boldsymbol{H}^\dagger}{\sigma_c^2} & \boldsymbol{0}_{M\times 1} \\ \frac{\boldsymbol{H}}{\sigma_c^2} & \frac{1}{\sigma_c^2}(\boldsymbol{I}_D + \boldsymbol{s}\boldsymbol{s}^\dagger) & \frac{\boldsymbol{s}}{\sigma_c^2} \\ \boldsymbol{0}_{1\times M} & \frac{\boldsymbol{s}^\dagger}{\sigma_c^2} & \frac{1}{\sigma_c^2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y}_a \\ y \end{bmatrix} . \tag{4.6c}$$

In the above, Z is the normalization constant of the multivariate complex Gaussian, and depends on $\boldsymbol{J}_{\mathbf{x}}$ and σ_c^2 (but nothing else). The noise variance σ_c^2 was defined in (4.2a). $\boldsymbol{I}_D$ refers to the $D \times D$ identity matrix, where D was defined in (4.2b).

To compute $\hat{\boldsymbol{y}}_a^k(n)$, we use the Maximum a-Posteriori (MAP) estimate of $\mathbf{y}_a$ conditioned on $\mathbf{x} = \boldsymbol{x}(n), \mathsf{y} = \boldsymbol{y}(n)$ and with $\boldsymbol{H} = \hat{\boldsymbol{H}}^{k-1}$, i.e., with the current estimate of the parameter. Note that $\hat{\boldsymbol{H}}^{k-1}$ is obtained by populating an $M \times D$ matrix with the estimates $\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}_c}$ and $\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}_s}$ in the manner of (4.4c).

A well known result for multivariate Gaussians is that if

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} \sim \mathcal{CN}\left(\boldsymbol{0}\,,\, \boldsymbol{J}_{\mathbf{z}}^{-1}\right)$$

is a zero-mean Gaussian vector, whose inverse covariance matrix can be partitioned as

$$\boldsymbol{J}_{\mathbf{z}} = \begin{bmatrix} \boldsymbol{J}_{11} & \boldsymbol{J}_{12} \\ \boldsymbol{J}_{12}^\dagger & \boldsymbol{J}_{22} \end{bmatrix} ,$$

then [21, Section 2.3.1]

$$\mathbf{z}_1 \mid (\mathbf{z}_2 = \boldsymbol{z}_2) \sim \mathcal{CN}\left(\boldsymbol{J}_{11}^{-1} \boldsymbol{J}_{12} \boldsymbol{z}_2\,,\, \boldsymbol{J}_{11}^{-1}\right) . \tag{4.7}$$

As $\mathbf{y}_a$ is Gaussian conditioned upon $(\mathbf{x}, \mathsf{y})$, the MAP estimate of $\mathbf{y}_a$ given $\mathbf{x}, \mathsf{y}$ is just the mean of the conditional distribution. Hence,

$$\hat{\boldsymbol{y}}_a^k(n) = \mathbb{E}\left[\mathbf{y}_a \mid \mathbf{x} = \boldsymbol{x}(n), \mathsf{y} = y(n)\,;\, \boldsymbol{H} = \hat{\boldsymbol{H}}^{k-1}\right] \tag{4.8a}$$

$$= \left[\frac{1}{\tilde{\sigma}^2}\left(\boldsymbol{I}_D + \boldsymbol{s}\boldsymbol{s}^\dagger\right)\right]^{-1} \begin{bmatrix} \dfrac{(\hat{\boldsymbol{H}}^{k-1})^\dagger}{\tilde{\sigma}^2} & \dfrac{\boldsymbol{s}}{\tilde{\sigma}^2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(n) \\ y(n) \end{bmatrix} \tag{4.8b}$$

$$= \left(\boldsymbol{I}_D - \frac{1}{1+\boldsymbol{s}^\dagger\boldsymbol{s}}\boldsymbol{s}\boldsymbol{s}^\dagger\right)\left((\hat{\boldsymbol{H}}^{k-1})^\dagger\boldsymbol{x}(n) + \boldsymbol{s}y(n)\right) \tag{4.8c}$$

$$= (\hat{\boldsymbol{H}}^{k-1})^\dagger\boldsymbol{x}(n) + \boldsymbol{s}y(n) - \frac{1}{D+1}\boldsymbol{s}(\hat{\boldsymbol{h}}^{k-1})^\dagger\boldsymbol{x}(n) - \frac{D}{D+1}\boldsymbol{s}y(n) \tag{4.8d}$$

$$= (\hat{\boldsymbol{H}}^{k-1})^\dagger\boldsymbol{x}(n) + \frac{1}{D+1}\left(y(n) - (\hat{\boldsymbol{h}}^{k-1})^\dagger\boldsymbol{x}(n)\right)\boldsymbol{s}\,, \tag{4.8e}$$

where (4.8b) is obtained from (4.7), and (4.8c) is obtained by applying the matrix inversion lemma. Additionally the result that $\boldsymbol{s}^\dagger\boldsymbol{s} = D$ is used. Recall that the estimate $\hat{\boldsymbol{h}}^{k-1}$ of the overall vector $\boldsymbol{h}$ can be obtained using $\hat{\boldsymbol{h}}^{k-1} = \hat{\boldsymbol{H}}^{k-1}\boldsymbol{s}$.

This is a very simple result. Note that as the non-zero entries of $\boldsymbol{H}$ in any column contain the estimate of $\hat{\boldsymbol{h}}_{\text{gem}-\text{ls}_c}$ for the clique (or separator) encoded in that column, this result can be unwrapped for each clique and separator as follows:

$$\hat{y}_c^k(n) = \left(\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}_c}\right)^\dagger \boldsymbol{x}_c(n) + \frac{1}{D+1}e^{k-1}(n),\ c \in \mathcal{C}\,, \tag{4.9a}$$

$$\hat{y}_s^k(n) = \left(\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}_s}\right)^\dagger \boldsymbol{x}_c(n) - \frac{1}{D+1}e^{k-1}(n),\ s \in \mathcal{S}\,, \tag{4.9b}$$

where $e^{k-1}(n) = y(n) - \left(\hat{\boldsymbol{h}}^{k-1}\right)^\dagger \boldsymbol{x}(n)$ is the prediction error in the $k-1$th iteration of the nth symbol, i.e., in the previous iteration.

In other words, the estimate of each clique (and separator) output is the output based on the input and the parameter estimate, plus a portion of the prediction error of the current parameter estimate to account for the fact that the clique outputs need to sum to the overall output $y(n)$ (plus noise). The portions are all equal because the noise on all the cliques has been assumed equal—otherwise the derivation above would be amended by replacing $\boldsymbol{I}_D$ with a diagonal matrix containing the noise variances across the different cliques.

The M-Step: Estimating $\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_c}$

Given $y^k_c(n)$ for every clique and separator, the M-step estimates the parameter $\boldsymbol{h}^k_c$. As previously mentioned, GEM-LS simply computes a least squares solution for each clique and separator. Specifically, for each clique,

$$\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_c} = \left(\sum_{n=1}^{N} \boldsymbol{x}_c(n) \boldsymbol{x}^\dagger_c(n) \right)^{-1} \left(\sum_{n=1}^{N} \boldsymbol{x}_c(n) (\hat{y}^k_c(n))^* \right) , \tag{4.10}$$

and similarly for each separator.

Under the system model of (4.1) and with the Gaussianity assumptions that have been made, $\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_c}$ is the ML estimate of $\boldsymbol{h}_c$, using the current estimate of the (latent) clique output. Equivalently, this estimate minimizes the least squares cost over each clique, using the current (best) estimate of the clique output.

Algorithm Summary

The whole algorithm is presented in Figure 4-4. To simply the notation, the following matrices are defined for the observations:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}(1) & \boldsymbol{x}(2) & \cdots & \boldsymbol{x}(N) \end{bmatrix} \tag{4.11a}$$

$$\boldsymbol{Y} = \begin{bmatrix} y(1) & y(2) & \cdots & y(N) \end{bmatrix} \tag{4.11b}$$

$$\boldsymbol{X}_c = \boldsymbol{X}_{c,:},\ c \in \mathcal{C}, \tag{4.11c}$$

$$\mathbf{Y}_c = \begin{bmatrix} y_c(1) & y_c(2) & \cdots & y_c(N) \end{bmatrix},\ c \in \mathcal{C}\,, \tag{4.11d}$$

and the counterparts of (4.11c) and (4.11d) for $s \in \mathcal{S}$. With these, (4.9) and (4.10) can be rewritten as

$$\hat{\mathbf{Y}}^k_c = \left(\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}_c} \right)^\dagger \boldsymbol{X}_c + \frac{1}{D+1} \left(\boldsymbol{Y} - \left(\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}} \right)^\dagger \boldsymbol{X} \right)^* \tag{4.12a}$$

$$\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_c} = \left(\boldsymbol{X}_c \boldsymbol{X}^\dagger_c \right)^{-1} \boldsymbol{X}_c \left(\hat{\mathbf{Y}}^k_c \right)^\dagger \tag{4.12b}$$

for each $c \in \mathcal{C}$, and similarly for each $s \in \mathcal{S}$.

The implementation of Figure 4-4 uses a fixed number of iterations K. Practical methods will dictate different methods of choosing the number of iterations—this issue will be considered in the next section. Additionally, the implementation of Figure 4-4 is completely serial, but the operations that are performed sequentially for each clique and separator can be performed in parallel, which would lead to a very fast parallel implementation for the algorithm.

The entire development has been founded on the assumption that $\boldsymbol{\mathsf{x}}$ is Gaussian. If the Gaussian assumptions are not valid, neither is (4.10) an ML parameter estimate, nor is the development of Section 4.2 valid, as it hinges on a joint Gaussianity assumption that no longer holds. It may of course be possible, if the exact signal and noise statistics are available, to compute the precise ML solution for the system. That said, in much the same way that the LS solution is often used without regard to the underlying model for reasons of simplicity, the solutions of (4.9) and (4.10) are linear solutions and can be computed efficiently, so there may be a case for exploiting them in the form currently stated, whether or not the precise statistical assumptions hold.

If $N \geq \max_{c\in\mathcal{C}} |c|$, all the inversions required for GEM-LS are well defined, so the algorithm requires $N \geq \max_{c\in\mathcal{C}} |c|$ to produce a unique solution, rather than $N \geq M$ as required for conventional LS. However, it is not yet clear that the estimate of the GEM-LS is meaningful. Even if it were, it is unclear how many iterations would be needed to achieve reasonable performance, so the complexity is unclear. In the following sections, a variety of questions regarding the convergence and performance of the GEM-LS solution are considered.

4.3 Convergence Analysis

Begin by assuming that $N \geq \max_{c\in\mathcal{C}} |c|$, i.e., that the number of observations exceeds the size of the largest clique (it will be assumed henceforth that this always holds); and let

$$\hat{\boldsymbol{J}}_{\boldsymbol{\mathsf{x}}_{\mathcal{G}}}(N) = \sum_{c\in\mathcal{C}} \left[\left(\frac{1}{N}\boldsymbol{X}_c\boldsymbol{X}_c^{\dagger}\right)^{-1}\right]_{c\times c}^{M\times M} - \sum_{s\in\mathcal{S}} \left[\left(\frac{1}{N}\boldsymbol{X}_s\boldsymbol{X}_s^{\dagger}\right)^{-1}\right]_{s\times s}^{M\times M}. \qquad (4.13)$$

```
Require:
    $\boldsymbol{X} = [\boldsymbol{x}(1), \boldsymbol{x}(2), \cdots, \boldsymbol{x}(N)]$
    $\boldsymbol{Y} = [y(1), y(2), \cdots, y(N)]$
    $\mathcal{C}$, $\mathcal{S}$
    $K =$#(iterations)
    starting point $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}$
{Initialization:}
$C \leftarrow |\mathcal{C}|, S \leftarrow |\mathcal{S}|, D \leftarrow C + S$
for $c \in \mathcal{C}$ do
    $\boldsymbol{X}_c \leftarrow \boldsymbol{X}_{c,:}$
    $\hat{\boldsymbol{h}}^0_c \leftarrow \hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}_c}$
end for
for $s \in \mathcal{S}$ do
    $\boldsymbol{X}_s \leftarrow \boldsymbol{X}_{s,:}$
    $\hat{\boldsymbol{h}}^0_s \leftarrow \hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}_s}$
end for
{Main Algorithm:}
for $k = 1$ to $K$ do
    $\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}} \leftarrow \sum_{c\in\mathcal{C}} \left[\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}_c}\right]^{M\times 1}_{c\times 1} - \sum_{s\in\mathcal{S}} \left[\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}_s}\right]^{M\times 1}_{s\times 1}$
    $\boldsymbol{E}^k \leftarrow \boldsymbol{Y} - (\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}})^\dagger \boldsymbol{X}$
    {E-step:}
    for $c \in \mathcal{C}$ do
        $\boldsymbol{Y}^k_c \leftarrow (\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}_c})^\dagger \boldsymbol{X}_c + \boldsymbol{E}^k/(D+1)$
    end for
    for $s \in \mathcal{S}$ do
        $\boldsymbol{Y}^k_s \leftarrow (\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}_s})^\dagger \boldsymbol{X}_s - \boldsymbol{E}^k/(D+1)$
    end for
    {M-step:}
    for $c \in \mathcal{C}$ do
        $\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_c} \leftarrow (\boldsymbol{X}_c \boldsymbol{X}_c^\dagger)^{-1} \boldsymbol{X}_c (\boldsymbol{Y}^k_c)^\dagger$
    end for
    for $s \in \mathcal{S}$ do
        $\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_s} \leftarrow (\boldsymbol{X}_s \boldsymbol{X}_s^\dagger)^{-1} \boldsymbol{X}_s (\boldsymbol{Y}^k_s)^\dagger$
    end for
end for
return $\hat{\boldsymbol{h}}^K_{\text{gem}-\text{ls}} \leftarrow \sum_{c\in\mathcal{C}} \left[\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_c}\right]^{M\times 1}_{c\times 1} - \sum_{s\in\mathcal{S}} \left[\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}_s}\right]^{M\times 1}_{s\times 1}$
```

Figure 4-4: Steps of the graphical expectation-maximization least squares algorithm

Additionally, define the matrix

$$\boldsymbol{\rho}_{\mathbf{x}}(N) = \frac{1}{D+1}\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\hat{\boldsymbol{R}}_{\mathbf{x}}(N)\,. \tag{4.14}$$

The matrix $\boldsymbol{\rho}_{\mathbf{x}}(N)$ holds the key to the update that the GEM-LS algorithm performs in each iteration. The eigenvalues of $\boldsymbol{\rho}_{\mathbf{x}}(N)$ are denoted $\lambda_m, m = 1, 2, \ldots, M$.

Combining (4.12a) and (4.12b), the following is seen for each $c \in \mathcal{C}$,

$$\begin{aligned}\hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}_c} &= \left(\boldsymbol{X}_c\boldsymbol{X}_c^{\dagger}\right)^{-1}\left[\boldsymbol{X}_c\boldsymbol{X}_c^{\dagger}\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}_c} + \frac{1}{D+1}\boldsymbol{X}_c(\boldsymbol{Y}-\hat{\boldsymbol{Y}}^{k})^{\dagger}\right]\\ &= \hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}_c} + \frac{1}{D+1}\left(\boldsymbol{X}_c\boldsymbol{X}_c^{\dagger}\right)^{-1}\boldsymbol{X}_c\left(\boldsymbol{Y}-(\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}})^{\dagger}\boldsymbol{X}\right)^{\dagger}\,,\end{aligned} \tag{4.15}$$

and similarly for each $s \in \mathcal{S}$.

Combining the clique and separator parameter estimate yields

$$\begin{aligned}\hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} &= \sum_{c\in\mathcal{C}}\left[\hat{\boldsymbol{h}}^{k}_{c}\right]^{M\times 1}_{c\times 1} - \sum_{s\in\mathcal{S}}\left[\hat{\boldsymbol{h}}^{k}_{s}\right]^{M\times 1}_{s\times 1}\\ &= \sum_{c\in\mathcal{C}}\left[\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}_c}\right]^{M\times 1}_{c\times 1} - \sum_{s\in\mathcal{S}}\left[\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}_s}\right]^{M\times 1}_{s\times 1}\\ &\quad + \frac{1}{D+1}\left(\sum_{c\in\mathcal{C}}\left[(\boldsymbol{X}_c\boldsymbol{X}_c^{\dagger})^{-1}\boldsymbol{X}_c\right]^{M\times N}_{c,:} - \sum_{s\in\mathcal{S}}\left[(\boldsymbol{X}_s\boldsymbol{X}_s^{\dagger})^{-1}\boldsymbol{X}_s\right]^{M\times N}_{s,:}\right)\left(\boldsymbol{Y}-(\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}})^{\dagger}\boldsymbol{X}\right)^{\dagger}\,.\end{aligned}$$

A little algebra shows that

$$\left(\sum_{c\in\mathcal{C}}\left[(\boldsymbol{X}_c\boldsymbol{X}_c^{\dagger})^{-1}\boldsymbol{X}_c\right]^{M\times N}_{c,:} - \sum_{s\in\mathcal{S}}\left[(\boldsymbol{X}_s\boldsymbol{X}_s^{\dagger})^{-1}\boldsymbol{X}_s\right]^{M\times N}_{s,:}\right) = \frac{1}{N}\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\boldsymbol{X}$$

and so

$$\begin{aligned}\hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} &= \hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}} + \frac{1}{D+1}\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\left[\left(\frac{1}{N}\boldsymbol{X}\boldsymbol{Y}^{\dagger}\right) - \left(\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\dagger}\right)\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}}\right]\\ &= \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\mathbf{x}}(N)\right)\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}} + \frac{1}{D+1}\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\left(\frac{1}{N}\boldsymbol{X}\boldsymbol{Y}^{\dagger}\right)\end{aligned} \tag{4.16}$$

Two cases are now considered.

4.3.1 Case 1: $N \geq M$

In this case, there is a unique least squares solution given by

$$\hat{\boldsymbol{h}}_{\text{ls}} = \left(\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\dagger\right)^{-1}\left(\frac{1}{N}\boldsymbol{X}\boldsymbol{Y}^\dagger\right).$$

Plugging this into (4.16) we obtain

$$\hat{\boldsymbol{h}}^{k}_{\text{gem}-\text{ls}} = \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)\hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}} + \boldsymbol{\rho}_{\boldsymbol{x}}(N)\hat{\boldsymbol{h}}_{\text{ls}}\,. \tag{4.17}$$

In order that this linear system be stable, it is required that all the eigenvalues of $\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)$ have absolute value less than 1. In other words, for every eigenvalue λ_m of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$, it is required that

$$-1 < 1 - \lambda_m < 1, \text{ or } 0 < \lambda_m < 2\,.$$

In order to show that the required property holds, we begin with the following lemma.

Lemma 4.3.1. *The eigenvalues of* $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ *are the same as the eigenvalues of*

$$\frac{1}{D+1}\boldsymbol{P}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{P}^\dagger\,,$$

where $\boldsymbol{P}^\dagger\boldsymbol{P} = \hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)$.

Proof. Since $N \geq \max_{c\in\mathcal{C}}|c|$, the matrix $\hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)$ of (4.13) is almost surely a positive definite Hermitian matrix. Thus, there exists an invertible matrix $\boldsymbol{P}$, so that

$$\boldsymbol{P}^\dagger\boldsymbol{P} = \hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)\,. \tag{4.18}$$

Let λ be an eigenvalue of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ and $\boldsymbol{v}$ be the corresponding eigenvector. Then

$$\begin{aligned}
&\frac{1}{D+1}\hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{v} = \lambda\boldsymbol{v}\\
\Longrightarrow\ &\frac{1}{D+1}\boldsymbol{P}^\dagger\boldsymbol{P}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{P}^\dagger(\boldsymbol{P}^\dagger)^{-1}\boldsymbol{v} = \lambda\boldsymbol{v}\\
\Longrightarrow\ &\frac{1}{D+1}\boldsymbol{P}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{P}^\dagger\left((\boldsymbol{P}^\dagger)^{-1}\boldsymbol{v}\right) = \lambda\left((\boldsymbol{P}^\dagger)^{-1}\boldsymbol{v}\right)
\end{aligned}$$

and the result is shown. □

If $N \geq M$, then $\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)$ is a positive definite Hermitian matrix almost surely, and thus, the matrix $\boldsymbol{P}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{P}^{\dagger}$ is also positive definite Hermitian, so its eigenvalues are all positive (almost surely). Thus, when $N \geq M, \lambda_m > 0$ almost surely.

To upper bound the eigenvalues, the following result from Random Matrix Theory is useful.

Lemma 4.3.2. *As $N \to \infty$ with $M/N = \alpha$ being held constant, the largest eigenvalue of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ converges almost surely to*

$$\lambda_{max} = \frac{1}{D+1}(1+\sqrt{\alpha})^2 \,.$$

Proof. Observe that

$$\begin{aligned} \boldsymbol{P}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{P}^{\dagger} &= \frac{1}{N}(\boldsymbol{P}\boldsymbol{X})(\boldsymbol{P}\boldsymbol{X})^{\dagger} \\ &= \frac{1}{N}\boldsymbol{U}\boldsymbol{U}^{\dagger} \end{aligned}$$

where $\boldsymbol{X}$ was defined in (4.11a). If the graph is correct, it can be shown that the elements of the random matrix $\boldsymbol{U}$ are uncorrelated zero-mean random variables with unit variance.

The distribution of largest eigenvalue of the random matrix $\boldsymbol{U}\boldsymbol{U}^{\dagger}/N$ is universal (independent of the marginal distribution of the elements) provided the matrix entries are zero-mean, unit variance and mutually independent [10].[3] Thus, the classical result on almost sure convergence of the largest eigenvalue for Wishart matrices [69] can be applied, leading to

$$\lambda_{\max} \xrightarrow{\text{a.s.}} \frac{1}{D+1}(1+\sqrt{\alpha})^2 \,, \tag{4.19}$$

[3] The universality result of [10] used in the proof of Lemma 4.3.2 relies on a matrix $\boldsymbol{U}$ containing independent, zero-mean, unit-variance elements, but the columns of the matrix $\boldsymbol{U}$ in the proof of Lemma 4.3.2 are not independent (although the entries are uncorrelated). Nevertheless, various authors have shown that the eigenvalue results that are shown for matrices with independent Gaussian entries apply in a remarkably wide range of settings, including matrices with dependent columns—e.g., [178] and references therein. Indeed, it is easily verified by simulation that, in spite of the deviations of the matrices involved from the theoretical random matrix framework, the result of (4.19) is very accurate.

as $N \to \infty$. □

A consequence of Lemma 4.3.2 is that if $\alpha = M/N \leq 1$, then $\lambda_{\max} < 2$ if $D > 1$. Note that $D = 1$ is the trivial case of a fully connected random vector $\boldsymbol{x}$. In this case, when $N \geq M$, $\hat{\boldsymbol{J}}_{\boldsymbol{x}_\mathcal{G}}(N) = \hat{\boldsymbol{R}}_{\boldsymbol{x}}^{-1}(N)$, and GEM-LS converges to the LS solution in a single iteration. This case is not considered further.

Thus, when $N \geq M$, the GEM-LS converges to the LS solution if enough iterations are run.

4.3.2 Case 2: $N < M$

In this case, there are infinitely many least squares solutions, all of which have the same likelihood. To analyze this case, let $\mathcal{H}$ be the set of all least squares solutions, i.e.,

$$\mathcal{H} = \left\{ \boldsymbol{h} \in \mathbb{C}^M \, : \, (\boldsymbol{X}\boldsymbol{X}^\dagger)\boldsymbol{h} = \boldsymbol{X}\boldsymbol{Y}^\dagger \right\} \,, \tag{4.20a}$$

and let $\mathcal{R}$ be the column space of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$,

$$\mathcal{R} = \left\{ \boldsymbol{r} \in \mathbb{C}^M \, : \, \boldsymbol{r} = \boldsymbol{\rho}_{\boldsymbol{x}}(N)\boldsymbol{t}, \in \mathbb{C}^M \right\} \,. \tag{4.20b}$$

Now, the following can be established.

Theorem 4.3.1. *$\mathcal{H}$ and $\mathcal{R}$ almost surely intersect in a unique point.*

Proof. Let $\mathcal{N}$ be the null space of $\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)$, i.e.,

$$\mathcal{N} = \left\{ \boldsymbol{n} \in \mathbb{C}^M \, : \, \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{n} = 0 \right\}$$

$\mathcal{N}$ and $\mathcal{R}$ intersect almost surely:

Note that if $\boldsymbol{r} \in \mathcal{R}$, then $\boldsymbol{r}$ is orthogonal to the left null space of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$. Let $\boldsymbol{q}$ be in the left null space of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$. Then,

$$\boldsymbol{\rho}_{\boldsymbol{x}}^\dagger(N)\boldsymbol{q} = 0 \implies \hat{\boldsymbol{J}}_{\boldsymbol{x}_\mathcal{G}}(N)\boldsymbol{q} \in \mathcal{N} \,.$$

This means that for any $\boldsymbol{q}$ in the left null space of $\boldsymbol{\rho}_{\mathbf{x}}(N)$, a vector $\boldsymbol{n} = \hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\boldsymbol{q}$ can be found in $\mathcal{N}$ so that

$$\boldsymbol{q}^{\dagger}\boldsymbol{n} = \boldsymbol{q}^{\dagger}\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\boldsymbol{q} > 0$$

almost surely (since $\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)$ is almost surely positive definite because $N \geq \max_{c \in \mathcal{C}} |c|$). This implies that $\mathcal{N}$ is almost surely not orthogonal to the left null space of $\boldsymbol{\rho}_{\mathbf{x}}(N)$, which in turn means that $\mathcal{R}$ and $\mathcal{N}$ are not parallel, and neither one is a subspace of the other. Thus they intersect almost surely.

$\mathcal{N}$ and $\mathcal{R}$ intersect in a point:
Since $\mathcal{R}, \mathcal{N}$ lie in an M-dimensional space, $\dim(\mathcal{R}) = N$, $\dim(\mathcal{N}) = M - N$, and neither $\mathcal{R}$ nor $\mathcal{N}$ is the subspace of the other (almost surely), we have that $\dim(\mathcal{N} \cap \mathcal{R}) = 0$, i.e., they intersect in a point.

$\mathcal{H}$ and $\mathcal{R}$ intersect in a point:
Let $\boldsymbol{h}_m = \boldsymbol{X}(\boldsymbol{X}^{\dagger}\boldsymbol{X})^{-1}\boldsymbol{Y}^{\dagger}$. It is easily verified that $\boldsymbol{h}_m \in \mathcal{H}$, and that, for every $\boldsymbol{h} \in \mathcal{H}, \exists \boldsymbol{n} \in \mathcal{N}$ such that $\boldsymbol{h} = \boldsymbol{h}_m + \boldsymbol{n}$. From this it follows that as $\mathcal{N} \cap \mathcal{R}$ is a unique point, so $\mathcal{H} \cap \mathcal{R}$ is also a unique point, as $\mathcal{H}$ is simply $\mathcal{N}$ shifted by a fixed vector. □

Now, it is simple to see from (4.16) that

$$\hat{\boldsymbol{h}}^{k}_{\text{gem-ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \in \mathcal{R}\,.$$

Thus,

- All GEM-LS updates lie in the column space of $\boldsymbol{\rho}_{\mathbf{x}}(N)$ and
- The fixed point of the update equation must lie in $\mathcal{H}$.

Thus, if GEM-LS converges, then it converges to the intersection of $\mathcal{H}$ and $\mathcal{R}^0$, where

$$\mathcal{R}^0 = \left\{\hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} + \boldsymbol{r} \,:\, \boldsymbol{r} \in \mathcal{R}\right\}\,,$$

i.e., $\mathcal{R}^0$ is the subspace $\mathcal{R}$ shifted to contain the initial vector of the GEM-LS algorithm.

Hence, a point to which the algorithm can converge has been established. The question of whether it does converge remains. In this case, as all the updates are in the column space of

$\boldsymbol{\rho}_{\mathbf{x}}(N)$, GEM-LS converges provided that all the eigenvalues of $\boldsymbol{I}_M - \boldsymbol{\rho}_{\mathbf{x}}(N)$ that correspond to the eigenvectors that span the column space of the matrix have absolute value less than 1. Equivalently, it is required that all the non-zero[4] eigenvalues of $\boldsymbol{\rho}_{\mathbf{x}}(N)$ lie strictly between 0 and 2.

Lemma 4.3.1 can be applied to show that every $\lambda_m \geq 0$ as the matrix $\hat{\boldsymbol{R}}_{\mathbf{x}}(N)$ is positive semi-definite, so all the non-zero eigenvalues of $\boldsymbol{\rho}_{\mathbf{x}}(N)$ are positive. For the upper bound, additional results are required.

Lemma 4.3.3. *The diagonal of $\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\hat{\boldsymbol{R}}_{\mathbf{x}}(N)$ contains all ones.*

Proof. The matrix $\boldsymbol{T}(N) = \hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\hat{\boldsymbol{R}}_{\mathbf{x}}(N)$ can be written as

$$\begin{aligned}
\boldsymbol{T}(N) &= \left(\sum_{c\in\mathcal{C}} \left[\left(\frac{1}{N}\boldsymbol{X}_c\boldsymbol{X}_c^\dagger \right)^{-1} \right]_{c\times c}^{M\times M} - \sum_{s\in\mathcal{S}} \left[\left(\frac{1}{N}\boldsymbol{X}_s\boldsymbol{X}_s^\dagger \right)^{-1} \right]_{s\times s}^{M\times M} \right) (\boldsymbol{X}\boldsymbol{X}^\dagger) \\
&= \sum_{c\in\mathcal{C}} \left\{ \left[\left(\frac{1}{N}\boldsymbol{X}_c\boldsymbol{X}_c^\dagger \right)^{-1} \right]_{c\times c}^{M\times M} (\boldsymbol{X}\boldsymbol{X}^\dagger) \right\} - \sum_{s\in\mathcal{S}} \left\{ \left[\left(\frac{1}{N}\boldsymbol{X}_s\boldsymbol{X}_s^\dagger \right)^{-1} \right]_{s\times s}^{M\times M} (\boldsymbol{X}\boldsymbol{X}^\dagger) \right\} \\
&= \sum_{c\in\mathcal{C}} \boldsymbol{T}_c(N) - \sum_{s\in\mathcal{S}} \boldsymbol{T}_s(N)
\end{aligned}$$

It is easy to verify that each of the matrices $T_c(N)$ has diagonal entries of 1 in the locations corresponding to clique c, and zeros in the other diagonal entries.

The way that the sets $\mathcal{C}$ and $\mathcal{S}$ have been defined so that if a node v appears in C_v cliques, then it appears in $C_v - 1$ separators. Therefore, the vth diagonal entry of the matrix $\boldsymbol{T}(N)$ is the sum of ones from each of the C_v cliques that contains the node v, minus the sum of ones from the $C_v - 1$ separators that contain node v, and thus $T_{vv}(N) = 1$. □

Lemma 4.3.4. *If $D > M/2 - 1$, then $\lambda_m < 2$.*

Proof. From Lemma 4.3.3, the diagonal of the matrix $\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\hat{\boldsymbol{R}}_{\mathbf{x}}(N)$ contains all ones, so $\mathrm{Tr}\left\{\hat{\boldsymbol{J}}_{\mathbf{x}_{\mathcal{G}}}(N)\hat{\boldsymbol{R}}_{\mathbf{x}}(N)\right\} = M$.

[4] The eigenvalues of $\boldsymbol{\rho}_{\mathbf{x}}(N)$ that are equal to 0 are precisely those that correspond to the complement of its column space, and the corresponding eigenvalues of $\boldsymbol{I}_M - \boldsymbol{\rho}_{\mathbf{x}}(N)$ are 1. These do not contribute to the convergence.

Thus, if $D > M/2 + 1$, then $1/(D+1) < 2/M$, so that

$$\operatorname{Tr}\{\boldsymbol{\rho}_{\boldsymbol{x}}(N)\} = \sum_{m=1}^{M} \lambda_m < 2\,.$$

By Lemma 4.3.1, $\lambda_m \geq 0$, which implies that $\lambda_m < 2$. □

Combining Lemmas 4.3.2 and 4.3.4, it is obtained that $\lambda_m < 2$ provided

$$D > \min\left(\frac{1}{2}(1+\sqrt{\alpha})^2 - 1, \frac{M}{2} - 1\right)\,. \tag{4.21}$$

Loosely speaking, the implication of (4.21) is that the less data that is available, the "more structured" the system needs to be for convergence. When $N \geq M$, as previously discussed, any $D > 1$ is sufficient, whereas when $N < M$, the smaller the value of N (relative to M), the larger D needs to be (and hence, the larger the number of cliques and separators that are required in the graph).

Hence, provided (4.21) holds, when $N < M$, the GEM-LS algorithm converges to a unique point. This point lies on the space of least squares solutions $\mathcal{H}$, and is the point where the column space of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ shifted to contain $\hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}$ intersects $\mathcal{H}$. The existence of a unique intersection point is guaranteed almost surely by Theorem 4.3.1.

This is demonstrated in Figure 4-5. This is a noise-free under-determined system, with $N = 2, M = 3$. The set of LS solutions lies on a subspace of dimension 1, i.e., a line; and the column space of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ is a plane, which intersects the line in a point. Clearly, the red line representing the path of the solutions lies in the column space of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$, and the final solution to which GEM-LS converges is the intersection of this plane with the line of LS solutions, as the analysis shows.

Thus, the final convergence point of GEM-LS is

$$\hat{\boldsymbol{h}}^{\infty}_{\text{gem}-\text{ls}} = \begin{cases} \hat{\boldsymbol{h}}_{\text{ls}}, & \text{if } N \geq M \\ \mathcal{H} \cap \mathcal{R}^0, & \text{otherwise} \end{cases}\,. \tag{4.22}$$

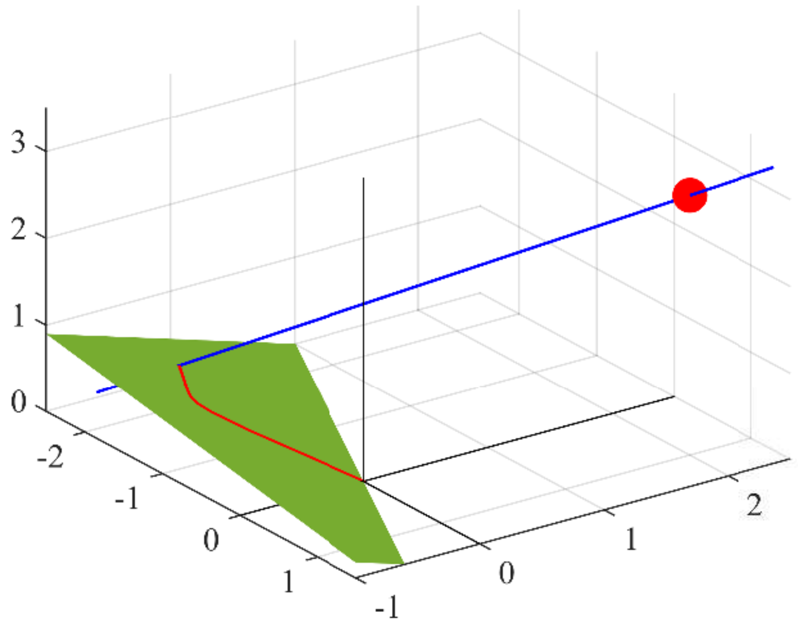


Figure 4-5: An example of how the GEM-LS solution behaves when $N < M$. In this case $M = 3$, $N = 2$ and the system is noisefree. The blue line is the set of LS solutions, the red filled circle is the true vector. The green plane is the column space of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$, and the red line represents the path taken by the GEM-LS algorithm over iterations. In this case $\hat{\boldsymbol{h}}^{0}_{\mathrm{gem-ls}} = \mathbf{0}_{3\times 1}$

4.3.3 Effect of the Wrong Model on Convergence

In the sections above, it has been assumed the graph $\mathcal{G}$ that characterizes $\boldsymbol{x}$ is known exactly—in other words, the GEM-LS algorithm has access to the true data structure. It is possible to consider what happens when the graph that underlies the data is different from the one assumed by the GEM-LS algorithm.

Most of the conditions for convergence are still the same as before. It is still required that all the non-zero eigenvalues of $\boldsymbol{I} - \boldsymbol{\rho}_{\boldsymbol{x}}(N)$ have absolute value less than 1. In other words, the requirement that $0 < \lambda_m < 2$ does not change.

The requirements for this condition to be met, and, in particular, the result of (4.21), need to be updated. Define the matrix

$$\boldsymbol{J}_{\boldsymbol{x}_{\mathcal{G}}} = \lim_{N\to\infty} \hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)\,.$$

When $\mathcal{G}$ is the correct graph then $\boldsymbol{J}_{\boldsymbol{x}_\mathcal{G}} = \boldsymbol{J}_{\boldsymbol{x}}$, but in general this does not hold. It is simple to see that

$$\lim_{N\to\infty} \hat{\boldsymbol{J}}_{\boldsymbol{x}_\mathcal{G}}(N)\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) = \boldsymbol{J}_{\boldsymbol{x}_\mathcal{G}}\boldsymbol{R}_{\boldsymbol{x}}\,.$$

Let $\mu_{\max}$ represent the largest eigenvalue of the above matrix.

Because the trace of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ is M and all eigenvalues λ_m of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ are non-negative (these properties can be shown in exactly the same way as before), it must imply that $\mu_{\max} > 1$. Otherwise, $\mu_{\max}$ would have to be equal to 1, which would mean that all eigenvalues of the matrix above would be equal to 1. This in turn would mean that $\lim_{N\to\infty} \hat{\boldsymbol{J}}_{\boldsymbol{x}_\mathcal{G}}(N) = \boldsymbol{J}_{\boldsymbol{x}}$, i.e., that the correct graph is available.

Then, the proof of Lemma 4.3.2 can be generalized to show that, in the case when the graph assumed to compute the matrix is not the actual graph that characterizes the data, as $N \to \infty$ but $M/N = \alpha$ is held constant, the largest eigenvalue of the matrix $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ converges to

$$\lambda_{\max} \xrightarrow{\text{a.s.}} \frac{1}{D+1}(1+\sqrt{\alpha})^2\mu_{\max}\,. \tag{4.23}$$

The other properties of the eigenvalues of $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ derived in Sections 4.3.1 and 4.3.2 hold. Thus, the condition for ensuring convergence is now

$$D > \min\left(\frac{1}{2}(1+\sqrt{\alpha})^2\mu_{\max} - 1, \frac{M}{2} - 1\right)\,. \tag{4.24}$$

One last caveat is that, when $N < M$, for a unique point of convergence to exist, the matrix $\hat{\boldsymbol{J}}_{\boldsymbol{x}_\mathcal{G}}(N)$ needs to be positive definite (see proof of Theorem 4.3.1). This happens provided $N \geq \max_{c\in\mathcal{C}} |c|$, where $\mathcal{C}$ refers to the set of cliques in the *assumed* graph.

In summary, provided that D satisfies (4.24) and $N \geq \max_{c\in\mathcal{C}} |c|$ the GEM-LS algorithm converges to a unique point even if the wrong graphical model is assumed to characterize the data. It is simple to verify that the point of convergence is given by (4.22), where the definitions of $\mathcal{H}$ and $\mathcal{R}$ in (4.20) continue to hold.

4.4 Improving Upon Least Squares

While convergence is a useful property in and of itself, a stronger result would be if GEM-LS could be shown to outperform the LS algorithm in some regimes. In this section, it is shown that the GEM-LS solution can be considerably better than the conventional LS solution in terms of mean squared error. As it will be shown, for a suitable choice of the iteration, the GEM-LS algorithm performs "shrinkage" along a data dependent curve, which will prove to result in a solution that is more accurate than the LS solution.

It is assumed in this section (and henceforth) that the GEM-LS algorithm has access to the correct graphical model. The analysis of this section is not easy to generalize for the case of incorrect graphical models. In other words, it is not possible in general to get a sense of whether an incorrect graphical model assumption will allow GEM-LS to improve upon LS or not. It certainly converges to the LS solution under some conditions (as has been shown in Section 4.3.3), but beyond that, it is difficult to predict whether improvement is possible or not.

For this section, a different form of (4.16) will be useful. It has been shown that the update equation (4.16) has a unique point of convergence with iterations given by (4.22), and thus

$$\begin{aligned}
\hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} &= \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}} + \frac{1}{D+1}\hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)\left(\frac{1}{N}\boldsymbol{X}\boldsymbol{Y}^{\dagger}\right) \\
&= \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)\hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}} + \boldsymbol{\rho}_{\boldsymbol{x}}(N)\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} && (4.25\mathrm{a}) \\
&= \hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)\left(\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}}\right) \\
\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} &= \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)\left(\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}}\right) && (4.25\mathrm{b}) \\
\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} &= \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)^{k}\left(\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{0}_{\mathrm{gem-ls}}\right) \\
\hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} &= \hat{\boldsymbol{h}}^{0}_{\mathrm{gem-ls}} + \left[\boldsymbol{I}_M - \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)^{k}\right]\left(\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{0}_{\mathrm{gem-ls}}\right)\,, && (4.25\mathrm{c})
\end{aligned}$$

where (4.25a) is obtained by replacing

$$\left(\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\dagger}\right)\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} = \frac{1}{N}\boldsymbol{X}\boldsymbol{Y}^{\dagger}\,.$$

The substitution is valid because $\hat{\boldsymbol{h}}^{\infty}_{\text{gem-ls}} \in \mathcal{H}$ for any N.

4.4.1 Large N and Shrinkage Solutions

To gain insight into why GEM-LS can improve performance, begin with a somewhat artificial case that can be analyzed exactly. An intuitive explanation for the general case is provided in the next section.

The assumption made is that M is held constant and N is allowed to grow very large, $N \gg M$, so that $\alpha = M/N \approx 0$. As $N > M$, it follows that $\hat{\boldsymbol{h}}^{\infty}_{\text{gem-ls}} = \hat{\boldsymbol{h}}_{\text{ls}}$. In order that $\hat{\boldsymbol{h}}_{\text{ls}}$ does not just collapse to the true vector $\boldsymbol{h}_0$, assume that as N increases, the noise power σ^2 also increases so that $\sigma^2/N = \beta$ is a constant. Applying this regime to the result of (2.12a) [130, Sec. III-B], it is seen that

$$\mathbb{E}\left[(\boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\text{ls}})(\boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\text{ls}})^{\dagger}\right] = \beta \boldsymbol{J}_{\boldsymbol{x}} \tag{4.26}$$

When N is very large, the following result applies to the matrix $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ independent of the amount of noise in the system.

Lemma 4.4.1. *Provided that the distribution of $\boldsymbol{x}$ factors according to the decomposable graphical model $\mathcal{G}$,*

$$\lim_{N\to\infty} \boldsymbol{\rho}_{\boldsymbol{x}}(N) = \frac{1}{D+1}\boldsymbol{I}_M$$

Proof. The result follows because:

- $\lim_{N\to\infty} \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) = \boldsymbol{R}_{\boldsymbol{x}}$ by the Strong Law of Large Numbers

- $\lim_{N\to\infty} \hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N) = \boldsymbol{J}_{\boldsymbol{x}}$ [181], provided the graphical model used to compute the inverse covariance matrix truly does characterize the data.

- $\boldsymbol{R}_{\boldsymbol{x}} = \boldsymbol{J}_{\boldsymbol{x}}^{-1}$

Since $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ is the product of two sequences of matrices each with a well-defined finite limit, the limit of the product is the product of the limits. □

Using the fact that $\hat{\boldsymbol{h}}^{\infty}_{\text{gem}-\text{ls}} = \hat{\boldsymbol{h}}_{\text{ls}}$ in this regime and Lemma 4.4.1, (4.25c) becomes

$$\hat{\boldsymbol{h}}^{k}_{\text{gem}-\text{ls}} = \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}} + \left[1 - \left(\frac{D}{D+1}\right)^{k}\right] \left(\hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}\right) . \tag{4.27}$$

Evidently, the solution at each iteration lies on the straight line connecting the starting point $\hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}$ and the eventual point of convergence—in this case, $\hat{\boldsymbol{h}}_{\text{ls}}$, i.e., the set of points given by

$$\mathcal{L} = \left\{\boldsymbol{h} \,:\, \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}} + \kappa \left(\hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}\right) ,\, \kappa \in \mathbb{R}\right\} . \tag{4.28}$$

Immediately, one can see why stopping at an intermediate iteration may help. It has been long understood that *shrinkage*, i.e., shrinking the LS solution towards a fixed point along a line can result in a new solution that *dominates* the LS solution in MSE, meaning that its MSE is strictly better than the MSE of the LS solution. While the term shrinkage estimation has grown to include various kinds of regularization and penalization methods, including ℓ_1 and ℓ_2 penalized LS methods as described in Section 2.1.4, the classical result of shrinkage of an LS estimator towards a fixed point [151] is being considered here.

From simple geometric considerations, the point on the line joining $\hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}$ and $\hat{\boldsymbol{h}}_{\text{ls}}$ that minimizes the distance to $\boldsymbol{h}_0$ is given by

$$\hat{\boldsymbol{h}}_{\text{opt-shrink}} = \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}} + \kappa_{\text{opt}}(\hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}) , \tag{4.29a}$$

where

$$\kappa_{\text{opt}} = \frac{\Re\left\{\left(\hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}\right)^{\dagger} \left(\boldsymbol{h}_0 - \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}\right)\right\}}{\left|\hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}\right|^{2}} . \tag{4.29b}$$

This oracle shrinkage estimator can not be computed in reality, but its performance is a lower bound for any estimator that performs shrinkage along the line through $\hat{\boldsymbol{h}}_{\text{ls}}$ and $\hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}$, so it is a suitable comparison point.

Equation (4.27) implies that, for every k, the estimate $\hat{\boldsymbol{h}}^{k}_{\text{gem}-\text{ls}}$ lies on the line $\mathcal{L}$ for some $0 < \kappa < 1$. Thus, it can be inferred that if $0 < \kappa_{\text{opt}} < 1$, then there is some iteration for which GEM-LS outperforms LS. In particular, if $0 < \kappa_{\text{opt}} < 1$, then (4.27) implies that when

$$\kappa_{\text{opt}} = 1 - \left(\frac{D}{D+1} \right)^{k},$$

or, equivalently, when

$$k_{\text{opt}} = \text{round} \left(\frac{\log(1 - \kappa_{\text{opt}})}{\log D - \log(D+1)} \right) \tag{4.30}$$

the performance of GEM-LS is as good as the oracle shrinkage solution of (4.29a). Note of course that as k_{opt} can only take integer values, the performance is not exactly the same for the optimal GEM-LS iteration and optimum shrinkage, but assuming that the step size is not very large (which is true), the performance should be very close.

It can be argued that the average value of κ_{opt} is between 0 and 1, as follows:

$$\begin{aligned}
\mathbb{E}\left[\kappa_{\text{opt}}\right] &= \mathbb{E}\left[\frac{\Re\left\{ \left(\hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right)^{\dagger} \left(\boldsymbol{h}_0 - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right) \right\}}{\left| \hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right|^{2}} \right] \\
&\approx \frac{\mathbb{E}\left[\Re\left\{ \left(\hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right)^{\dagger} \left(\boldsymbol{h}_0 - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right) \right\} \right]}{\mathbb{E}\left[\left| \hat{\boldsymbol{h}}_{\text{ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right|^{2} \right]} && \text{(4.31a)} \\
&= \frac{\left| \boldsymbol{h}_0 - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right|^{2}}{\left| \boldsymbol{h}_0 - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right|^{2} + \beta \text{Tr}\left\{ \boldsymbol{J}_{\boldsymbol{x}} \right\}}, && \text{(4.31b)}
\end{aligned}$$

where (4.31a) is obtained by the application of a first-order Taylor series approximation to the ratio of random variables [163, Pg. 351], and the result of (4.26) has been applied in the denominator.

Thus, on average (and to a first-order approximation) some finite iteration exists for which the GEM-LS algorithm is closer to the true parameter vector than is the LS solution. Additionally, an insight gained from (4.30) is that as κ_{opt} decreases, k_{opt} decreases. Equation (4.31b) shows that, on average, κ_{opt} decreases as the value of $\beta \text{Tr}\left\{ \boldsymbol{J}_{\boldsymbol{x}} \right\} / \left| \boldsymbol{h}_0 - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} \right|^{2}$ increases. Loosely speaking this implies the following:

- As the value of β increases (i.e., the noise power per sample increases), the best iteration decreases.

- As the starting point gets closer to the true vector, the best iteration decreases.

It is also well understood that these are also the conditions under which a linear shrinkage solution provides the most improvement over the LS solution. Thus, the GEM-LS algorithm behaves the same way as linear shrinkage in this regime.

It may appear strange that the GEM-LS algorithm has been designed to exploit graphical model structure, but the performance improvements do not appear linked to the graphical model at all. However, it is very important to note that the mathematical results of this section is for the rather artificially constructed regime in which $N \to \infty$ and $\sigma^2 \to \infty$. In reality, as described next, the performance of GEM-LS is expected to be better than shrinkage along a line; and the improvements are linked to the model.

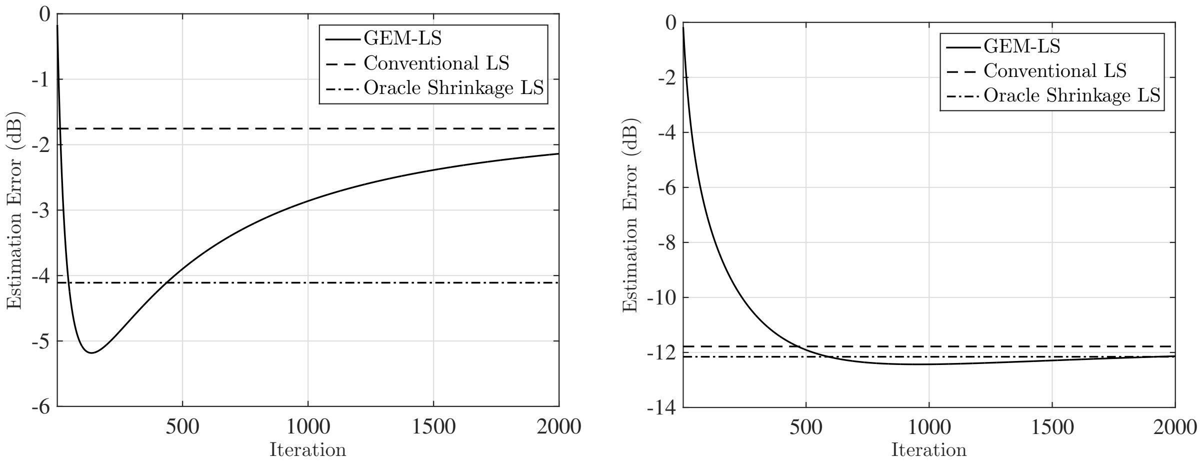
4.4.2 Finite N: Shrinkage of Sub-Problems

Equation (4.25c) gives the path described by the GEM-LS solutions as

$$\hat{\boldsymbol{h}}^{k}_{\text{gem-ls}} = \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}} + \left[\boldsymbol{I}_M - \left(\boldsymbol{I}_M - \boldsymbol{\rho}_{\boldsymbol{x}}(N)\right)^k\right]\left(\hat{\boldsymbol{h}}^{\infty}_{\text{gem-ls}} - \hat{\boldsymbol{h}}^{0}_{\text{gem-ls}}\right) .$$

In the general case, this equation defines a curve in the $\mathbb{C}^M$ space. Attempts to characterize the properties of this curve are complicated by the fact that both $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ and $\hat{\boldsymbol{h}}^{\infty}_{\text{gem-ls}}$ are *dependent*, data driven quantities. In other words, the shape of the curve depends upon its end-points in a fairly non-trivial way, in contrast to the straight line path that the GEM-LS solutions describe for large N.

Thus, for finite N and finite number of iterations, the GEM-LS algorithm for may be seen as performing shrinkage along a data-driven curve, rather than a straight line. The natural question that arises is whether such shrinkage is better than shrinkage along a line in general. To answer this, consider the simulation results of Figure 4-6, which has the performance of the GEM-LS, conventional LS and the oracle shrinkage algorithm of (4.29) as a function of GEM-LS iteration for $M = 50, N = 75$ and where the inputs are generated from a Gaussian graph with all independent nodes. Of course, LS and optimally shrunk LS are not iterative,



(a) SNR=5 dB

(b) SNR=15 dB

Figure 4-6: Performance of GEM-LS as a function of iteration with $M = 50, N = 75$ and SNRs of 5 dB and 15 dB averaged over 500 trials. The random vector $\boldsymbol{x}$ is drawn from a Gaussian graph with 50 independent nodes. The performance of conventional LS and the oracle shrunk LS of (4.29) are also plotted.

so their performances are plotted as horizontal lines for reference. Note also that any other choice of linear shrinkage along the line joining $\hat{\boldsymbol{h}}^{0}_{\text{gem}-\text{ls}}$ and $\hat{\boldsymbol{h}}_{\text{ls}}$ has a higher MSE than the oracle by definition because the oracle is the point on the line that minimizes the distance to $\boldsymbol{h}_0$. The plots show that for these parameters and a suitable choice of iteration, GEM-LS improves upon both LS and the best possible linear shrinkage estimator.

Properties of the Path Described by GEM-LS Solutions

To understand the properties of the path described by the GEM-LS solution vectors as k (the iteration index) varies, assume that $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ has M linearly independent eigenvectors, so that it can be eigendecomposed as

$$\boldsymbol{\rho}_{\boldsymbol{x}}(N) = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\,,$$

where $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M)$ is a diagonal matrix containing the eigenvalues of the matrix and $\boldsymbol{V}$ contains the corresponding eigenvectors in the columns.

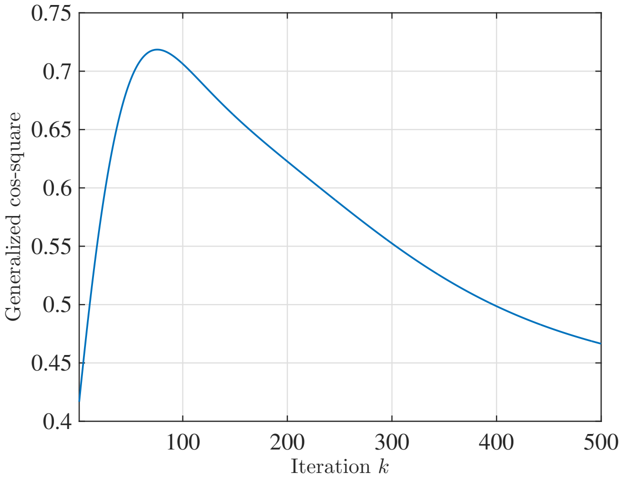


Figure 4-7: Generalized cos-squared between the LS solution $\hat{\boldsymbol{h}}_{\text{ls}}$ and the GEM-LS correction in the k^{th} iteration, $\delta^k = \hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}} - \hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}}$ $\cos^2(\hat{\boldsymbol{h}}_{\text{ls}}; \boldsymbol{\delta}^k)$ as a function of iteration k. In this case $M = 50, N = 75$ and SNR $= 5$ dB, the input process is white and $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}} = \mathbf{0}$. The smoothness of variation of the angle is indicative of the fact that the path followed by the GEM-LS solutions is smooth—a consequence of the analytic nature of the exponential function.

Then, (4.25c) can be written as

$$\begin{aligned}\boldsymbol{V}^{-1}\hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}} &= \boldsymbol{V}^{-1}\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}} + \left[\boldsymbol{I}_M - (\boldsymbol{I}_M - \boldsymbol{\Lambda})^k\right]\boldsymbol{V}^{-1}\left(\hat{\boldsymbol{h}}^\infty_{\text{gem}-\text{ls}} - \hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right) \\ &= \boldsymbol{V}^{-1}\hat{\boldsymbol{h}}^\infty_{\text{gem}-\text{ls}} - \begin{bmatrix} (1-\lambda_1)^k \tilde{\boldsymbol{v}}_1(\hat{\boldsymbol{h}}^\infty_{\text{gem}-\text{ls}} - \hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}) \\ (1-\lambda_2)^k \tilde{\boldsymbol{v}}_2(\hat{\boldsymbol{h}}^\infty_{\text{gem}-\text{ls}} - \hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}) \\ \vdots \\ (1-\lambda_M)^k \tilde{\boldsymbol{v}}_M(\hat{\boldsymbol{h}}^\infty_{\text{gem}-\text{ls}} - \hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}) \end{bmatrix},\end{aligned} \tag{4.32}$$

where $\tilde{\boldsymbol{v}}_m$ is the m^{th} row of $\boldsymbol{V}^{-1}$.

Thus, when transformed by $\boldsymbol{V}^{-1}$, the path of solutions is described by exponential functions in separate dimensions. In this transformed space, therefore the solutions follow a smooth path as k varies as the exponential function is analytic (infinitely differentiable). As transformation by $\boldsymbol{V}^{-1}$ does not affect the smoothness of the curve (the mapping is linear), the path of solutions is a smooth curve in $\mathbb{C}^M$.

Figure 4-7 verifies this smoothness property. The GEM-LS algorithm is run with $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}} = \mathbf{0}$, and the square of the generalized cosine between $\hat{\boldsymbol{h}}_{\text{ls}}$ and $\boldsymbol{\delta}^k = \hat{\boldsymbol{h}}^k_{\text{gem}-\text{ls}} - \hat{\boldsymbol{h}}^{k-1}_{\text{gem}-\text{ls}}$ is plotted, which is given by

$$\cos^2(\hat{\boldsymbol{h}}_{\text{ls}}; \boldsymbol{\delta}^k) = \frac{\left|\hat{\boldsymbol{h}}_{\text{ls}}^{\dagger}\boldsymbol{\delta}^k\right|^2}{|\boldsymbol{h}_0|^2\,|\boldsymbol{\delta}^k|^2}\,. \tag{4.33}$$

The idea is just to plot the angle between the correction in any iteration and a fixed vector (in this case, $\hat{\boldsymbol{h}}_{\text{ls}}$). The angle varies smoothly, indicating that in any given realization, the solution path is smooth as expected.

Moreover, in the transformed space, different dimensions of the parameter vector converge at different rates controlled by the corresponding eigenvalue. This leads to the intuition that perhaps it is possible to relate the properties of the curve to the error and thereby obtain intuition about the optimum iteration.

This turns out to be challenging, as in order to say more about the behavior of the error mathematically, it is necessary to understand the curve in the original space. This would require a more precise characterization of the matrix $\boldsymbol{V}^{-1}$, which contains the eigenvectors of a non-Hermitian random matrix that does not belong to a well-studied class of random matrices. While there exist general results on the eigenvectors of arbitrary random matrices (e.g., [113], [166]), identities that would allow for an easy study of the properties of the solution curve do not appear to have been obtained. In particular, $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ is obtained by the multiplication of two highly dependent random matrices; it is quite unclear how the eigenspace of such a product may behave.

As a mathematical characterization is hard to obtain, an intuitive characterization of the operation of the GEM-LS algorithm is considered in order to understand the behavior.

Shrinkage of Sub-Problems

Consider what the GEM-LS algorithm is attempting to do—namely, to separate the single LS algorithm into smaller LS algorithms over each clique and separator. As an example, take the simple case when the input is white. Suppose the oracle shrinkage estimator were computed independently for every dimension. Then, as all the problems are one-dimensional, it is trivial to show that

$$\hat{h}_{\text{opt-shrink}_m} = \begin{cases} h_{0m}, & \text{when } |h_{0m}| \leq \left|\hat{h}_{\text{ls}_m}\right| \\ \hat{h}_{\text{ls}_m}, & \text{otherwise}\,, \end{cases}$$

where the subscript m refers to the mth element of the vector. In other words, the oracle shrinkage estimator can be perfect in any dimension.

Suppose now that the GEM-LS algorithm were truly able to split the problem into a set of one-dimensional LS problems, and the optimum iteration number was known for each of those problems. In that case, it is clear that the combined GEM-LS solution can be considerably better than either conventional LS or the oracle shrinkage LS algorithm.

Of course, in reality, the GEM-LS algorithm does not compute the solution to all the separate LS problems independently, as it does not have access to the individual clique outputs y_c^k (recall that the process being assumed here is white)—rather, it estimates these jointly using the previous estimate of the parameter vector. Therefore, in each iteration, the GEM-LS solution steps towards the LS solution in each dimension with a step size influenced by the overall estimation error. It may be concluded that the algorithm is unlikely to attain the optimum shrinkage solution for each dimension, but may trade-off better solutions in some dimensions for worse solutions in others.

Figure 4-8 illustrates these concepts. Evidently, the GEM-LS algorithm gets very close to the optimum shrinkage (i.e., the true vector) along both h_0 and h_1 dimensions. However, it can not get the exact shrinkage solution because the solutions in both dimensions move jointly (although at different rates). Nonetheless, around the optimum iteration, it is clear that GEM-LS performs very well in this case.

In the general case, it is hypothesized that the same insight holds: GEM-LS has the potential to perform well if the iteration is optimally chosen because it separates a large problem into smaller subproblems, and each of those subproblems can have an optimum shrinkage solution much closer to the true parameter vector in those dimensions than is possible for the overall problem. This can be shown rigorously only in very limited settings.

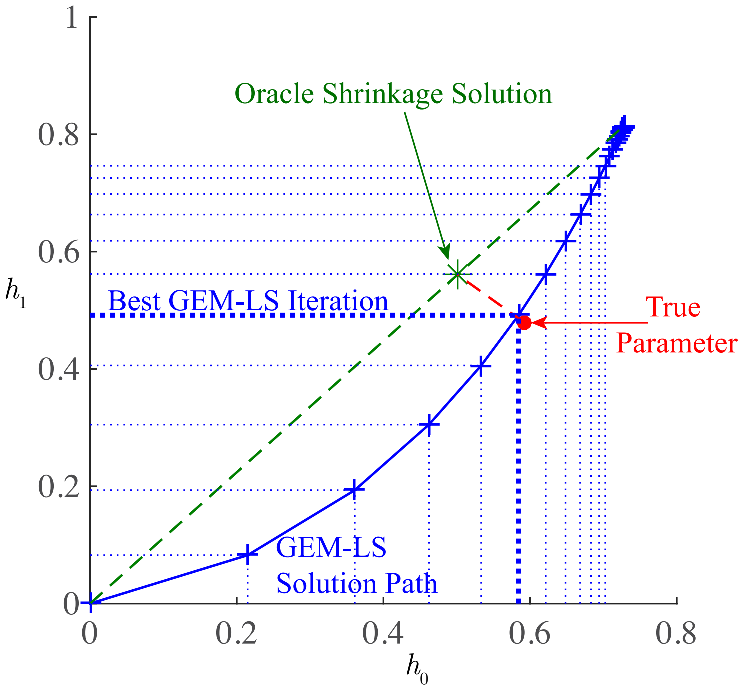


Figure 4-8: An example of the mechanism by which GEM-LS can get a better solution than LS or oracle shrinkage. This system has $M = 2, N = 3$ and SNR $= 0$ dB. The solid blue line is the GEM-LS solution (markers represent solutions at different points), the solid red circle is the true parameter, the dashed green line is the straight line from the origin (starting point for GEM-LS) to the LS solution and the red dashed line shows how to get the oracle shrinkage solution (shown with a green asterisk). The thick blue dotted line shows the best GEM-LS iteration, wherein the algorithm achieves near optimal shrinkage on both dimensions.

For instance, it can be shown that if $N > M$ and

$$D < \frac{\sigma^2 \mathrm{Tr}\left\{\boldsymbol{J_x}\right\} + \left|\boldsymbol{h}_0\right|^2 \left(N - \max_{c \in \mathcal{C}} |c|\right)}{\sigma^2 \mathrm{Tr}\left\{\boldsymbol{J_x}\right\} + \left|\boldsymbol{h}_0\right|^2 \left(N - M\right)}, \tag{4.34}$$

then GEM-LS outperforms the oracle shrinkage estimator for some choice of iteration number. However, this condition is sufficient, not necessary, and it turns out in practice to be far too pessimistic to be useful in terms of choosing regimes to operate the algorithm in, although it may provide insight. Making the bounds tighter is complicated by two issues: the presence of separators which correlate the clique outputs, and the inability to meaningfully bound the parameter vector over the cliques.[5]

Nonetheless some very interesting insights about the algorithm operation are obtained as a result of the analysis above. The presence of the clique-separator structure allows the GEM-LS algorithm to meaningfully separate the problem into a number of sub-problems, as it is this structure that allows for the correct estimation of clique and separator outputs. Once separated, the algorithm can then potentially exploit shrinkage in the smaller problems, albeit with some tradeoffs, while simultaneously maintaining consistency across the correlated parts of the problem using the separator structure. GEM-LS thus allows for an efficient way to simultaneously consider a fairly complicated set of tradeoffs amongst the subproblems. Overall, it appears that the potential for gain is quite large over even an oracle shrinkage estimator, if the iteration is intelligently chosen. How such an iteration choice should be made is considered next.

4.4.3 Practical Choice of Stopping Iteration

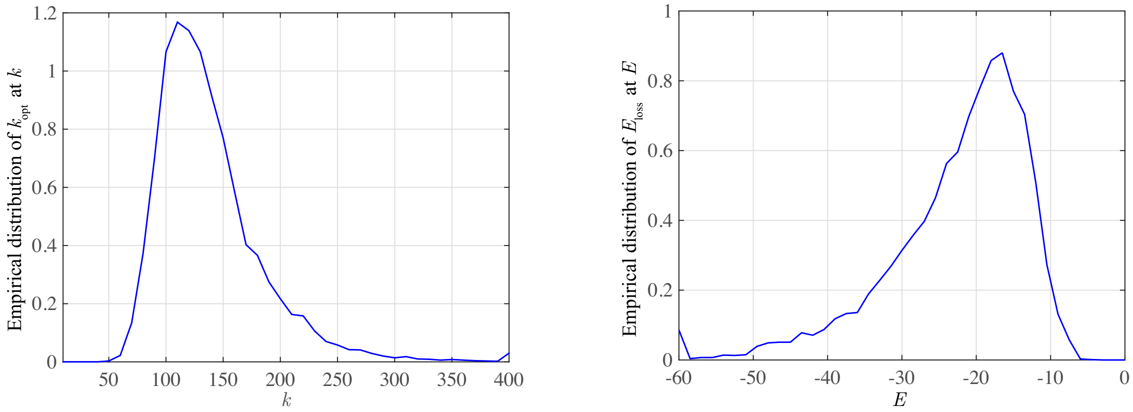
In Sections 4.4.1 and 4.4.2 it has been shown that GEM-LS can potentially improve upon both the LS solution and a comparable oracle linear shrinkage solution for an optimum choice of iteration. However, it has not been explained how such a choice should be made in practice.

Consider Figure 4-9. In Figure 4-9a is plotted the empirical distribution of k_{opt}, i.e., the iteration for GEM-LS that minimizes the MSE, as obtained from simulating $10,000$ trials of a system where $M = 50, N = 75$ and the input is characterized by a graph with M independent nodes. k_{opt} is fairly peaked[6] about its empirical mean $\hat{k}_{\text{opt}}$ (which in this case is about 139). However, there appears to be sufficient variance in this regime that it is not trivial to simply replace k_{opt} by its mean. Put a different way, there is no strong concentration of k_{opt} about its mean in the typical regime of interest.

However, Figure 4-9b shows that the performance loss from approximating $k_{\text{opt}} \approx \hat{k}_{\text{opt}}$

[5] Specifically, one issue that arises while trying to improve the bounds is that the performance of the GEM-LS algorithm (and optimum shrinkage) for any clique is dependent on the true parameter vector of that clique $\boldsymbol{h}_{0_c}$. Without further assumptions about $\boldsymbol{h}_0$, it is difficult to make any global inequality statements about $\boldsymbol{h}_{0_c}$, other than $|\boldsymbol{h}_{0_c}|^2 \leq |\boldsymbol{h}_0|^2$, which is used in obtaining (4.34).

[6] The excess kurtosis of the empirical distribution is $\gamma_2 \approx 5$, indicating that the distribution is more "peaky" than a standard normal distribution.



(a) Distribution of optimum iteration k_{opt}

(b) Distribution of performance loss E_{loss} defined in (4.35)

Figure 4-9: Empirical distributions of k_{opt} and the performance loss E_{loss} sustained by replacing k_{opt} by its expected value, for a system with $M = 50, N = 75$ and the graph contains all independent nodes. While k_{opt} does not strongly concentrate about its mean, the performance loss by using $\hat{k}_{\text{opt}}$ in the place of k_{opt} is not very large.

in the GEM-LS algorithm may not be too large. The figure shows the distribution of the performance loss caused by replacing the optimum iteration by its expected value, defined as

$$E_{\text{loss}} = 10 \log_{10} \left(\frac{\left| \boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\text{gem}-\text{ls}}^{\hat{k}_{\text{opt}}} \right|^2 - \left| \boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\text{gem}-\text{ls}}^{k_{\text{opt}}} \right|^2}{\left| \boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\text{gem}-\text{ls}}^{k_{\text{opt}}} \right|^2} \right). \tag{4.35}$$

The figure shows that the typically expected value of E_{loss} is quite small (about -17 dB, meaning that the expected performance loss is at least 1 or 2 orders of magnitude less than the error). It may be expected therefore that using $\hat{k}_{\text{opt}}$ as a surrogate for k_{opt} can lead to an estimator with good performance.

The advantage of being able to use $\hat{k}_{\text{opt}}$ to approximate the optimum iteration is that it can be obtained by an off-line simulation. In other words, suppose a problem is characterized by a particular graph $\mathcal{G}$. It is possible to run a large number of simulated trials with data characterized by $\mathcal{G}$ for the regimes of interest compute the average optimum iteration off-line and store all the values in a look-up table. Then, the optimum stopping iteration can be read off this table given the set of parameters that occurs in practice.

In order to define the parameters of the simulation (i.e., the possible inputs to the look-up table), it is necessary to consider what the optimum iteration can depend upon. Sections 4.4.1 and 4.4.2 indicate that the optimum iteration is related to the optimum amount of shrinkage necessary, which depends upon the noise power σ^2, the distance $\left|\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}} - \boldsymbol{h}_0\right|^2$, and the inverse covariance matrix $\boldsymbol{J}_{\boldsymbol{x}}$, M and α. However, in a practical environment, most of these parameters are not available to use as inputs to the look up table, so generating a look-up table as a function of all of these parameters is a meaningless exercise. Instead, the simulation is carried out for:

- A set of values of M

- A set of values of SNR

- A set of values of α.

Note that the simulation is carried out separately for each graphical model of interest. In other words, applications that are characterized by different graphical models do require separate off-line simulations to generate the look-up table.

On the other hand, for reasons of practicality, the same look-up table is used for all problems that satisfy a graphical model structure without regard to the true inverse covariance matrix. For example, the results of the look-up table generated using a white input process is used for all input processes with a diagonal inverse covariance matrix (whether white or not). There are two reasons for this. First, it is not guaranteed in every application that the precise inverse covariance matrix of the input data would be known. Even if it were, generating the look-up table is a time-consuming task. Thus, some mismatch is accepted and the look-up table is computed using some typical inverse covariance matrix that satisfies the model structure.

To exemplify the generated look-up table, the results of the proposed simulation are shown in Table 4-10 for an input graphical model with M independent nodes for several different values of M (in particular, a white input process is used in the simulation). While there are a few statistical inconsistencies in the dataset due to averaging and the fixed maximum iteration that is set, the general trend is quite clear—as SNR is decreased or α is increased (i.e., the amount of data is decreased), $\hat{k}_{\text{opt}}$ decreases. This gels with the insights from the shrinkage interpretation—the lower the SNR or fewer the number of snapshots, the fewer the iterations that are needed, which corresponds to more aggressive shrinkage along each clique and separator (in this simple case, along each dimension of the parameter vector). While the simulation provides the value of $\hat{k}_{\text{opt}}$ at a set of pre-determined points in the M, α, SNR space, the value of $\hat{k}_{\text{opt}}$ at any different operating point is obtained using interpolation. In all the practical implementations henceforth, spline interpolation is used to obtain $\hat{k}_{\text{opt}}$ at any required M, α and SNR, which adds a small computational overhead.

Finally, for practical reasons, a cap is placed the number of iterations that will be run. This is most relevant to the high SNR, large number of snapshot regime, wherein the "optimal" choice of GEM-LS iteration may be quite large. In that case, the performance of GEM-LS may not match that of LS—however, provided the cap is chosen judiciously, the regime in which the loss of performance due to running insufficient iterations can become an issue is also the regime in which GEM-LS does not provide a large benefit over using LS since LS is already quite accurate in the high SNR, large data regime (referring to the shrinkage estimation, the worse LS is, the more there is to gain by using GEM-LS). In such regimes, using LS may be preferable to running a large number of iterations.

Why Not A Data-Driven Stopping Criterion?

It may naturally occur to the reader that, rather than surrogating the optimum iteration with its expected value and justifying that replacement, it would be better to obtain a data-driven method of obtaining the stopping criterion. It is evident that, if it were possible to determine the stopping criterion using a data driven approach, such a method would be far more robust to model mismatch than the simulation based approach that has been described. However, it has not been possible to determine a suitable data-driven criterion that was practically feasible.

Broadly speaking, two different kinds of data-driven approaches were considered. The first category were methods that involved choosing the iteration of the GEM-LS algorithm that optimally predicted new data. For instance, suppose that some N_{pred} inputs and their corresponding outputs were available that are not part of the dataset used for estimation,

SNR α	**5dB**	**10dB**	**15dB**	**20dB**	**25dB**	**30dB**
0.25	130	205	255	277	292	298
0.5	99	210	278	297	300	300
0.75	79	200	291	300	300	300
1	62	179	285	300	300	300
1.5	40	115	224	280	294	297
2	32	88	158	200	231	242
3	26	63	92	108	115	116
4	24	47	71	83	82	87

(a) $M = 25$

SNR α	**5dB**	**10dB**	**15dB**	**20dB**	**25dB**	**30dB**
0.25	215	293	300	300	300	300
0.5	181	295	300	300	300	300
0.75	134	289	300	300	300	300
1	104	268	300	300	300	300
1.5	68	190	289	300	300	300
2	48	131	229	281	293	296
3	31	84	142	177	200	206
4	30	70	109	134	142	145

(b) $M = 50$

SNR α	**5dB**	**10dB**	**15dB**	**20dB**	**25dB**	**30dB**
0.25	300	300	300	300	300	300
0.5	292	300	300	300	300	300
0.75	258	300	300	300	300	300
1	203	300	300	300	300	300
1.5	133	291	300	300	300	300
2	96	239	299	300	300	300
3	59	141	231	277	291	295
4	45	103	167	205	223	228

(c) $M = 100$

Figure 4-10: Example of the look-up table that contains $\hat{k}_{\text{opt}}$ for the GEM-LS algorithm, $M = 25, 50, 100$, for a graph with all independent nodes. The maximum iteration is 300.

i.e., we had access to some $\tilde{\boldsymbol{x}}(n)$ and $\tilde{y}(n)$ that satisfied

$$\tilde{y}(n) = \boldsymbol{h}_0 \tilde{\boldsymbol{x}}(n) + \tilde{v}(n),\, n = 1, 2, \ldots, N_{\text{pred}}\,.$$

Then methods such as choosing the iteration that minimizes the prediction error over this dataset, i.e., picking

$$\hat{\boldsymbol{h}}_{\text{gem}-\text{ls}}^{\text{pred-error}} = \arg\min_{k} \sum_{n=1}^{N_{\text{pred}}} \left| \tilde{y}(n) - \hat{\boldsymbol{h}}_{\text{gem}-\text{ls}}^{k} \tilde{\boldsymbol{x}}(n) \right|^2 ,$$

or that maximizes the empirical mutual information between the true and predicted outputs can be considered. Indeed, if such a "prediction dataset" is available, then these methods are quite accurate. They are also useful when operating in a recursive framework.

The trouble with this, of course, is that new data is being introduced. The same data cannot be used for both estimation and computation of the prediction error.[7] If an extra prediction dataset is not available, then the N samples of data that are used for estimation would need to be divided into separate datasets: one that is used as the input to the GEM-LS algorithm and the other to predict the iteration. However, given that the regime of interest is sample-limited, this is not a good approach, as the performance of both the problems of estimation as well as stopping criterion tracking will be compromised.

The second strategy that was considered was to try to find a data-driven statistic that captures some property of the curve of the GEM-LS solutions and to relate that curve to the properties of the optimal GEM-LS solution. However, there was not found to be an easily exploitable link between the shape of the curve and the optimal iteration.

Having explored these strategies, it was concluded that among the methods, the one that most reliably performed well was to pre-simulate the optimum iteration for the given model and regime of interest. Nevertheless, determining a good data-driven method to predict the stopping point remains one of the biggest open questions that remains about the GEM-LS algorithm.

[7] It is trivially true that if the entire dataset is used to compute the prediction error, then the best possible solution simply corresponds to LS estimation if $N > M$. Similar results are obtained for empirical mutual information.

4.4.4 Computational Complexity

Thus far, many properties of the GEM-LS algorithm have emerged from the analysis. However, one point that has not been addressed is: what is the run-time complexity of the GEM-LS algorithm? The reason that this issue has been deferred up until this point is that the complexity of the algorithm is dependent upon the number of iterations that are required to be run. For comparison, the complexity of the LS algorithm is $\mathcal{O}\left(M^2 \max(M, N)\right)$.

To obtain the per iteration complexity of the GEM-LS algorithm, begin by assuming that $D \max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(M\right)$. Nearly every graph of interest appears to fall into this category. Taking some extreme examples:

- For a Markov chain, $\max_{c\in\mathcal{C}} |c| = 2$ regardless of M, but $D = 2M - 1$
- For a fully connected graph, $\max_{c\in\mathcal{C}} |c| = M$, but $D = 1$.

Next, note that the matrices $\boldsymbol{X}_c\boldsymbol{X}_c^\dagger$ do not change with the iteration number. Assuming $N \geq \max_{c\in\mathcal{C}} |c|$, it takes a total time of

$$\mathcal{O}\left(ND\left(\max_{c\in\mathcal{C}} |c|\right)^2\right) = \mathcal{O}\left(MN\left(\max_{c\in\mathcal{C}} |c|\right)\right)$$

to compute the inverses of these matrices, but this is a one-time cost. The simplification above is due to the assumption made previously that $D \max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(M\right)$.

In each iteration, the complexity is governed by the cost of computing $\boldsymbol{X}_c(\boldsymbol{Y}_c^k)^\dagger$ and the cost of computing $\boldsymbol{Y}_c^k$ itself, and it is simple to verify that both costs are $N \max_{c\in\mathcal{C}} |c|$ (per clique, separator). Thus, the per-iteration complexity of the GEM-LS algorithm is $\mathcal{O}\left(MN\right)$ and the overall average case complexity is given by $\mathcal{O}\left(MN \max\left(\mathbb{E}\left[k_{\text{opt}}\right], \max_{c\in\mathcal{C}} |c|\right)\right)$, where the second term arises due to the one-time cost. Note that $\max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(M\right)$ in the worst case, so all that is needed is to obtain the scaling of the average number of iterations with M.

For the large N, large noise case discussed in Section 4.4.1, k_{opt} is purely related to optimal shrinkage and the following can be shown using (4.30) (ignoring rounding)

$$\mathbb{E}\left[k_{\text{opt}}\right] = \mathbb{E}\left[\frac{\log(1 - \kappa_{\text{opt}})}{\log D - \log(D + 1)}\right]$$

$$
\begin{aligned}
&= \frac{\mathbb{E}\left[\log(1-\kappa_{\text{opt}})\right]}{\log D - \log(D+1)} \\
&\leq \frac{\log(1-\mathbb{E}\left[\kappa_{\text{opt}}\right])}{\log D - \log(D+1)}\,, \qquad (4.36a)
\end{aligned}
$$

where (4.36a) is a consequence of Jensen's inequality as $\log(1-x)$ is concave.

Applying (4.31b) to the above,

$$
\mathbb{E}\left[k_{\text{opt}}\right] \leq \frac{\log\left(1-\dfrac{\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2}{\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2+\beta\text{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\}}\right)}{\log D - \log(D+1)}\,,,
$$

which upon simplification yields

$$
\begin{aligned}
\frac{\log\left(1-\dfrac{\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2}{\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2+\beta\text{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\}}\right)}{\log D - \log(D+1)}
&= \frac{\log\left(\dfrac{\beta\text{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\}}{\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2+\beta\text{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\}}\right)}{\log\left(\dfrac{D}{D+1}\right)} \\
&= \frac{\log\left(1+\dfrac{\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2}{\beta\text{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\}}\right)}{\log\left(1+\dfrac{1}{D}\right)}\,,
\end{aligned}
$$

and therefore

$$
\mathbb{E}\left[k_{\text{opt}}\right] \leq \frac{\log\left(1+\dfrac{\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2}{\beta\text{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\}}\right)}{\log\left(1+\dfrac{1}{D}\right)}\,. \qquad (4.36b)
$$

Now, assume that $\left|\boldsymbol{h}_0-\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}\right|^2$ is $\mathcal{O}\left(M\right)$, i.e., it increases at most linearly with M, that $\text{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\} = \mathcal{O}\left(M\right)$. Recall that it has also been assumed that $D = \mathcal{O}\left(M\right)$ in the worst case. These assumptions are very mild and will almost always be true in practice. Then,

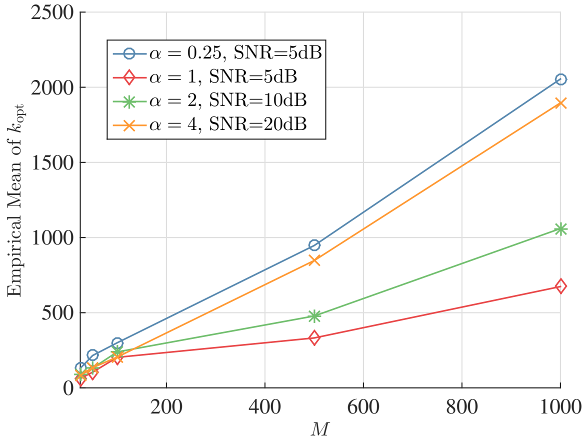


Figure 4-11: Scaling of $\mathbb{E}\left[k_{\text{opt}}\right]$ with M for different values of α, SNR. For this plot, the input structure is diagonal. The number of iterations is nearly linear in M for fixed α and SNR.

there exist some constants A, B (independent of M) so that

$$\mathbb{E}\left[k_{\text{opt}}\right] \leq \frac{\log\left(1+A\right)}{\log\left(1+\dfrac{B}{M}\right)}. \tag{4.37}$$

Finally,

$$\begin{aligned}\lim_{M\to\infty}\left(\frac{1}{M}\right)\frac{\log(1+A)}{\log\left(1+\dfrac{B}{M}\right)} &= \lim_{M\to\infty}\frac{\log(1+A)}{M\left(\dfrac{B}{M}+\dfrac{B^2}{M^2}+\dots\right)}\\ &= \lim_{M\to\infty}\frac{\log(1+A)}{\left(B+\dfrac{B^2}{M}+\dots\right)}\\ &= \frac{\log(1+A)}{B},\end{aligned}$$

which is a constant.

Combining this with (4.37), we obtain $\mathbb{E}\left[k_{\text{opt}}\right] = \mathcal{O}\left(M\right)$, and the overall average complexity of the GEM-LS algorithm in this regime is $\mathcal{O}\left(M^2 N\right)$. Thus, the GEM-LS algorithm is no more complex than the LS algorithm.

In the general case, no closed form expression for k_{opt} has been determined, so a rigorous proof of complexity can not be provided—at least not in the sense of complexity of the optimal solution.[8] However, it is fairly easy to verify that for a fixed α and SNR, the scaling of $\mathbb{E}\left[k_{\mathrm{opt}}\right]$ with M is nearly linear, as demonstrated in Figure 4-11 for the case of a system with diagonal input inverse covariance matrix (i.e., all independent nodes). This scaling property can be verified in a similar manner for input structures where D scales linearly with M—the same assumption that was made earlier in this section for the large N analysis.

Therefore, it can be concluded that the computational complexity of GEM-LS is no worse than that of the conventional LS algorithm for a purely serial implementation. As mentioned in Section 4.2, the complexity can be reduced by a factor of D by parallelizing the operations over every clique and separator.

4.4.5 Simulated Performance

A variety of analytical statements have been made about the GEM-LS algorithm in Sections 4.3 and 4.4. In this section, the performance of the GEM-LS algorithm is compared and contrasted to that of several other algorithms designed to operate in a sample-limited, high SNR regime.

Simulation Parameters

For all the simulations considered, the length of the vector being estimated is $M = 64$. In each simulation trial, the true parameter is chosen uniformly at random from the surface of the unit sphere in M dimensions [108]. Then, inputs and outputs are generated for different values of N ranging from $M/4$ to $4M$ and for different values of SNR ranging from 5 dB to 20 dB. Results are presented for the following two cases:

1. Each input is a realization of a white process.

2. Each input is a realization of a Markov chain.

[8] It is possible to show that for every $\delta > 0$, there exists a c independent of M, so that if $k > cM$ then $\left|\hat{\boldsymbol{h}}_{\mathrm{ls}} - \hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}}\right|^2 < \delta$. In other words, as a theoretical point, one may show that the complexity of convergence of the GEM-LS algorithm is also $\mathcal{O}\left(M^2 N\right)$. However, as a practical matter one would not be likely to run the algorithm until it gets close to LS, because it has been shown that there is a better stopping point in terms of estimation error. Thus, the proof of this convergence result has not been included, as it is not too relevant in practice.

The metric of performance is mean-squared estimation error (in dB) in this general simulation. In Section 4.5, the GEM-LS algorithm is applied to some practical problems and in those cases, metrics appropriate to the specific application will be considered. The results are plotted as a function of the number of realizations N at several different SNRs.

Algorithms Considered

The algorithms for which results are presented are:

1. The conventional LS algorithm. When $N \geq M$, this is unique. For $N < M$, the LS solution is determined by the algorithm used to compute the pseudo-inverse. For the purposes of the simulation, the default behavior of Matlab's \ operator is used, which produces the solution with the fewest non-zero components.

2. The oracle shrinkage solution of (4.29), as a lower bound on all possible shrinkage estimators on the line joining $\hat{\boldsymbol{h}}^0_{\mathrm{gem-ls}}$ and $\hat{\boldsymbol{h}}_{\mathrm{ls}}$.

3. The diagonally loaded LS algorithm of (2.15). The diagonal loading factor is chosen as $\delta_2 = 10^{-\mathrm{SNR}_{\mathrm{dB}}/10}$. This biases the diagonal loading to be larger if the noise level is higher, in which case less reliance should be placed on the sample data, which appears a reasonable choice. Note that, while some insights on choosing optimal diagonal loading have been provided by Pajovic [129], no closed form method appears to exist for accurately choosing the diagonal loading factor for an LS algorithm.

4. The LASSO implementation of the SPGL1 (Spectral Projection Gradient for ℓ_1-minimization) solver [16], [17], which solves the sparse LS problem described in Section 2.1.4. Rather than the penalized form of the LASSO regression of (2.16), the SPGL1 solver poses the LASSO problem in its constraint form, given by

$$\hat{\boldsymbol{h}}_{\mathrm{lasso}} = \arg\min_{\boldsymbol{h}\in\mathbb{C}^M} \sum_{n=1}^{N} \left|y(n) - \boldsymbol{h}^{\dagger}\boldsymbol{x}(n)\right|^2, \text{ subject to } \|\boldsymbol{h}\|_1 \leq \tau_1 \,. \tag{4.38}$$

For the purpose of the simulation, $\boldsymbol{h}_0$ is uniformly distributed on the unit M-sphere, and it can be shown that its expected ℓ_1 norm is given by

$$\mathbb{E}\left[\|\boldsymbol{h}_0\|_1\right] = \frac{\pi}{4}\sqrt{M}$$

Thus, the τ value for the LASSO problem of (4.38) is set to be a small factor above this, i.e., $\tau = 1.05\pi\sqrt{M}/4$, where the 1.05 is to allow a little buffer space.

5. GEM-LS with a starting point of $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}} = \mathbf{0}$ using a look-up table as described in Section 4.4.3 to determine the stopping iteration. A separate look-up table is used for the different models.

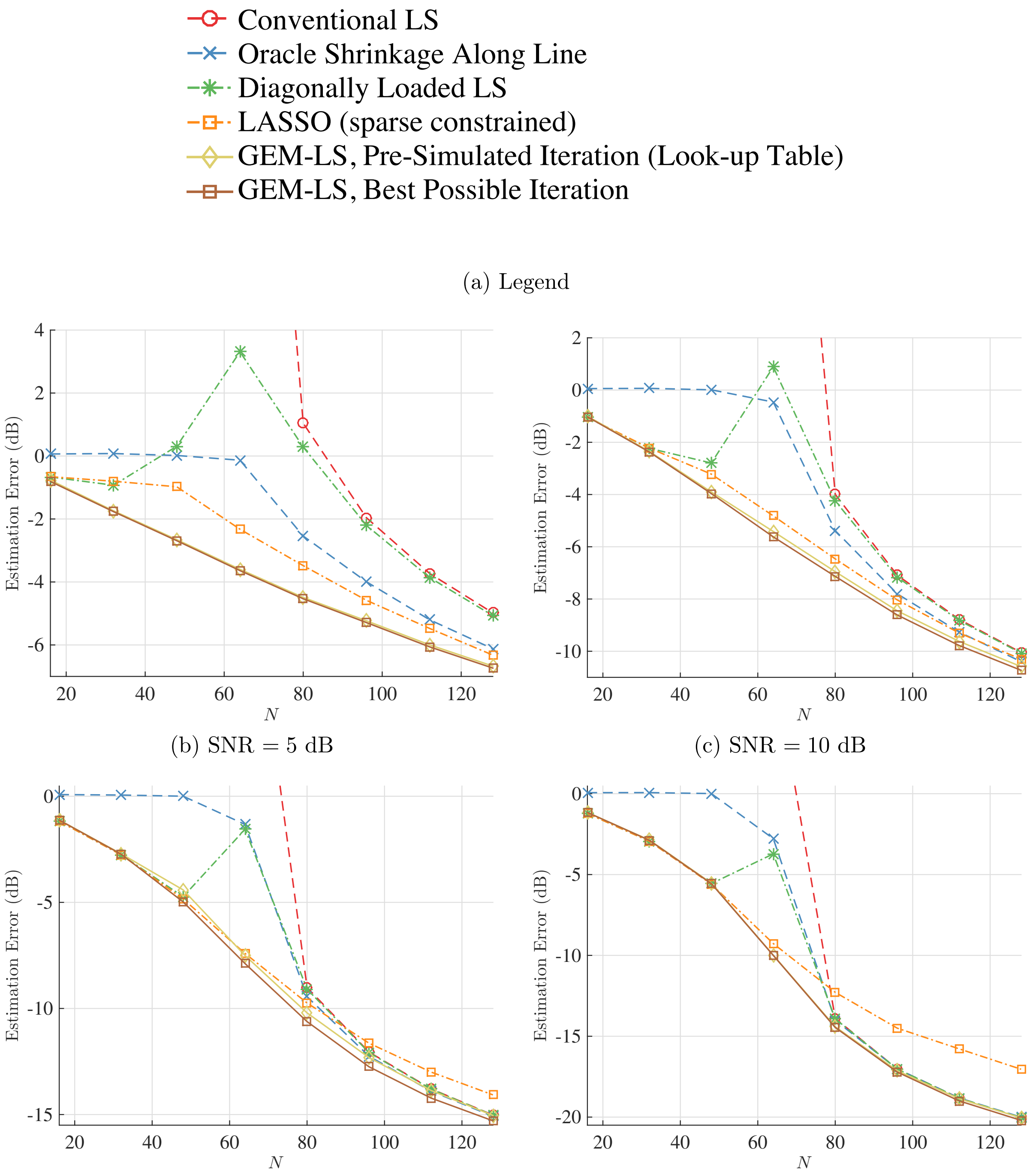
6. For reference, the average of the best error obtained by GEM-LS on any iteration (an "oracle" GEM-LS estimator) is also plotted.

Results

The results of the simulation of GEM-LS and the other methods described above when the input process is white are in Figure 4-12; and Figure 4-13 contains the results when the input $\boldsymbol{x}$ decomposes according to a Markov chain. The results for the two models are very similar.

At the outset, it should be noted that the performance loss from using the pre-simulated value of the mean of the optimum iteration, as discussed in Section 4.4.3, is neglible when compared to the improvement that GEM-LS provides over using the other methods in the comparison. This is as expected from the results of Figure 4-9b.

It is encouraging to note that the GEM-LS algorithm is consistently one of the best of the methods in terms of estimation error across the entire range of SNRs and values of N being considered. First, it considerably outperforms the oracle shrinkage solution along the line joining $\hat{\boldsymbol{h}}^0_{\text{gem}-\text{ls}}$ and $\hat{\boldsymbol{h}}_{\text{ls}}$, indicating that the data driven curve along which it allows for shrinkage has the property of being closer to the true parameter vector on average than the straight line. It should be noted, however, that for $N < M$, the oracle shrinkage performs poorly because the conventional LS solution $\hat{\boldsymbol{h}}_{\text{ls}}$ is not a good solution. A different choice of point on the LS line may lead to better results, but GEM-LS appears to choose a meaningful point regardless.



(a) Legend

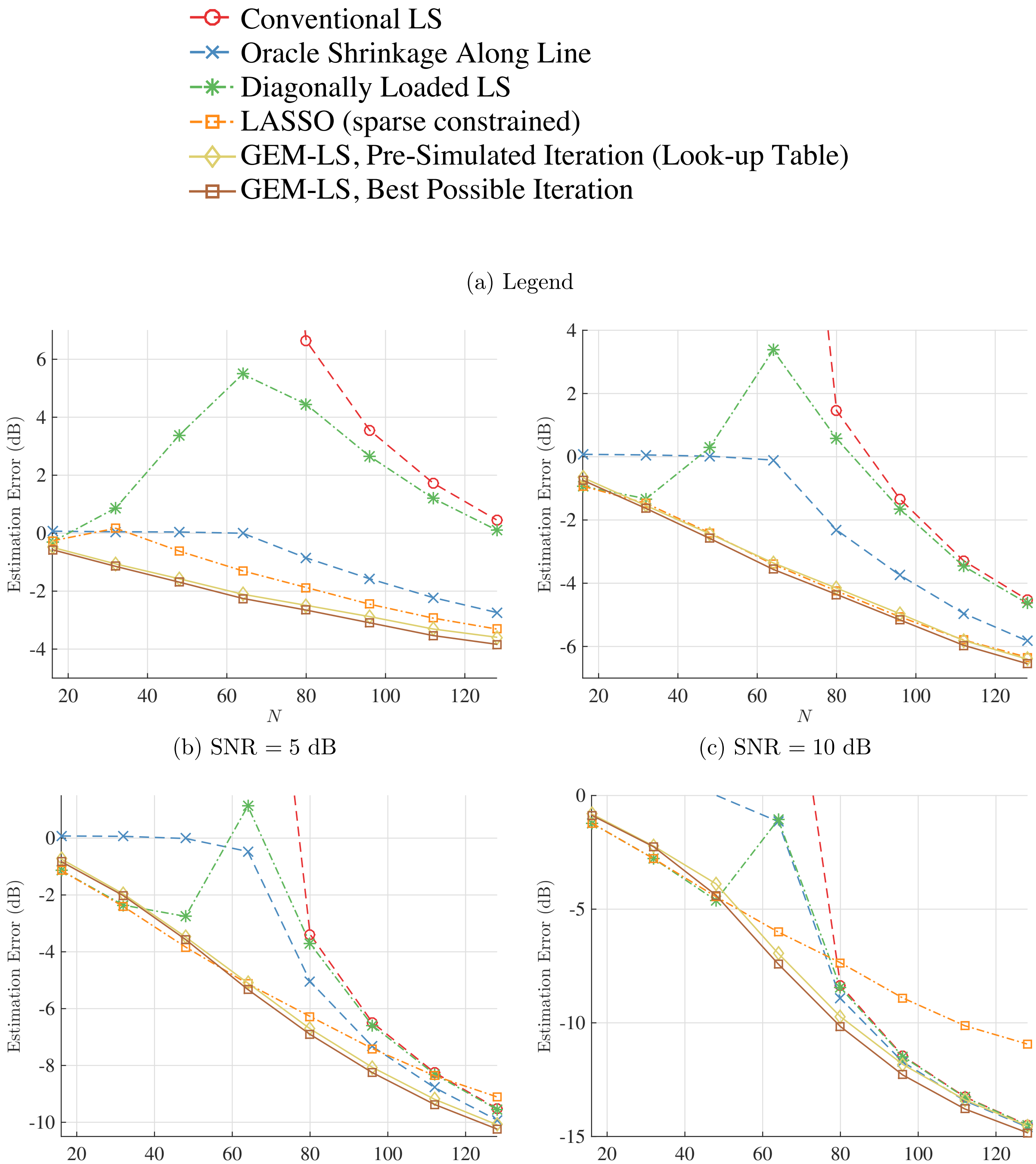
(b) SNR = 5 dB

(c) SNR = 10 dB

(d) SNR = 15 dB

(e) SNR = 20 dB

Figure 4-12: Comparison of GEM-LS to other system identification algorithms designed for the low-sample high-noise regime, including optimal linear shrinkage (4.29), diagonal loading (2.15) and the constraint form of LASSO (4.38). The performance of conventional LS is included for reference, and so is the performance of GEM-LS when the optimal stopping iteration in any given realization is given by an oracle. The input graph in this case is a graph with M independent nodes.



(a) Legend

(b) SNR = 5 dB

(c) SNR = 10 dB

(d) SNR = 15 dB

(e) SNR = 20 dB

Figure 4-13: Comparison of GEM-LS to other system identification algorithms designed for the low-sample high-noise regime, including optimal linear shrinkage (4.29), diagonal loading (2.15) and the constraint form of LASSO (4.38). The performance of conventional LS is included for reference, and so is the performance of GEM-LS when the optimal stopping iteration in any given realization is given by an oracle. The input graph in this case is a Markov chain, and in particular, the inputs are independent realizations of a length M AR-1 process $x_1, x_2, \ldots, x_M$.

More importantly, comparing it to the other practical methods—namely, ℓ_1 and ℓ_2 regularization—GEM-LS appears to be very accurate. For low SNRs, LASSO comes close and for high SNRs, diagonal loading can come close; but even when they do, GEM-LS improves upon them. Note the performance degradation of diagonally loaded LS around $N = M$. This degradation can be predicted by random matrix theory [129], [130].

4.5 Applications

In this section, the performance of GEM-LS in practical applications is considered. This will provide insight into quality of predictions made by GEM-LS with regard to practical metrics and when the chosen models have inaccuracies, as they always do in practice.

The two applications considered are fractionally-spaced channel identification (introduced in Section 3.3.1) and initializing acoustic echo cancellation filters (introduced in Section 3.2.1).

4.5.1 Channel Identification in Communication Systems

It was shown in Section 3.3.1 that for fractionally spaced channel identification, the input signal has a block diagonal inverse covariance matrix, i.e., it is characterized by a graph with M/T cliques of size T, where M is the length of the channel being identified and T the number of samples per transmitted symbol (the fractional spacing of the channel, given by $T = f_s/f_{\text{sym}}$, the sampling rate divided by the signalling rate). In this section, the GEM-LS algorithm is applied with this model to identify a channel generated using a realistic simulator of the underwater acoustic communication channel.

Channel Generator and Experiment Parameters

The data used is obtained from the simulation of acoustic modeled data from a 15 meter deep, 300 meter long underwater acoustic communication channel. The channel is assumed to be isovelocity, with a wind speed of 3 m/s. The signal is modulated onto a Gaussian pulse with 6 kHz bandwidth and transmitted on a carrier whose frequency is 15 kHz. The ambient noise is assumed to be Gaussian distributed.

The total channel length is assumed to be about 10.5 ms, which corresponds to about 62.5 symbol times. The equivalent baseband channel is assumed to have $T = 2$ samples per symbol, so the number of taps being identified is $M = 125$.

Channels are simulated from a fixed transmitted to each element of a fixed 50-element uniform vertical array of receivers separated by 10 cm, where the top element is 8.5 m below the water surface. Each of the 50 resulting channels are independently identified. The results presented are averaged across all 50 channels.

In the equivalent discrete time system, a delay line structure is used to produce the blocks $\boldsymbol{x}(n)$ that are transmitted across the channel. Thus, if the length of the pilot sequence is N_{pilot} symbols, then the number of available realizations of the input vector (i.e., the number of transmitted pilot blocks) is $N = N_{\text{pilot}} - M/T$. The goal is to identify the channel using a very short pilot sequence and at the lowest possible SNR—in effect, to push the limits of channel identification with limited information.

The results are presented as a function of the number of realizations N for different values of SNR. The metric of performance is the output prediction error, given by

$$E_{\text{pred}} = \frac{1}{N_{\text{pred}}} \sum_{n=1}^{N_{\text{pred}}} \left| y(n) - \hat{\boldsymbol{h}}^{\dagger} \boldsymbol{x}(n) \right|^2 , \tag{4.39}$$

where $\hat{\boldsymbol{h}}$ is the estimated channel, and the prediction error is computed using N_{pred} data samples.

Algorithms Considered

The following algorithms are considered for this problem

1. Conventional LS, where, as in Section 4.4.5, the default Matlab behavior of the \-operator is retained when $N < M$.

2. Diagonally loaded LS, with $\delta_2 = 10^{-\text{SNR}_{\text{dB}}/10}$.

3. A sparse-constrained LS algorithm. While it has been proposed that the underwater communication channel has sparse properties (e.g., [9], [18], [30], [161]) the precise sparsity of the channel under consideration is not known a-priori. Thus, rather than

the LASSO problem posed in (4.38), the sparse constrained solution is posed using the Basis-Pursuit De-Noise problem as

$$\hat{\boldsymbol{h}}_{\text{bpdn}} = \underset{h \in \mathbb{C}^M}{\arg\min} \left\|\boldsymbol{h}\right\|_1, \text{ subject to } \left|\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{h} - \hat{\boldsymbol{r}}_{\boldsymbol{xy}}(N)\right| \leq 1.05\sqrt{\sigma^2 \text{Tr}\left\{\boldsymbol{R}_{\boldsymbol{x}}\right\}/N}, \tag{4.40}$$

where the constraint is chosen as the mean value of $\left|\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{h}_0 - \hat{\boldsymbol{r}}_{\boldsymbol{xy}}(N)\right|$, with a factor of 1.05 to add a little buffer space.

LASSO is replaced by BPDN because that is the best way to utilize sparsity in this application. The precise amount of sparsity is unknown, but as the channel is understood to be sparse, it is reasonable to look for the "most sparse" solution for which the remainder is as close to the noise variance as possible. The noise variance is scaled for consistency with the LHS of the constraint.

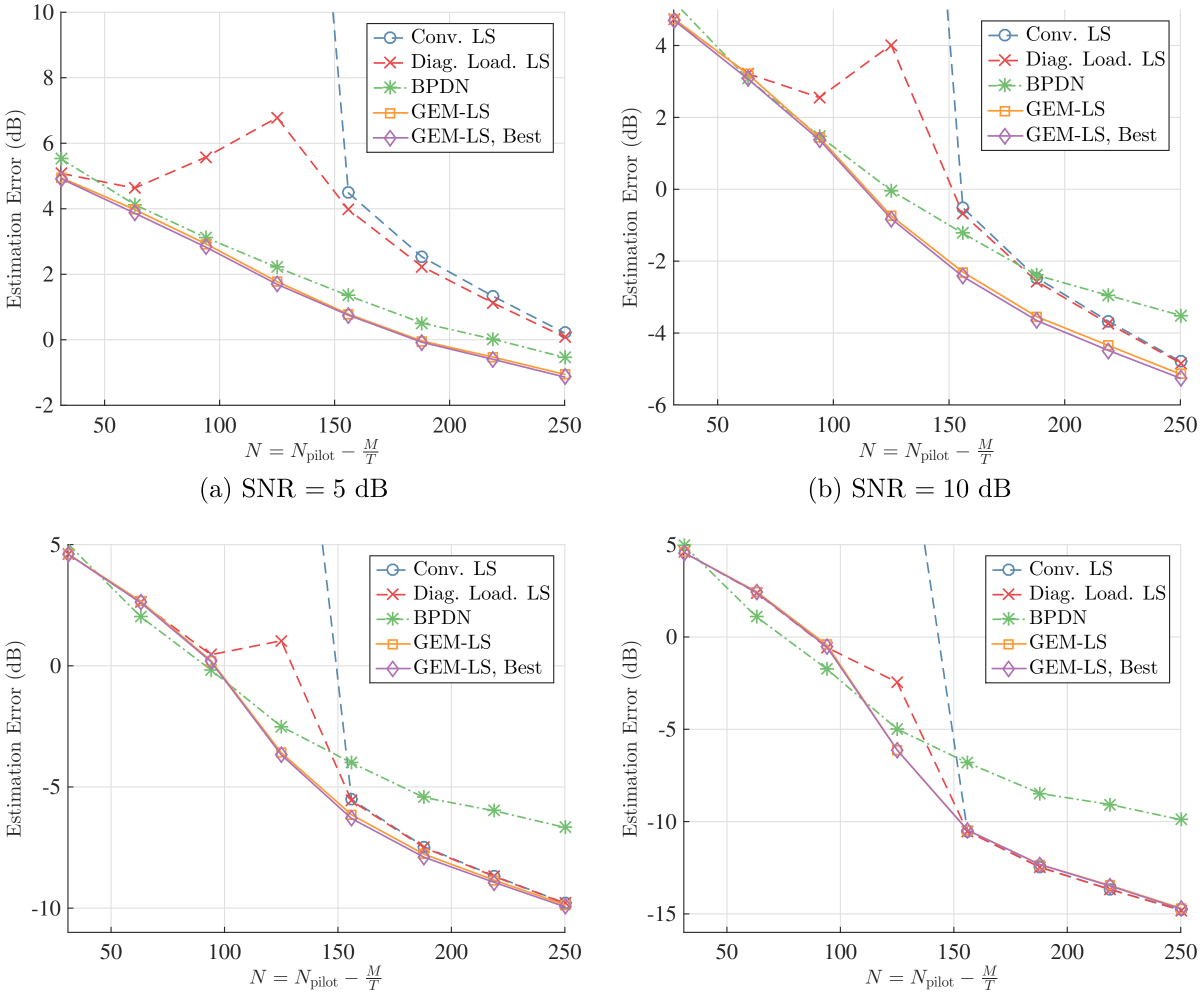
4. GEM-LS using the pre-simulated average optimal iteration. It is worth noting that there are two differences between the practical channel identification data and the theoretical framework in which GEM-LS has been derived:

    1. It cannot be assumed that the input realizations are independent, as they are successive outputs of a fractionally spaced delay line.
    2. While the inputs are realizations of a random variable $\boldsymbol{x}$ whose inverse covariance matrix is block-diagonal, the exact inverse covariance matrix is unknown, because it depends upon the sampling offset of the system. Although GEM-LS does not require knowledge of the exact inverse covariance matrix, the choice of the stopping iteration may be suboptimal as a result of this.

    For the data used, in the off-line simulation to pre-compute the iteration, the input inverse covariance matrix is a block diagonal matrix with blocks of size 2, where each block is given by

$$\begin{bmatrix} 4/3 & -2/3 \\ -2/3 & 4/3 \end{bmatrix}.$$

The inverse covariance matrix of the data is also block diagonal where each block



(a) SNR = 5 dB

(b) SNR = 10 dB

(c) SNR = 15 dB

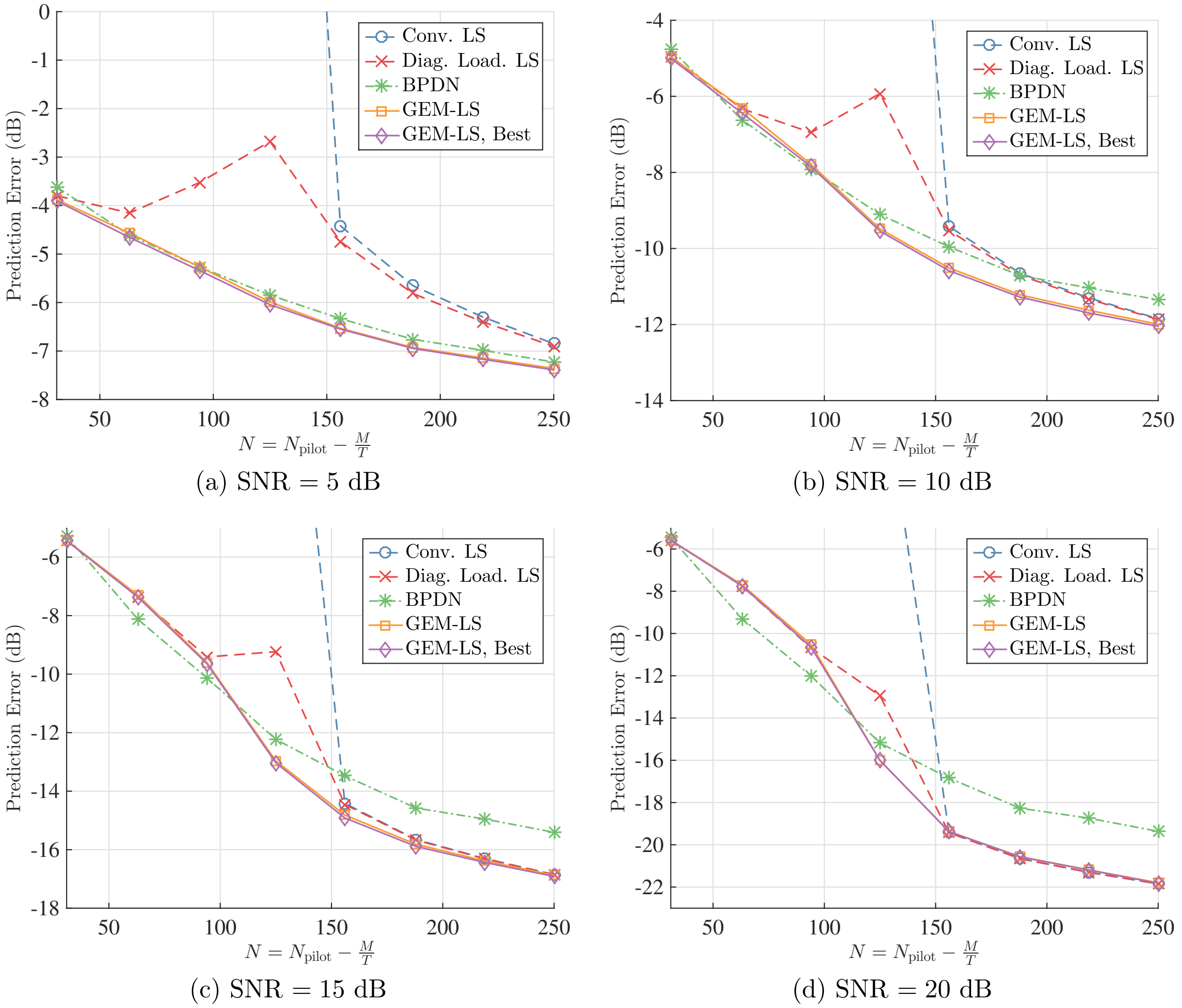
(d) SNR = 20 dB

Figure 4-14: Estimation error of GEM-LS, Conventional LS, BPDN and Diagonally Loaded LS for identification of a fractionally spaced wireless underwater communication channel.

is of size 2, but each block is now given by

$$\begin{bmatrix} 17.84 & -4.85 \\ -4.85 & 17.84 \end{bmatrix}.$$

So, while the structure is retained, the exact input signal statistics are different from those assumed to pre-compute the iteration.



(a) SNR = 5 dB

(b) SNR = 10 dB

(c) SNR = 15 dB

(d) SNR = 20 dB

Figure 4-15: Prediction error of GEM-LS, Conventional LS, BPDN and Diagonally Loaded LS for identification of a fractionally spaced wireless underwater communication channel.

Results

The prediction error performance of the various algorithms at different SNRs is shown in Figure 4-15, and, for reference, the estimation error performance is shown in Figure 4-14.

The GEM-LS algorithm with the genie iteration and with the pre-simulated best iteration have very similar performance, in spite of the pre-simulated iteration being computed using a different model. Thus, there appears to be sufficient robustness in the method of pre-computing the iteration to use it in practice.

Additionally, the GEM-LS algorithm is again consistently among the best performing algorithms. The performance of BPDN and GEM-LS is very close for small amounts of data, but BPDN loses its advantage as the amount of data grows. It may be possible[9] to recover the lost performance of BPDN by choosing a different penalty scaling, but that goes to show that regularized methods are sensitive to the amount of penalization. Similarly, diagonally loaded LS suffers around $M = N$ [129].

In other words, it is possible for regularized LS methods to match the performance of GEM-LS for a suitable choice of regularization parameter, but choosing the regularization parameter is not trivial, and moreover a particular choice leads to good performance only in particular regimes. On the other hand, throughout the entire regime, GEM-LS, using the practically chosen stopping iteration, performs very well.

Thus the practical results continue to imply that problem separation using input data structure can be a powerful tool in estimation.

4.5.2 Initializing Acoustic Echo Cancellation Filters

A second application of GEM-LS that is considered here is the problem of initializing acoustic echo cancellation filters, introduced in 3.2.1. As explained in that section, the structure of the signal arises because the effective calibration tone is white due to passage through a prewhitening filter. However, the whitening process is not perfect, so the calibration tone is only approximately white.

[9] The results shown for BPDN are the best that were obtained for this dataset.

Data Parameters

The data was obtained by a simulated room impulse response generated using the room impulse response generator described by Habets, et al. [73], which is simply an efficient implementation of the classical image method for simulating room acoustics [5]. The origin being chosen as one bottom corner of the room, the opposite top corner of the simulated cuboidal room was $(x, y, z) = (3.7, 2.74, 3.7)$ (all dimensions are in meters, y-dimension is vertical). The microphone was located at $(1.85, 0.6, 0.3)$ and the speaker at $(3.4, 0.4, 3.1)$ in the same coordinate system. The T_{60} time for the room was 0.25 seconds.

The sampling frequency for the system was 16 kHz (typical for modern infotainment systems). The first 50 ms of the response were estimated for a filter length of 800 taps (beyond this, the received signal level is less than the lowest chosen noise level during calibration). The noise is generated by a point source that outputs white noise from a randomly chosen (in each realization) point in the room at any desired level.

The measure of performance in these systems is Echo Return Loss Enhancement (ERLE) defined in (3.2), measured as a function of the noise level during calibration (termed Calibration ENR, or Echo-to-Noise Ratio). It is assumed that $N = 1600$ blocks are available for calibration. Note that, as with the practical channel identification example of Section 4.5.1, the blocks are generated using a tapped delay line, so the total calibration signal length assumed was about 150 ms. The goal in this case is to achieve the best possible ERLE at very low ENRs.

Algorithms and Results

Typically, a simple pre-whitened matched filter would be used to initialize the filters due to the complexity of using LS type approaches. There is also no sparsity to speak of if only a single microphone is used (if an array of microphones is used, then it is possible to use a sparse basis to represent signal structure [187]). There appears to be little easily exploitable structure in this problem, other than the input structure.

The plot of the ERLE as a function of calibration ENR is shown in Figure 4-16 along with the corresponding ERLEs of the matched filter and the LS algorithms. In the very

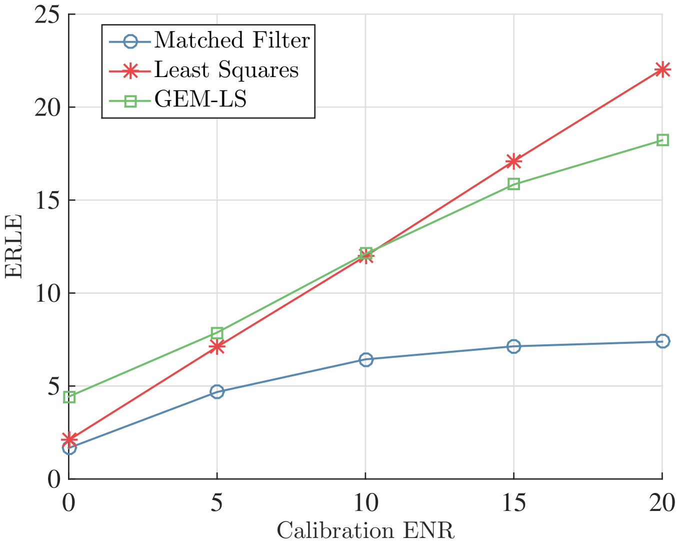


Figure 4-16: ERLE (dB) vs calibration ENR (dB) for the matched filter, LS and GEM-LS algorithms.

low ENR regime (ENR < 10 dB), the GEM-LS algorithm is quite helpful. Even at 0 dB ENR, GEM-LS produces nearly 5 dB ERLE. This ENR regime is prevalent in applications such as automobile voice-controlled infotainment systems, where the amount of noise in the environment is very large. At higher ENRs, LS appears better than GEM-LS, and both considerably outperform the typically-used matched filter. Note that the reason for the dominance of LS in this regime is that insufficient iterations of GEM-LS were run—as has been shown, GEM-LS can always be at least as good as LS.

Typically, the complexity of LS is considered too large for the estimation of filters of dimension 800, as used in this example. As the complexity of GEM-LS is the same as that of LS, it may not be clear how, at a practical level, GEM-LS could be used in this application. However, note that GEM-LS is highly parallelizable; and particularly when the input signal is characterized by the simple graph with M independent nodes, the operation for each clique is a scalar operation, which can be performed very efficiently in parallel. Moreover, in the low ENR regime of interest, the GEM-LS algorithm requires very few iterations for the best possible performance, which leads to a further complexity reduction.

4.6 Some Comments and Looking Ahead

In this chapter, the core algorithm that allows for the exploitation of input structure has been introduced and extensively analyzed. The benefits and drawbacks of the algorithm have been introduced, and various unanswered questions have been noted. It has been shown using analysis and simulation that the GEM-LS algorithm can be very effective.

The algorithm has been applied to the problems of acoustic echo cancellation (with the graph described in Section 3.2) and channel identification (with the graph described in Section 3.3) and it has been shown that the gains indicated by the analysis are realizable in practice.

The GEM-LS algorithm uses input structure to improve performance of linear system identification, but it is also possible to use the same kind of input structure to reduce the complexity of the problem at the expense of performance. In the next chapter, this problem is considered.

The relaxed approximate graph structured-LS (RAGS-LS) algorithm introduced in the next chapter is a relaxation of the GEM-LS algorithm that does not require iterations, and thus, solves a set of smaller problems just one time; thereby reducing the complexity. However, by giving up the iterations, the algorithm is forced to solve each of the smaller (clique/separator) LS problems in a regime where the "effective noise" seen by each problem is very large. Thus, it improves complexity, but does so at the expense of performance; except in a very low-SNR regimes.

A recursive variant of the RAGS-LS algorithm is also developed that is able to mitigate the large effective noise issue to some extent. The resulting RAGS-RLS algorithm is particularly suited to tracking time-varying systems.

Chapter 5

Non-Iterative Relaxation, Recursive Formulation and Tracking

The GEM-LS algorithm is a precise framework for improving performance of linear system identification using input graphical model structure. However, its complexity is on the same order as conventional LS—in other words, it can improve performance, but not complexity.

On the other hand, it is also possible to use such structure to reduce the complexity of the estimation problem, rather than to improve performance.[1] In this chapter, the relaxed approximate graph-structured LS algorithm (RAGS-LS) is introduced, which presents a method for doing so.

The derivation of RAGS-LS is presented without regard to the GEM-LS algorithm. However, it is subsequently shown that RAGS-LS may be viewed as a relaxation of GEM-LS, which explains the word "relaxed" in its title. As with GEM-LS, RAGS-LS reduces the data requirements by using the input structure to reduce the problem dimension. In some regimes, especially at low SNRs, RAGS-LS may also improve upon LS in terms of performance. However, in other regimes, the performance of RAGS-LS can be inferior to that of LS, and the performance loss can be quite significant.

The RAGS-LS algorithm does not consistently apply constraints across the available data, which is why it is an "approximate" algorithm. This leads to the performance loss

[1] Without parallelization, no methods have thus far been developed that can simultaneously improve both in every regime.

described in some regimes. The inconsistency in applying constraints leads the RAGS-LS algorithm to treat much of the signal portion of the data (the portion of the received signal that is due to the passing of the input signal through the black box system) as additive noise. This creates a situation where the effective SNR from the point of view of each clique or separator LS estimator is much lower than the true SNR.

However, it will be shown in Section 5.4 that the RAGS-LS algorithm can easily be modified into a recursive framework termed the RAGS-RLS algorithm. Due to the form of the recursive implementation, RAGS-RLS is able to mitigate the reduction in effective SNR to some extent. Additionally, with the addition of exponential windowing, the recursive formulation is particularly suited to tracking time-varying systems. It will be shown that RAGS-RLS performs comparably to the RLS algorithm that is conventionally used for tracking, and can be considerably less computationally complex.

Finally, the RAGS-RLS algorithm is applied to two tracking problems: tracking acoustic echo cancellation filter coefficients in a room with a changing environment, and tracking adaptive equalizer coefficients in underwater wireless communication systems. In both the applications, it will be shown that the performance of RAGS-RLS is comparable to that of state-of-the-art algorithms with a reduced computational cost.

5.1 The Relaxed Approximate Graph-Structured Least Squares Algorithm

As discussed in Section 2.1.2, $\hat{\boldsymbol{R}}_{\boldsymbol{x}}^{-1}(N)$ of (2.9a) is the ML estimate of the inverse covariance matrix $\boldsymbol{J}_{\boldsymbol{x}}$ when the distribution $p_{\boldsymbol{x}}(\boldsymbol{x})$ from which $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(N)$ are drawn is an unstructured Gaussian distribution. When graphical model structure is present, therefore, one way to simplify the estimation problem is to replace the unstructured ML estimate of the inverse covariance matrix with the ML estimate of $\boldsymbol{J}_{\boldsymbol{x}}$ subject to the fact that $\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(N)$ are realizations of $\boldsymbol{x}$, a random variable that is characterized by a decomposable graphical model.

This problem was extensively investigated by Wiesel et al. [181], in which it was shown that, when $\boldsymbol{\mathsf{x}}$ is characterized by a decomposable graph $\mathcal{G}$, the ML estimate of the inverse covariance matrix is given by

$$\hat{\boldsymbol{J}}_{\boldsymbol{\mathsf{x}}_{\mathcal{G}}}(N) = \sum_{c \in \mathcal{C}} \left[\left(\frac{1}{N} \boldsymbol{X}_c \boldsymbol{X}_c \right)^{-1} \right]_{c \times c}^{M \times M} - \sum_{s \in \mathcal{S}} \left[\left(\frac{1}{N} \boldsymbol{X}_s \boldsymbol{X}_s \right)^{-1} \right]_{s \times s}^{M \times M} . \tag{5.1}$$

This is the same matrix that appeared in the analysis of the GEM-LS algorithm in (4.13), but the interpretation as an ML estimator of a structured inverse covariance matrix was not introduced at that time. The quantities $\boldsymbol{X}_c$ and $\boldsymbol{X}_s$ are given by (cf. (4.11c))

$$\boldsymbol{X}_c = \boldsymbol{X}_{c,:},\ c \in \mathcal{C} , \tag{5.2a}$$

where,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}(1) & \boldsymbol{x}(2) & \cdots & \boldsymbol{x}(N) \end{bmatrix} . \tag{5.2b}$$

The relaxed approximate graph-structured least squares estimate is simply obtained by replacing the unconstrained estimate $\hat{\boldsymbol{R}}_{\boldsymbol{\mathsf{x}}}^{-1}(N)$ in the LS estimate of (2.8) with the structured estimate of (5.1), giving

$$\begin{aligned}
\hat{\boldsymbol{h}}_{\text{rags}-\text{ls}} &= \hat{\boldsymbol{J}}_{\boldsymbol{\mathsf{x}}_{\mathcal{G}}}(N) \hat{\boldsymbol{r}}_{\boldsymbol{\mathsf{x}}\boldsymbol{y}}(N) && (5.3\text{a}) \\
&= \left(\sum_{c \in \mathcal{C}} \left[\left(\frac{1}{N} \boldsymbol{X}_c \boldsymbol{X}_c \right)^{-1} \right]_{c \times c}^{M \times M} - \sum_{s \in \mathcal{S}} \left[\left(\frac{1}{N} \boldsymbol{X}_s \boldsymbol{X}_s \right)^{-1} \right]_{s \times s}^{M \times M} \right) \hat{\boldsymbol{r}}_{\boldsymbol{\mathsf{x}}\boldsymbol{y}}(N) && (5.3\text{b}) \\
&= \sum_{c \in \mathcal{C}} \left[\hat{\boldsymbol{R}}_{\boldsymbol{\mathsf{x}}_c}^{-1}(N) \hat{\boldsymbol{r}}_{\boldsymbol{\mathsf{x}}_c \boldsymbol{y}}(N) \right]_{c \times 1}^{M \times 1} - \sum_{s \in \mathcal{S}} \left[\hat{\boldsymbol{R}}_{\boldsymbol{\mathsf{x}}_s}^{-1}(N) \hat{\boldsymbol{r}}_{\boldsymbol{\mathsf{x}}_s y}(N) \right]_{s \times 1}^{M \times 1} , && (5.3\text{c})
\end{aligned}$$

where

$$\hat{\boldsymbol{R}}_{\boldsymbol{\mathsf{x}}_c}(N) = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_c(n) \boldsymbol{x}_c^{\dagger}(n) \tag{5.4a}$$

$$\hat{\boldsymbol{r}}_{\boldsymbol{\mathsf{x}}_c \boldsymbol{y}}(N) = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_c(n) y^{*}(n) , \tag{5.4b}$$

and where $\boldsymbol{x}_c(n)$ is the subvector of $\boldsymbol{x}(n)$ formed by the elements corresponding to clique c.

The decomposed solution of (5.3c) is obtained because, when the matrix $\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}_c}\right]_{c\times c}^{M\times M}$ is multiplied by a vector, only the elements of the vector in positions corresponding to c remain since all the other elements are multiplied by 0.

It is emphasized that the RAGS-LS estimator (5.3c) only applies when $\boldsymbol{x}$ is characterized by a *decomposable* graphical model. This is a consequence of the fact that (5.1) is the ML estimator of the inverse covariance matrix when the underlying structure is decomposable. For non-decomposable models, there is some work on estimating the inverse covariance matrix ([115] and references therein), but the results of those works were not found to be readily applicable to linear system identification problems. Indeed, the question of linear system identification when the input is characterized by a non-decomposable graphical model has not been considered in this thesis, and remains an open problem.

5.2 Properties of RAGS-LS

Some useful properties of the RAGS-LS algorithm are discussed in this section.

5.2.1 Separability

Define, for any clique c,

$$\hat{\boldsymbol{h}}_{\text{rags-ls}_c} = \hat{\boldsymbol{R}}_{\boldsymbol{x}_c}(N)\hat{\boldsymbol{r}}_{\boldsymbol{x}_c\boldsymbol{y}}(N) \tag{5.5}$$

as the *local least squares* estimate of the clique parameter $\boldsymbol{h}_c$. Then the RAGS-LS estimate is the combination of the local least squares estimates.

Note that the local LS estimates are *not* the same as the LS estimates of the clique parameters $\boldsymbol{h}_c, \boldsymbol{h}_s$! The true LS estimates of $\boldsymbol{h}_c, \boldsymbol{h}_s$ would have to be computed using the values of the clique and separator outputs y_c, y_s, as explained in Section 4.1 (additionally, see Section 5.3.4), which are not available. In the RAGS-LS algorithm, the estimates for all the cliques and separators are simply computed using the overall output, $y(n)$, and those estimates are called the "local LS" estimates.

As the local LS solution for each clique and separator is computed without regard to the other cliques and separators, and the estimates are combined only after all the solutions are computed, this property is termed the *separability* of RAGS-LS.

Due to the separability property, the RAGS-LS algorithm is parallelizable, as the LS solutions to each clique and separator can be computed simultaneously. Note that while the GEM-LS algorithm is parallelizable in each iteration, the RAGS-LS algorithm is not iterative, which means that one set of parallel operations is sufficient to obtain an estimate.

5.2.2 Data Requirements

Like the GEM-LS algorithm, every matrix inversion operation in the RAGS-LS solution of (5.3c) is well defined provided $N \geq \max_{c\in\mathcal{C}} |c|$. Thus, for small clique sizes, the algorithm is far less likely to be sample deficient than the conventional LS solution, meaning that it is helpful in solving sample deficient linear system identification problems.

5.2.3 Computational Complexity

The computational complexity is again determined by the matrix inversion step. For the conventional LS algorithm, this has a complexity of $\mathcal{O}\left(M^{2.8}\right)$ [183].[2] The RAGS-LS has a complexity of

$$\mathcal{O}\left(\sum_{c\in\mathcal{C}} |c|^{2.8} + \sum_{s\in\mathcal{S}} |s|^{2.8}\right) = \mathcal{O}\left(D\left(\max_{c\in\mathcal{C}} |c|\right)^{2.8}\right) .$$

A speed-up of $\mathcal{O}\left(D\right)$ is possible with parallelization, giving a complexity of $\mathcal{O}\left(\left(\max_{c\in\mathcal{C}} |c|\right)^{2.8}\right)$ for a parallel implementation.

While precise statements about computational complexity improvements can only be made with reference to specific graphical models, roughly speaking, if $D = \mathcal{O}\left(M\right)$, then the complexity of RAGS-LS is less than that of conventional LS if $\max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(M^{\frac{2}{3}}\right)$. For relatively sparse graphs, the complexity gains can be quite significant. For instance, with $M = 50$ and with the input data characterized by a Markov Chain ($\max_{c\in\mathcal{C}} |c| = 2$) the complexity of RAGS-LS is about 1/10 that of conventional LS.

[2] Better algorithms are known but only for impractically enormous matrices

5.2.4 RAGS-LS as a Relaxation of GEM-LS

It has been noted that RAGS-LS is a relaxation of GEM-LS. In this section, it is explained why this is the case. The answer stems from the analysis of GEM-LS of Section 4.4. In particular, rearranging (4.25c), the following is obtained for the GEM-LS solution

$$\hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}} = (\boldsymbol{I} - \boldsymbol{\rho}_{\boldsymbol{x}}(N))^{k-1} \boldsymbol{\rho}_{\boldsymbol{x}}(N) \left(\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{0}_{\mathrm{gem-ls}} \right) . \tag{5.6}$$

Provided the mild conditions of (4.21) are met, the absolute value of every eigenvalue of $\boldsymbol{I} - \boldsymbol{\rho}_{\boldsymbol{x}}(N)$ is less than 1, which implies that $\left| \hat{\boldsymbol{h}}^{k}_{\mathrm{gem-ls}} - \hat{\boldsymbol{h}}^{k-1}_{\mathrm{gem-ls}} \right|$ is a decreasing function (and, in particular, decreases exponentially quickly). Thus, the largest step taken by the GEM-LS algorithm is in the first step.

Assuming that $\hat{\boldsymbol{h}}^{0}_{\mathrm{gem-ls}} = \boldsymbol{0}$, it is easily seen that

$$\begin{aligned}
\hat{\boldsymbol{h}}^{1}_{\mathrm{gem-ls}} &= \boldsymbol{\rho}_{\boldsymbol{x}}(N) \hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} \\
&= \frac{1}{D+1} \hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N) \left[\frac{1}{N} (\boldsymbol{X}\boldsymbol{X}^{\dagger}) \hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} \right] \\
&= \frac{1}{D+1} \hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N) \hat{\boldsymbol{r}}_{\boldsymbol{xy}}(N) \qquad (5.7\text{a}) \\
&= \frac{1}{D+1} \hat{\boldsymbol{h}}_{\mathrm{rags-ls}} , \qquad (5.7\text{b})
\end{aligned}$$

where (5.7a) is a consequence of the fact that $\hat{\boldsymbol{h}}^{\infty}_{\mathrm{gem-ls}} \in \mathcal{H}$, the set of all possible LS solutions, due to Theorem 4.3.1, and, by definition

$$\boldsymbol{h} \in \mathcal{H} \implies (\boldsymbol{X}\boldsymbol{X}^{\dagger})\boldsymbol{h} = \boldsymbol{X}\boldsymbol{Y}^{\dagger} .$$

Thus, the largest step of the GEM-LS algorithm is a fraction of the RAGS-LS solution. Viewed a different way, the RAGS-LS solution is obtained by dropping the iterations of the GEM-LS algorithm and simply taking a large step in the direction of the biggest step of the GEM-LS algorithm.

Geometrically speaking, GEM-LS algorithm attempts to improve the performance by setting up a data-driven curve by using the structure and performing shrinkage along that curve for a suitable choice of iteration (see Section 4.4.2). RAGS-LS attempts to reduce complexity by dropping the iterations, thereby needing only to solve relatively simple subproblems *one time*. The subproblems it solves takes it along the starting direction of the data-driven curve set up by GEM-LS although it takes a larger step along that direction than does GEM-LS in the first step.

5.3 Performance of RAGS-LS

While computationally very efficient, the RAGS-LS algorithm can suffer in terms of estimation error performance while estimating time-invariant systems. This is first shown analytically and then verified by simulation.

5.3.1 Analysis

In the regime where $N \sim M$, but where the amount of data is much larger than the size of the largest clique in the graph, $N \gg \max_{c\in\mathcal{C}} |c|$, the performance of the RAGS-LS algorithm is established in the following theorem.

Theorem 5.3.1. *In the regime where $N \not\gg M$, and assuming $N \gg \max_{c\in\mathcal{C}} |c|$, the estimation error of the RAGS-LS algorithm is given by*

$$\mathbb{E}\left[\left\|\boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\mathrm{rags-ls}}\right\|^2\right] = \frac{\sigma^2\, Tr\{\boldsymbol{J}_{\boldsymbol{x}}\}}{N} + \frac{\|\boldsymbol{h}_0\|^2 + \boldsymbol{h}_0^\dagger \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{h}_0\, Tr\{\boldsymbol{J}_{\boldsymbol{x}}\}}{N} \tag{5.8a}$$

$$= \frac{\left(\sigma^2 + \boldsymbol{h}_0^\dagger \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{h}_0\right)\, Tr\{\boldsymbol{J}_{\boldsymbol{x}}\} + \|\boldsymbol{h}_0\|^2}{N} \tag{5.8b}$$

Proof. For $N \gg \max_{c\in\mathcal{C}} |c|$, the convergence of $\hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)$ to $\boldsymbol{J}_{\boldsymbol{x}}$ is quite rapid assuming the algorithm has access to the correct graphical model $\mathcal{G}$ [181] (no attempt is made in this case to analyze the effect of an incorrect model). Thus, using $\hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N) \approx \boldsymbol{J}_{\boldsymbol{x}}$ in (5.3a),

$$\begin{aligned}\hat{\boldsymbol{h}}_{\mathrm{rags-ls}} &= \hat{\boldsymbol{J}}_{\boldsymbol{x}_{\mathcal{G}}}(N)\hat{\boldsymbol{r}}_{\boldsymbol{x}y}(N) \\ &\approx \boldsymbol{J}_{\boldsymbol{x}}\left[\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}(n)\left(\boldsymbol{h}_0^\dagger \boldsymbol{x}(n) + v(n)\right)^*\right]\end{aligned}$$

$$
\begin{aligned}
&= \boldsymbol{J}_{\boldsymbol{x}}\left[\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)\boldsymbol{x}^{\dagger}(n)\boldsymbol{h}_0 + \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)v^{*}(n)\right] \\
&= \boldsymbol{J}_{\boldsymbol{x}}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\boldsymbol{h}_0 + \frac{\boldsymbol{J}_{\boldsymbol{x}}}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)v^{*}(n)\,,
\end{aligned}
$$

for any given realization of the input.

Note that the assumption that the regime $N \not\gg M$ is implicit in the above simplifications. Otherwise, $\hat{\boldsymbol{r}}_{\boldsymbol{xy}}(N) \approx \boldsymbol{r}_{\boldsymbol{xy}}$ and the algorithm is perfect.

Define

$$
\begin{aligned}
\boldsymbol{\epsilon}_{\text{rags}-\text{ls}} &= \boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\text{rags}-\text{ls}} \\
&= \left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{x}}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\right)\boldsymbol{h}_0 - \frac{\boldsymbol{J}_{\boldsymbol{x}}}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)v^{*}(n)\,.
\end{aligned}
$$

Averaging this over all possible realizations of the input and noise gives

$$
\begin{aligned}
\mathbb{E}\left[\boldsymbol{\epsilon}^{\dagger}_{\text{rags}-\text{ls}}\boldsymbol{\epsilon}_{\text{rags}-\text{ls}}\right] &= \boldsymbol{h}_0^{\dagger}\underbrace{\mathbb{E}\left[\left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{x}}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\right)^{\dagger}\left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{x}}\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\right)\right]}_{\triangleq \boldsymbol{T}_1}\boldsymbol{h}_0 \\
&\quad + \underbrace{\mathbb{E}\left[\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)\boldsymbol{v}^{*}(n)\right)^{\dagger}\boldsymbol{J}_{\boldsymbol{x}}^{\dagger}\boldsymbol{J}_{\boldsymbol{x}}\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)\boldsymbol{v}^{*}(n)\right)\right]}_{\triangleq T_2} \\
&= \boldsymbol{h}^{\dagger}\boldsymbol{T}_1\boldsymbol{h} + T_2\,,
\end{aligned}
$$

where the cross terms vanish due to the independence of $\boldsymbol{v}(n)$ and $\boldsymbol{x}(m)$ for all m, n.

Beginning with the second term, as T_2 is a scalar,

$$
\begin{aligned}
T_2 &= \operatorname{Tr}\left\{T_2\right\} \\
&= \mathbb{E}\left[\operatorname{Tr}\left\{\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{\dagger}(n)\boldsymbol{v}(n)\right)\boldsymbol{J}_{\boldsymbol{x}}^{\dagger}\boldsymbol{J}_{\boldsymbol{x}}\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)\boldsymbol{v}^{*}(n)\right)\right\}\right] \\
&= \mathbb{E}\left[\operatorname{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}^{2}\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}(n)\boldsymbol{v}^{*}(n)\right)\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{\dagger}(n)\boldsymbol{v}(n)\right)\right\}\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \mathrm{Tr}\left\{ \frac{\boldsymbol{J}_{\boldsymbol{x}}^2}{N} \mathbb{E}\left[\sum_{n=1}^{N} \frac{1}{N} \boldsymbol{x}(n)\boldsymbol{x}^\dagger(n) \left|\boldsymbol{v}(n)\right|^2 \right] \right\} \\
&= \frac{\sigma^2}{N} \mathrm{Tr}\left\{ \boldsymbol{J}_{\boldsymbol{x}}^2 \mathbb{E}\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) \right] \right\} \\
&= \frac{\sigma^2 \mathrm{Tr}\left\{ \boldsymbol{J}_{\boldsymbol{x}} \right\}}{N} .
\end{aligned}
$$

We have repeatedly used above that the order of expectation and trace can be reversed. In the second step, we have used that trace is invariant under cyclic permutation and that $\boldsymbol{J}_{\boldsymbol{x}}$ is a Hermitian matrix. To go from the second to the third step, it is sufficient to note that $\boldsymbol{v}(i) \perp\!\!\!\perp \boldsymbol{v}(j)$ when $i \neq j$, and the next step follows as $\boldsymbol{v}(n) \perp\!\!\!\perp \boldsymbol{x}(n)$. Finally, $\mathbb{E}\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)\right] = \boldsymbol{R}_{\boldsymbol{x}} = \boldsymbol{J}_{\boldsymbol{x}}^{-1}$ using the known mean of a Wishart matrix [99].

For $\boldsymbol{T}_1$,

$$
\begin{aligned}
\boldsymbol{T}_1 &= \mathbb{E}\left[\left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{x}} \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) \right)^\dagger \left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{x}} \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) \right) \right] \\
&= \boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{x}} \mathbb{E}\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) \right] - \mathbb{E}\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}}^\dagger(N) \right] \boldsymbol{J}_{\boldsymbol{x}}^\dagger + \mathbb{E}\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}}^\dagger(N) \boldsymbol{J}_{\boldsymbol{x}}^\dagger \boldsymbol{J}_{\boldsymbol{x}} \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) \right] \\
&= -\boldsymbol{I} + \mathbb{E}\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}}^\dagger(N) \boldsymbol{J}_{\boldsymbol{x}}^\dagger \boldsymbol{J}_{\boldsymbol{x}} \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) \right] ,
\end{aligned}
$$

where each of the middle 2 terms of the second equation evaluates to $\boldsymbol{I}$.

Defining $\boldsymbol{Z} = \boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{X}$,

$$
\begin{aligned}
\boldsymbol{T}_1 &= -\boldsymbol{I} + \mathbb{E}\left[\hat{\boldsymbol{R}}_{\boldsymbol{x}}^\dagger(N) \boldsymbol{J}_{\boldsymbol{x}}^\dagger \boldsymbol{J}_{\boldsymbol{x}} \hat{\boldsymbol{R}}_{\boldsymbol{x}}(N) \right] \\
&= -\boldsymbol{I} + \frac{1}{N^2} \boldsymbol{R}_{\boldsymbol{x}} \mathbb{E}\left[\left(\boldsymbol{J}_{\boldsymbol{x}} \boldsymbol{X} \boldsymbol{X}^\dagger \boldsymbol{J}_{\boldsymbol{x}}^\dagger \right) \right] \boldsymbol{R}_{\boldsymbol{x}}^\dagger \\
&= -\boldsymbol{I} + \frac{1}{N^2} \boldsymbol{R}_{\boldsymbol{x}} \mathbb{E}\left[\left(\boldsymbol{Z} \boldsymbol{Z}^\dagger \right)^2 \right] \boldsymbol{R}_{\boldsymbol{x}}^\dagger ,
\end{aligned}
$$

where we pre- and post-multiplied by $\boldsymbol{I} = \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{J}_{\boldsymbol{x}} = \boldsymbol{J}_{\boldsymbol{x}}^\dagger \boldsymbol{R}_{\boldsymbol{x}}^\dagger$, respectively.

If $\boldsymbol{W} = \boldsymbol{Z}\boldsymbol{Z}^\dagger$, then $\boldsymbol{W}$ is a Wishart matrix, and its second moment is given by [99]

$$
\mathbb{E}\left[\boldsymbol{W}^2 \right] = N \boldsymbol{J}_{\boldsymbol{x}} \mathrm{Tr}\left\{ \boldsymbol{J}_{\boldsymbol{x}} \right\} + (N + N^2) \boldsymbol{J}_{\boldsymbol{x}}^2 .
$$

Plugging this back into the expression for $\boldsymbol{T}_1$ above,

$$\boldsymbol{T}_1 = \frac{1}{N}\left(\boldsymbol{R}_{\boldsymbol{x}}\mathrm{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\} + \boldsymbol{I}\right) .$$

From $\boldsymbol{T}_1$ and T_2,

$$\rho^2_{\text{rags-ls}}(N) = \frac{\boldsymbol{h}_0^\dagger\left(\boldsymbol{R}_{\boldsymbol{x}}\mathrm{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\} + \boldsymbol{I}\right)\boldsymbol{h}_0 + \sigma^2\mathrm{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\}}{N} .$$

□

Note that because of the assumptions in the analysis that $N \gg \max_{c\in\mathcal{C}}|c|$ and $N \not\gg M$, it is necessary that $M \gg \max_{c\in\mathcal{C}}|c|$. In other words, the analysis only applies to graphs with relatively small clique sizes (in relation to the size of the graph). An example of a useful case where the analysis is accurate is when the input is drawn from a Markov chain of length 50, and N ranges from (say) 30 to 100. In that case, N is 15—50 times the size of the largest clique (2), so it is reasonable to assume that $N \gg \max_{c\in\mathcal{C}}|c|$, whereas $N \sim M$. The accuracy of the analysis in this regime is verified by the simulation results of Figure 5-1.

5.3.2 Comparison to Conventional LS

For comparison, the performance of the conventional LS algorithm was seen using (2.12a) [130] to be

$$\mathbb{E}\left[\left\|\boldsymbol{h}_0 - \hat{\boldsymbol{h}}_{\text{ls}}\right\|^2\right] = \frac{\sigma^2}{N-M}\mathrm{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\} , \tag{5.9}$$

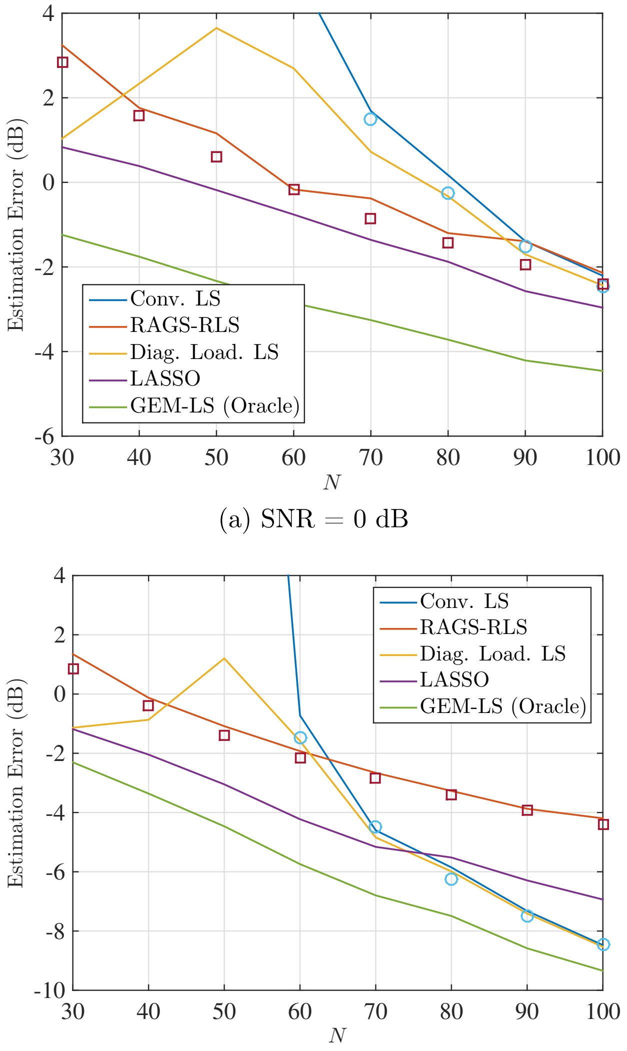
The first term of the RAGS-LS error of (5.8a) is very similar to the error of the conventional LS in (5.9). However, the denominator is different because $N \gg \max_{c\in\mathcal{C}}|c|$. It is very similar to the LS performance when $N \gg M$ [78]. The second term appears to be a kind of residual term. The residual term is due to improperly enforced constraints and manifests as a kind of effective noise. It may be seen from the rearranged expression of (5.8b) that, roughly speaking, the "effective noise level" in the algorithm is nearly the total signal plus noise power $\boldsymbol{h}_0^\dagger\boldsymbol{R}_{\boldsymbol{x}}\boldsymbol{h}_0 + \sigma^2$. In other words, the algorithm behaves similarly to LS in a regime with infinite data (this is a consequence of the assumption that $N \gg \max_{c\in\mathcal{C}}|c|$), but in a very noisy (nearly 0 dB SNR) regime. More explanation of the reasons behind this effect and the nature of the tradeoff may be found in Section 5.3.4.

The following features of the performance of RAGS-LS are worth noting:

- As $N \to \infty$ for fixed M, the error of both LS and RAGS-LS converges to 0. In other words, RAGS-LS is a consistent estimator of an LTI system, which is a nice property.
- As $\sigma^2 \to 0$, the error of the LS algorithm vanishes, but the error of the RAGS-LS algorithm does not, because of the first term which is independent of noise variance. Thus, even in a noise free environment, RAGS-LS does not necessarily perform well. The property that the error does not decrease beyond a point with noise does significantly reduce the utility of the RAGS-LS algorithm.
- It is useful to obtain regimes in which RAGS-LS outperforms LS. To do this, assume that $\mathrm{Tr}\left\{\boldsymbol{J}_{\boldsymbol{x}}\right\} \gg \|\boldsymbol{h}_0\|^2$. Then, identifying that the average signal power is given by $\boldsymbol{h}_0^\dagger \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{h}_0$, it is simple to see that the error of LS given by (5.9) exceeds that of RAGS-LS from (5.8a) when
$$\mathrm{SNR} < \frac{M}{N-M}\,, \tag{5.10a}$$
or equivalently when
$$N < M\left(\frac{1}{\mathrm{SNR}} + 1\right)\,. \tag{5.10b}$$

  The implication of (5.10) is that RAGS-LS can only outperform LS when the SNR is very small, or the amount of data is very small. For example if $N = 2M$, then the SNR would have to be less than 0 dB for RAGS-LS to outperform conventional LS. Similarly, for an SNR of 10 dB, (5.10b) indicates that the RAGS-LS algorithm is only better provided that $N < 1.1M$.

The above place strong constraints on regimes where RAGS-LS outperforms LS, to the extent that in most time-invariant practical applications, the RAGS-LS solution may not be suitable. On the other hand, it should also be emphasized that RAGS-LS is far more efficient than LS from a computational complexity point of view.



(a) SNR = 0 dB

(b) SNR = 6 dB

Figure 5-1: Mean Square Estimation Error for conventional LS, RAGS-LS, LASSO and Diagonally-Loaded LS and GEM-LS (with oracle iteration). The markers on the plots indicate the points predicted by theory, and the lines are simulated performance curves.

5.3.3 Simulated Performance of RAGS-LS

To understand how the RAGS-LS algorithm performs compared to other algorithms to solve least squares type problems, various methods are simulated and the results plotted in Figure 5-1 as a function of N for $M = 50$ for SNRs 0 dB and 6 dB. The data is generated as independent realizations of a Markov chain of length M. Thus there are $M - 1 = 49$ cliques of size 2 and 48 separators of size 1. The methods considered are:

1. Conventional LS

2. RAGS-LS

3. Diagonally loaded least squares

4. LASSO as described in the simulation results of Section 4.4.5, calculated as before using the SPGL solver [17].

5. The GEM-LS algorithm of Chapter 4, with the oracle iteration. As discussed in Section 4.4.3, practical methods are available that can get close to the oracle in terms of performance; the oracle is chosen for reference.

The markers on the conventional LS algorithm and RAGS-LS algorithm represent predictions from (5.9) and (5.8a), respectively; while the lines for each method represent simulated results.

In the low SNR, low sample size regime, the RAGS-LS algorithm can outperform conventional LS. However, given more data and at higher SNRs, the conventional LS algorithm does better, as predicted by (5.8a) and (5.9). The lower the SNR, the more observations are needed for conventional LS to outperform RAGS-LS. As explained in Chapter 4, the GEM-LS algorithm, which is designed to exploit the same graphical model structure as the RAGS-LS to improve performance rather than to decrease computational complexity, performs extremely well with the oracle iteration (i.e., the best possible iteration) and indeed, with the system in question, it outperforms all the other methods in terms of mean squared error.

When compared to diagonally loaded LS, RAGS-LS is comparable or better at low SNRs and small N, showing the same trend as against conventional LS. The peak in the error of the diagonally loaded LS around $M \sim N$ is due to random matrix effects [129], [130] (also mentioned in Section 4.4.5).

The LASSO algorithm outperforms RAGS-LS by about 3 dB in the entire regime of interest. While the complexity of LASSO can vary depending on how the problem is solved, with the SPGL solver which uses fast C routines for computation, the RAGS-LS algorithm is about 100 times faster on average, even though it is implemented entirely with much slower Matlab routines. The SPGL algorithm can be made more accurate at the expense of more computation, or faster at the expense of accuracy by tuning the parameter, but the presented results were judged to be a fair comparison in terms of both complexity and performance.

5.3.4 Interpreting the Performance Degradation

To obtain RAGS-LS, additional structure has been imposed on the problem, and in particular, the inverse covariance matrix has been replaced with a *better* estimate subject to the constraints [181]. However, the additional structure appears to have caused a performance deterioration in some regimes, a fact predicted by analysis and borne out by simulation.

Therefore, the question arises: why should the introduction of additional structure degrade performance, even when that structure is correctly assumed? The answer comes from the cross correlation term. This term has been computed in exactly the same way as in the conventional LS algorithm. However, this is not the correct way to enforce structure in the cross-correlation term. Indeed, in Section 4.1, it was noted that the cross correlation term is a statistic of the *joint* distribution $p_{\mathbf{x},y}(\boldsymbol{x}, y) = p_{\mathbf{x}}(\boldsymbol{x})p_{y|\mathbf{x}}(y|\boldsymbol{x})$, whose structure was captured by the augmented graph of Section 4.1.2.

This leads to an insight that ties several results of the preceding sections together and provides a useful way to think about the recursive framework. In order to make this more concrete, let the RAGS-LS solution for any clique $c \in \mathcal{C}$ computed using n observations (for any $n < N$) be given by $\hat{\boldsymbol{h}}_{\mathrm{rags-ls}_c}(n)$. Using the Sherman-Morrison Matrix inversion lemma, it is possible to write

$$\begin{aligned}
\hat{\boldsymbol{h}}_{\mathrm{rags-ls}_c}(n) &= \boldsymbol{R}_{\mathbf{x}_c}^{-1}(n)\left(\sum_{p=1}^{n} \boldsymbol{x}_c(p)y^*(p)\right) \\
&= \left[\boldsymbol{R}_{\boldsymbol{x}_c}(n-1) + \boldsymbol{x}_c(n)\boldsymbol{x}_c^{\dagger}(n)\right]^{-1}\left(\sum_{p=1}^{n} \boldsymbol{x}_c(n)y^*(n)\right) \\
&= \hat{\boldsymbol{h}}_{\mathrm{rags-ls}}(n-1) + \frac{\boldsymbol{R}_{\mathbf{x}_c}^{-1}(n-1)\boldsymbol{x}_c(n)\boldsymbol{x}_c^{\dagger}(n)}{1+\boldsymbol{x}_c^{\dagger}(n)\boldsymbol{R}_{\mathbf{x}_c}^{-1}(n-1)\boldsymbol{x}_c(n)}\Big(y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\mathrm{rags-ls}_c}(n-1)\boldsymbol{x}_c(n)\Big)^* .
\end{aligned} \tag{5.11}$$

The steps above are exactly the same as those used to write the LS solution in a recursive form to give the RLS algorithm [78] (see also Section 2.2.2). Note that $\hat{\boldsymbol{h}}_{\mathrm{rags-ls}_c}(N)$ is the same as $\hat{\boldsymbol{h}}_{\mathrm{rags-ls}_c}$ in (5.5), i.e., performing the recursive update of (5.11) for N steps is exactly identical to a block implementation of RAGS-LS.

By expressing the RAGS-LS solution in a recursive form, (5.11) provides a hint into how inconsistent modeling degrades performance. Specifically, using the linear model, it is easy to write

$$\begin{aligned} y(n) &= \boldsymbol{h}_0^\dagger \boldsymbol{x}(n) + v(n) \\ &= \boldsymbol{h}_{0_c}^\dagger \boldsymbol{x}_c(n) + \boldsymbol{h}_{0_{\backslash c}}^\dagger \boldsymbol{x}_{\backslash c}(n) + v(n)\,, \end{aligned} \tag{5.12}$$

where the subscript $\backslash c$ indicates elements of the vector that do not correspond to the elements of the clique c.

From this, it becomes evident that, for clique c, the "effective noise" seen by the RAGS-LS algorithm at time n is $v(n) + \boldsymbol{h}_{0_{\backslash c}}^\dagger \boldsymbol{x}_{\backslash c}(n)$; in contrast, the standard LS algorithm only sees the actual noise in the system (i.e., $v(n)$). Averaging over realizations, for clique c, the noise variance is

$$\sigma^2 + \boldsymbol{h}_{0_{\backslash c}}^\dagger \boldsymbol{R}_{\boldsymbol{x}\backslash c} \boldsymbol{h}_{0_{\backslash c}}\,,$$

and the SNR is

$$\frac{\boldsymbol{h}_{0_c}^\dagger \boldsymbol{R}_{\boldsymbol{x}c} \boldsymbol{h}_{0_c}}{\sigma^2 + \boldsymbol{h}_{0_{\backslash c}}^\dagger \boldsymbol{R}_{\boldsymbol{x}\backslash c} \boldsymbol{h}_{0_{\backslash c}}^\dagger}$$

whereas the actual SNR is $\boldsymbol{h}_0^\dagger \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{h}_0 / \sigma^2$.

Thus, the partial LS solutions of (5.5) are each computed in what appears to the algorithm to be a very noisy environment. On the other hand, each partial LS algorithm is of smaller dimension than M (for typical graphs of interest, it is significantly smaller). Thus, the effect of RAGS-LS is to solve several small problems, as intended. However, in doing so non-iteratively, it is forced to accept more noise in each problem.

Stated a different way: the two effects that contribute to error in linear parameter identification are problem size and noise level. RAGS-LS trades problem size for increased noise. The precise trade-off levels vis-à-vis conventional LS can be understood by comparing (5.8b) and (5.9). Ignoring the $\|\boldsymbol{h}_0\|^2$ term of (5.8b) and comparing to (5.9), RAGS-RLS behaves like the RLS algorithm in a regime where there is infinite data, but with a noise power of

$\boldsymbol{h}_0^\dagger \boldsymbol{R_x} \boldsymbol{h}_0 + \sigma^2$. This corresponds to an overall effective system SNR of

$$\mathrm{SNR}_{\text{eff}} = \frac{\boldsymbol{h}_0^\dagger \boldsymbol{R_x} \boldsymbol{h}_0}{\sigma^2 + \boldsymbol{h}_0^\dagger \boldsymbol{R_x} \boldsymbol{h}_0} = \frac{\mathrm{SNR}}{1 + \mathrm{SNR}} \,,$$

which is less than 0 dB!

This additionally explains why RAGS-LS has an advantage over LS when the underlying system is either very noisy or very large (the precise meaning of "very" noisy is given by (5.10)). In either of these cases, the penalty incurred by increasing the noise level for each clique (and separator) is sufficiently offset by the advantage of solving lower dimensional problems, either because the noise level is very large to begin with, or because the lack of data results in a poor estimate of the inverse covariance matrix when computed conventionally.

GEM-LS is able to avoid making the trade-off by iterating, between, in essence: distinguishing the signal from the noise for each clique (computing the clique outputs) and solving the set of reduced dimensional estimation problems.

Equation (5.11) suggests an improvement to the algorithm when implemented in a recursive context. Note that each clique is innovated by $y(n) - \hat{\boldsymbol{h}}^\dagger_{\text{rags-ls}_c}(n-1)\boldsymbol{x}_c(n)$, which leads to the fraction of $y(n)$ that is *not* encompassed by the clique to be treated as noise. This can be avoided by re-combining all the clique estimates into an overall vector *at each time* (rather than just once after all the data is received, as is done by RAGS-LS), and then utilizing that to compute a common innovation term for all the cliques. This is indeed the idea behind the RAGS-RLS[3] algorithm that is introduced in Section 5.4.

On an aside, the performance degradation reinforces a statement that was made in Section 2.1.5: that it is not easy in general to enforce input structure in linear system identification problems. The residual term in the error of RAGS-LS demonstrates this very clearly in the case of graphical model structure. The problem is that the naive method of incorporating the structure that is used by RAGS-LS does not account for the fact that if $\boldsymbol{x}$ is structured, then the output $\boldsymbol{y}$ is structured in a very non-obvious way. Typically, exploiting the input structure correctly means that the output also needs to be handled correctly. It appears

[3] It may occur to the reader that a more powerful way to develop a recursive algorithm may be to cast the GEM-LS algorithm, which correctly enforces the structural constraints, into a recursive framework; nonetheless, in this work, this was not done for reasons explained in Chapter 6.

to be a non-trivial and interesting problem as to how other kinds of input structure could be exploited for system identification, and what makes it challenging is the effect of input structure on the output of the system. The effect of not accounting for that effect may be seen in, e.g., the work of Blair [22].

For the graphical model GEM-LS does this for graphical model structured inputs by casting the entire problem in a graphical model framework. However, RAGS-LS ignores this point for the sake of complexity reduction and pays the penalty in terms of performance.

5.4 Relaxed Approximate Graph-Structured Recursive Least Squares (RAGS-RLS)

A recursive form of the RAGS-LS algorithm that computes a joint innovation term for all the cliques is now introduced. It is developed with exponential windowing so that it is applicable to the more general problem of tracking a time-varying system. The resulting algorithm is called the relaxed approximate graph-structured recursive least squares (RAGS-RLS) algorithm.

The problem of tracking a time-varying system recursively as its inputs and outputs are observed was introduced in Section 2.2.1. Conventional methods for solving the tracking problem include the EW-RLS and LMS algorithms and their variants introduced in Section 2.2.2.

Before proceeding, a few conventions are defined to simplify the notation. As explained in Section 2.2.2, for the RLS algorithm, an exponential weighting factor λ is introduced in order to handle time-variance; and the exponentially weighted sample inverse covariance matrix $\hat{\boldsymbol{R}}_{\boldsymbol{x}\lambda}(N)$ and sample cross correlation $\hat{\boldsymbol{r}}_{\boldsymbol{x}y_\lambda}(n)$ were defined in (2.20). In a slight abuse of notation, it will be assumed in this chapter that $\hat{\boldsymbol{R}}_{\boldsymbol{x}}(N)$ and $\hat{\boldsymbol{r}}_{\boldsymbol{x}y}(n)$ refer to the exponential weighted versions of these quantities. That is, denote

$$\hat{\boldsymbol{R}}_{\boldsymbol{x}}(n) \equiv \hat{\boldsymbol{R}}_{\boldsymbol{x}\lambda}(n) = \sum_{p=1}^{n} \lambda^{n-p} \boldsymbol{x}(p)\boldsymbol{x}^{\dagger}(p) \tag{5.13a}$$

$$\hat{\boldsymbol{r}}_{\boldsymbol{x}y}(n) \equiv \hat{\boldsymbol{r}}_{\boldsymbol{x}y_\lambda}(n) = \sum_{p=1}^{n} \lambda^{n-p}\boldsymbol{x}(p)y^*(p)\,, \tag{5.13b}$$

dropping the λs to avoid overly burdensome notation. In other words, all the sample statistics are assumed to be computed with some exponential weighting factor $0 < \lambda \leq 1$. Additionally, it will be assumed that RLS refers to the exponentially weighted RLS (EW-RLS) of Section 2.2.2 and RAGS-RLS refers to exponentially weighted RAGS-RLS. Setting $\lambda = 1$, the algorithms are suitable for recursive estimation of time-invariant systems.

With these clarifications in mind, the development of the algorithm is now considered.

5.4.1 Derivation of RAGS-RLS

The derivation of RAGS-RLS follows from Section 5.3.4 and, in particular, from (5.11). The LS algorithm over each clique and separator is implemented recursively (with the addition of an exponential weighting factor). However, the *innovation* term for clique c in (5.11), which is

$$y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\text{rags}-\text{ls}_c}(n-1)\boldsymbol{x}_c(n)\,,$$

is now replaced by

$$y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\text{rags}-\text{rls}}(n-1)\boldsymbol{x}(n)$$

for every $c \in \mathcal{C}$ and $s \in \mathcal{S}$, where,

$$\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}}(n) = \sum_{c\in\mathcal{C}} \left[\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_c}(n)\right]_{c\times 1}^{M\times 1} - \sum_{s\in\mathcal{S}} \left[\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_s}(n)\right]_{s\times 1}^{M\times 1}\,.$$

In other words, the separate innovation terms for the cliques and separators are replaced by a single combined innovation term. As the estimate is being computed recursively, $\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}}(n)$ has to be computed anyway, so this step does not add any extra computation beyond what is intrinsically required by the algorithm. The reason this is done is to mitigate the “effective noise” issue described in Section 5.3.4.

As a result of this modification, the RAGS-RLS algorithm is *not* just a recursive implementation of the RAGS-LS algorithm. For time-invariant systems, running RLS with N

	Conventional RLS	RAGS-RLS
Initialization	$\hat{\boldsymbol{J}}_{\boldsymbol{x}}(0) = \delta^{-1}\boldsymbol{I}_M$ $\hat{\boldsymbol{h}}_{\text{rls}}(0) = \boldsymbol{0}$	$\hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(0) = \delta^{-1}\boldsymbol{I}_{\lvert c\rvert}$ $\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_c}(0) = \boldsymbol{0}$
Kalman Gain Update	$\boldsymbol{k}(n) = \dfrac{\hat{\boldsymbol{J}}_{\boldsymbol{x}}(n-1)\boldsymbol{x}(n)}{\lambda_r + \boldsymbol{x}^{\dagger}(n)\hat{\boldsymbol{J}}_{\boldsymbol{x}}(n-1)\boldsymbol{x}(n)}$	$\boldsymbol{k}_c(n) = \dfrac{\hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(n-1)\boldsymbol{x}_c(n)}{\lambda_s + \boldsymbol{x}_c^{\dagger}(n)\hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(n-1)\boldsymbol{x}_c(n)}$
Filtering Current Input	$\hat{y}_{\text{rls}}(n) = \hat{\boldsymbol{h}}_{\text{rls}}^{\dagger}(n-1)\boldsymbol{x}(n)$	$\hat{y}_{\text{rags}-\text{rls}}(n) = \hat{\boldsymbol{h}}_{\text{rags}-\text{rls}}^{\dagger}(n-1)\boldsymbol{x}(n)$
Coefficient Update	$\hat{\boldsymbol{h}}_{\text{rls}}(n) = \hat{\boldsymbol{h}}_{\text{rls}}(n-1) + \boldsymbol{k}(n)\left(y(n) - \hat{y}_{\text{rls}}(n)\right)^{*}$	$\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_c}(n) = \hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_c}(n-1) + \boldsymbol{k}_c(n)\left(y(n) - \hat{y}_{\text{rags}-\text{rls}}(n)\right)^{*}$
Combining Estimates	—	$\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}}(n) = \sum_{c\in\mathcal{C}}\left[\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_c}(n)\right]_{c\times 1}^{M\times 1} - \sum_{s\in\mathcal{S}}\left[\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_s}(n)\right]_{s\times 1}^{M\times 1}$
Inv. Cov. Mtx. Update	$\hat{\boldsymbol{J}}_{\boldsymbol{x}}(n) = \dfrac{\hat{\boldsymbol{J}}_{\boldsymbol{x}}(n-1) - \boldsymbol{k}(n)\boldsymbol{x}^{\dagger}(n)\hat{\boldsymbol{J}}_{\boldsymbol{x}}(n-1)}{\lambda_r}$	$\hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(n) = \dfrac{\hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(n-1) - \boldsymbol{k}_c(n)\boldsymbol{x}_c^{\dagger}(n)\hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(n-1)}{\lambda_s}$

Figure 5-2: Steps of the RAGS-RLS algorithm listed alongside the steps of the RLS algorithm. For RAGS-RLS, the steps marked for each clique are applied to each clique and each separator at each time (this is not shown for conciseness).

data samples is the same as performing LS estimation with the data in a block. In contrast, running block RAGS-LS is not equivalent to running RAGS-RLS on the same data. A comparison between RAGS-RLS and RAGS-LS (with exponential weighting) is included in Figure 5-3.

Apart from the modified innovation term, an exponential weighting factor is introduced into the computation of the inverse covariance matrix and cross-correlation statistics for every clique and separator. Then, the steps of the (exponentially weighted) RAGS-RLS algorithm are shown alongside those of the conventional RLS algorithm in Figure 5-2. Note that the exponential weighting factor of the RLS algorithm and the RAGS-RLS algorithm are not necessarily the same, so λ_r, λ_s are used to denote the exponential weighting factor for the conventional RLS and RAGS-RLS algorithms, respectively (however, see Section 5.4.3).

Moreover, although it has been assumed in Figure 5-2 that the value of the exponential weighting factor is the same for each clique and separator, it may be possible to set a different value of λ for each clique/separator to allow them to vary at different rates because the RAGS-RLS coefficient update step for the cliques and separators are decoupled. However, there may be constraints on how the different λ can be chosen to keep the algorithm consistent. This issue has not been explored in any depth.

5.4.2 Computational Complexity

The complexity of RAGS-RLS is

$$\mathcal{O}\left(\sum_{c\in\mathcal{C}} |c|^2 + \sum_{s\in\mathcal{S}} |s|^2\right) = \mathcal{O}\left(D\left(\max_{c\in\mathcal{C}} |c|\right)^2\right) . \tag{5.14}$$

Depending upon the graph, RAGS-RLS can be considerably more efficient than RLS. To obtain some insight into computational complexity improvements as a function of graph structure, two cases are considered:

- Assuming $D = \mathcal{O}(M)$, RAGS-RLS is more efficient than RLS if

$$\max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(\sqrt{M}\right) .$$

- As mentioned in Section 4.4.4, several graphs of interest satisfy the stronger property that $D \max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(M\right)$, and for such graphs, it is sufficient that

  $$\max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(M\right) ,$$

  which is trivially true for this class of graphs. Thus, if $D \max_{c\in\mathcal{C}} |c| = \mathcal{O}\left(M\right)$ then RAGS-RLS is always less computationally complex than RLS.

This is without parallelization, which, as with RAGS-LS, could provide a speed up of up to $\mathcal{O}\left(D\right)$. The gains can be quite considerable for relatively sparse graphs.

5.4.3 Performance Analysis

As mentioned in Section 2.2.2, only very limited analysis of the performance of algorithms for the time-varying system tracking problem appears possible—see, in particular, the derivation in the book by Haykin [78, Section 14.5]. An analysis of the RAGS-RLS algorithm is now carried out with the same assumptions as made in this reference to provide first order insight into the relative performances of the RAGS-RLS and conventional RLS algorithms. This will also help to understand how to choose the exponential weighting factor.

The analysis for RLS begins by noting that the RLS solution at time n can be written in the following form:

$$\hat{\boldsymbol{h}}_{\mathrm{rls}}(n) = \hat{\boldsymbol{h}}_{\mathrm{rls}}(n-1) + \hat{\boldsymbol{J}}_{\boldsymbol{x}}(n)\boldsymbol{x}(n)\left(y(n) - \hat{\boldsymbol{h}}_{\mathrm{rls}}^{\dagger}(n-1)\boldsymbol{x}(n)\right)^{*} . \tag{5.15}$$

To this, the direct averaging approximation [57] is applied so that, for λ_r close to 1,

$$\hat{\boldsymbol{R}}_{\boldsymbol{x}}(n) = \sum_{p=1}^{n} \lambda^{n-p}\boldsymbol{x}(n)\boldsymbol{x}^{\dagger}(n) \approx \frac{1}{1-\lambda_r}\boldsymbol{R}_{\boldsymbol{x}} ,$$

so that its inverse, which appears in the analysis, can be written as

$$\hat{\boldsymbol{J}}_{\boldsymbol{x}}(n) = \left(\sum_{p=1}^{n} \lambda^{n-p}\boldsymbol{x}(n)\boldsymbol{x}^{\dagger}(n)\right)^{-1} \approx (1-\lambda_r)\boldsymbol{J}_{\boldsymbol{x}} . \tag{5.16}$$

The reader is referred to [57] for a justification of this approximation. For our purposes, it suffices to state that direct-averaging is valid when the time-varying inverse covariance matrix is quasi-deterministic, which in turn is true when the system variation is slow with respect to the memory of the adaptive algorithm. With this approximation, (5.15) becomes

$$\hat{\boldsymbol{h}}_{\mathrm{rls}}(n) = \hat{\boldsymbol{h}}_{\mathrm{rls}}(n-1) + (1-\lambda_r)\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{x}(n)\left(y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\mathrm{rls}}(n-1)\boldsymbol{x}(n)\right)^{*} . \tag{5.17}$$

For the RAGS-RLS algorithm, it suffices to recognize that the algorithm simply runs (exponentially-weighted) RLS over each clique using the overall output $y(n)$ as the desired output and the total signal estimate from all cliques and separators as the signal estimate. Thus, the RAGS-RLS solution over each clique can be written as

$$\hat{\boldsymbol{h}}_{\mathrm{rags-rls}_c}(n) = \hat{\boldsymbol{h}}_{\mathrm{rags-rls}_c}(n-1) + \hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(n)\boldsymbol{x}_c(n)\left(y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\mathrm{rags-rls}}(n-1)\boldsymbol{x}(n)\right)^{*} , \tag{5.18}$$

which is the same as (5.15), except over a single clique. In this case, make the direct-averaging assumption over each clique and separator, so that $\hat{\boldsymbol{J}}_{\boldsymbol{x}_c}(n) \approx (1-\lambda_s)\boldsymbol{J}_{\boldsymbol{x}_c}$. Then, combining all the clique and separator solutions,

$$\begin{aligned}
\hat{\boldsymbol{h}}_{\mathrm{rags-rls}}(n) &= \sum_{c\in\mathcal{C}} \left[\hat{\boldsymbol{h}}_{\mathrm{rags-rls}_c}(n)\right]_{c\times 1}^{M\times 1} - \sum_{s\in\mathcal{S}} \left[\hat{\boldsymbol{h}}_{\mathrm{rags-rls}_s}(n)\right]_{s\times 1}^{M\times 1} \\
&= \hat{\boldsymbol{h}}_{\mathrm{rags-rls}}(n-1) + (1-\lambda_s)\left(\sum_{c\in\mathcal{C}} \left[\boldsymbol{J}_{\boldsymbol{x}_c}\boldsymbol{x}_c(n)\right]_{c\times 1}^{M\times 1} - \sum_{s\in\mathcal{S}} \left[\boldsymbol{J}_{\boldsymbol{x}_s}\boldsymbol{x}_s(n)\right]_{c\times 1}^{M\times 1}\right) \times \\
&\qquad \left(y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\mathrm{rags-rls}}(n-1)\boldsymbol{x}(n)\right) \\
&= \hat{\boldsymbol{h}}_{\mathrm{rags-rls}}(n-1) + (1-\lambda_s)\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{x}(n)\left(y(n) - \hat{\boldsymbol{h}}^{\dagger}_{\mathrm{rags-rls}}(n-1)\boldsymbol{x}(n)\right) .
\end{aligned} \tag{5.19}$$

Going from the second to the third step is a simple matter of algebra.

Comparing (5.17) to (5.19), it becomes clear that to a first-order approximation, the update expressions for RAGS-RLS and RLS are identical when $\lambda_r = \lambda_s$. In other words, if $\lambda_r = \lambda_s$ the algorithms are expected to have identical performance in tracking time varying systems when direct-averaging is valid. This is true if the system variations are slow in comparison to the memory of the algorithm, and a long time has elapsed (i.e., sufficient data has been observed) so that the statistics of the solution have converged.

Note that direct averaging does not capture the effect of a finite averaging window (which is implied by $\lambda < 1$) on the estimate of the inverse covariance matrix. The only effect of the exponential weighting appears to be the scaling due to the assumptions made in the analysis. Nonetheless, experience shows that, in reality, the tracking and noise rejection abilities of the algorithm are related to the exponential windowing in a more involved manner, so the optimum value of λ can be different for RAGS-RLS and RLS in practice.

5.4.4 Additional Performance Characteristics

The simple modification to the innovation that was made in Section 5.4.1 appears to have had the desired effect, at least in the long term (a performance comparison between the two is shown in Figure 5-3). The updates of RAGS-RLS and RLS are the same, which implies that the steady state performance of both algorithms should be very similar. This result is independent of the system dynamics and sizes, which is quite encouraging.

It should be noted that direct averaging is a long time-scale result. In the transient regime, when the statistics have not yet settled, the local RLS algorithms (the GS-RLS subproblems) for the different cliques and separators operate with more noise than actually exists in the system. It is easy to see this intuitively. For the first data point, RAGS-RLS is the same as RAGS-LS, so for the first data point, the noise for clique c is given by

$$v(1) + \boldsymbol{h}_{0_{\setminus c}}^{\dagger}(1)\boldsymbol{x}_{\setminus c}(1)\,,$$

which has variance larger than the actual noise $v(1)$. The correction applied by combining the clique/separator estimates gradually mitigates this issue, so the noise variance decreases over time to converge to the actual noise in the system in the long term.

However, this transient behavior has implications on how the algorithm is initialized. It is well understood that the amount of diagonal loading δ with which the RLS algorithm should be initialized is related to the SNR—the more noise there is, the larger δ should be [78]. Because RAGS-RLS initially operates in a noisy environment, it may be necessary to use a larger value of δ to initialize RAGS-RLS than would typically be used for RLS in order to prevent the algorithm from initially diverging.[4]

Another consequence of the transient behavior is that RAGS-RLS is suitable for tracking systems over a long period of time, but may not be much better than RAGS-LS for estimating time-invariant systems with limited data, as it is unlikely that with limited data the statistics converge fast enough for the direct-averaging results to apply. Rather the transient behavior will be present, so the effective noise in the system is likely to be quite large in this case.

Finally, note that the updates are only identical up to a first-order approximation. In practice, it is unlikely that the mismatch caused by the approximate modeling of clique and separator outputs is ever completely overcome, especially with time-varying systems, for which the cross-correlation between $\boldsymbol{x}(n)$ and $y(n)$ varies with time and only limited observations are available for estimation. As such, the effective noise for the RAGS-RLS algorithm is likely to always be larger than the actual noise in the system (although less than that which would be observed if the clique parameters $\hat{\boldsymbol{h}}_{\text{rags}-\text{rls}_c}(n-1)$ were used to compute the innovation).

An interesting ramification of this is that a larger value of λ_s is likely to favor the RAGS-RLS algorithm. This is somewhat counterintuitive, because it may appear at first glance that RAGS-RLS is solving smaller tracking problems. However, it should be noted that the innovation term (which is the only term that depends upon the unknown system being tracked) is computed jointly for all the cliques and separators. So the "tracking" portion of the update is actually a joint update step. Additionally, the update is performed in a regime that appears to the algorithm to be noisy, so λ_s should be chosen with noise-rejection in mind, which favors choosing larger values of λ_s.

While the analysis is not sophisticated enough to be able to justify all the above from a mathematical perspective, it is hoped that the insights provided above will guide system design and perhaps lead to better analysis.

[4] There are other ways to initialize RLS that do not rely on diagonal loading, which would likely avoid this issue.

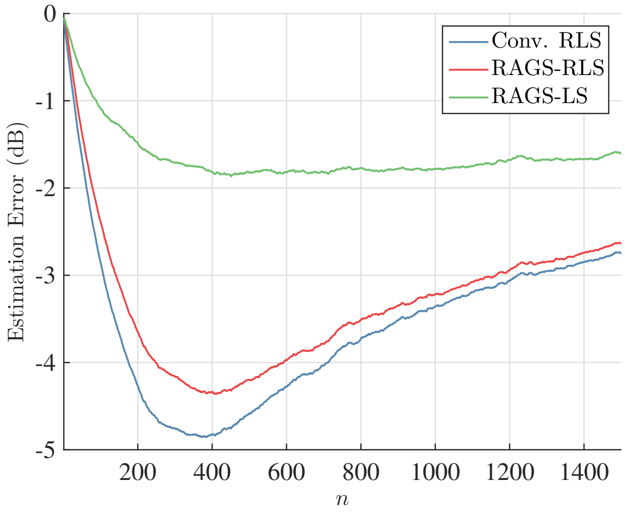


Figure 5-3: Simulated performance of RLS, RAGS-LS and RAGS-RLS in tracking a time-varying system. The plot shows Mean-Squared Estimation Error (MSE) as a function of time n, averaged over 250 realizations.

5.4.5 Simulation Results

The simulated performance of the conventional (exponentially weighted) RLS, (exponentially weighted) RAGS-RLS and RAGS-LS (without the modified innovation, but with exponential weighting) when tracking a time-varying system is shown in Figure 5-3. The dynamical model for the system being tracked is given by

$$\boldsymbol{h}(n) = a\boldsymbol{h}(n-1) + \sqrt{1-a^2}\boldsymbol{w}(n)\,, \tag{5.20}$$

where $a = 0.999$ and $\boldsymbol{w}(n)$ is a zero-mean white multivariate Gaussian process whose covariance matrix is the identity. For all the three algorithms, the exponential weighting factor is 0.999.

The inputs $\boldsymbol{x}(n)$ are realizations of a random vector x, where $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_M$ form a Markov chain (more specifically, an AR-1 process with parameter -0.5). The system being tracked (and hence each input vector) has dimension $M = 100$. The system is corrupted with AWGN with an SNR of 20 dB. The RAGS-RLS and RAGS-LS algorithms are assumed to have access to the true graphical model.

As predicted in Section 5.4.3, the RLS and RAGS-RLS algorithms have very similar long-term performance. Although RLS outperforms RAGS-RLS, the gap (after the convergence period) is only about 0.1 dB. Furthermore, the gap continues to narrow, which reflects the direct-averaging result that, after a very long period of time, the algorithms should have very similar performance. Additionally, the transient performance of RAGS-RLS appears somewhat worse than conventional RLS, as expected.

In contrast, the RAGS-LS algorithm performs poorly in comparison to RAGS-RLS and conventional RLS. This appears to confirm the insight of effective noise deterioration in the RAGS-LS algorithm and the ability of the modified innovation term to mitigate the issue, as explained in Section 5.3.4.

One curious aspect of the performance is that for both RAGS-RLS and RLS, the performance gets worse after about $n = 400$. The reason is that the algorithms are all initialized with diagonal loading, and as time goes on, the effect of the diagonal loading disappears. Diagonally loaded LS (with windowing) may be an improvement over either algorithm, but such an algorithm would not be recursive. The effect of the diagonal loading can be different on different algorithms, depending upon the noise level and size of the system being estimated [129], so the plot for the RAGS-LS algorithm does not have the same characteristic with the chosen value of diagonal loading.

In this case, the complexity of RAGS-RLS is about 4.94% that of the conventional RLS algorithm (99 cliques of size 2 plus 98 scalar separators), which is a significant saving.

The results of this section show that if a relatively sparse graphical model input structure is available, then the RAGS-RLS algorithm is a way to considerably reduce the complexity of the system tracking problem without making too large a compromise on performance.

5.5 Applications of RAGS-RLS

In this section, the RAGS-RLS algorithm is applied to two real-world problems—acoustic echo cancellation filter tracking and adaptive equalization in underwater communication. Note that, as is conventional in practice, the input vectors are not independent realizations of a random variable, but are generated using a tapped-delay line, in both these application. In other words each input vector contains M successive time-samples of an input sequence, and the next vector is obtained by pushing out the last sample and introducing a new one (for multichannel applications, this is done for each channel). Details have been provided in Chapter 3.:

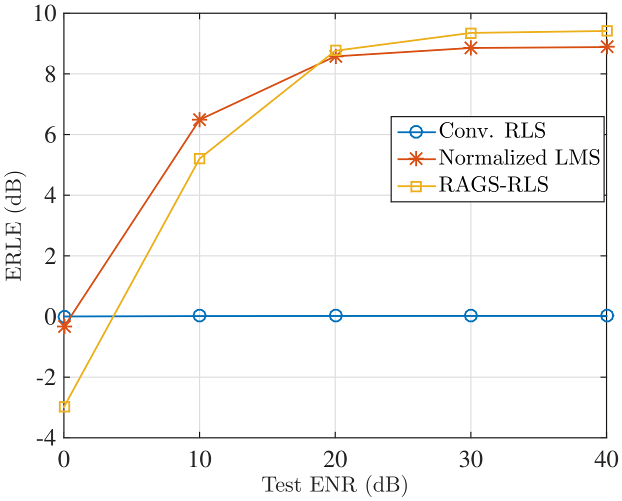
5.5.1 Tracking Acoustic Echo Cancellation Filters

The AEC filter tracking problem was described in Section 3.2.1. The environment for the simulation (room parameters, etc.) is the same as that described in detail in Section 4.5.2. To recap, the room dimensions are $(3.7, 2.74, 3.7)$ meters, and microphone is located at $(1.85, 0.6, 0.3)$. The T_{60} time of the room is 0.25 seconds. The impulse response calculated is the first 50 ms of the room impulse response, which corresponds at a sampling frequency of 16 kHz to 800 time-domain taps for adaptation.

Typically, tracking the filters is required because the environmental conditions (such as humidity, for instance) change with time and because objects move around the space. Due to the restrictions of available simulators (and in particular the simulator used herein developed by Habets [73], which only allows for calculation of the acoustic path between a transmitter and receiver with very minimal flexibility to specify the environment), it is assumed instead that the speaker is moving around the room, thereby causing the echo paths to change smoothly.

The speaker is assumed to move along a randomly defined triangle in the room (a new triangle is defined for each realization and the results are averaged over realizations) over a period of 30 seconds. The speed of the variation is thus different for each realization as it depends upon the length of the path taken by the speaker. As in Section 3.2.1, the noise is generated by a fixed point source outputting white noise from a randomly chosen location. A total of 500 realizations are generated and the results are averaged over realizations.

The filters for all the methods are initialized using a matched filter with a 1 second long white calibration tone from the starting point of the loudspeaker. Different algorithms are then used to track the filters over a time period of 3.125 seconds (corresponding to $50,000$ sample times). For the RAGS-RLS algorithm, the input signal from the moving loudspeaker is assumed to be obtained from a white time-series, so that the input graphical model structure is simply the trivial graph with M independent nodes, as explained in Section 3.2.1.



(a) ERLE vs. ENR

Algorithm	**Time per sample (μs)**
RLS	2500
N-LMS	36.54
RAGS-RLS	36.37

(b) Time for adaptation (per sample)

Figure 5-4: Performance of RLS, N-LMS and RAGS-RLS for tracking acoustic echo cancellation filters. The performance of N-LMS and RAGS-RLS is fairly close, and the times taken by the algorithms are also very close.

The algorithms for comparison are the N-LMS algorithm and the conventional RLS algorithm. However, it should be noted that, in practice, the RLS algorithm would never be used to perform adaptation in this application, because the filters are too large. This not makes the adaptation too computationally complex, but can also destabilize the algorithm due to numerical round-off errors [40].

The step size for the N-LMS algorithm is taken to be $\mu_{\text{nlms}} = 1$, which is the value of the parameter that makes the algorithm H^{∞}-optimal [75] for this application. The exponential weighting factor is chosen as $\lambda_r = \lambda_s = 0.9983$, which corresponds to an exponential window "length" of about 600 samples.

In this case, the ERLE varies with time. The results are presented in terms of ERLE averaged over time as a function of the average ENR in Figure 5-4a. The N-LMS algorithm and the RAGS-RLS algorithm are fairly similar in performance. N-LMS is better at low SNRs whereas RAGS-RLS appears to have an advantage at high SNRs. Both are generally far better than RLS, whose performance is likely poor due to numerical issues.

The same holds for the time taken by the algorithms. Typically, computational complexity would be compared because execution time depends on a variety of implementation-related factors, but for a graph with all independent nodes, (5.14) indicates that the complexity of RAGS-RLS is $\mathcal{O}(M)$. Indeed, the N-LMS and RAGS-RLS algorithms are very close in terms of time taken for very similar implementations, as seen in Figure 5-4b. The sampling time for this application is 62.5 μs, so both can reasonably be used.

Thus, while RAGS-RLS can be used for tracking AEC filters, it shows no real advantage over N-LMS for this application, except perhaps in the high ENR regime.

5.5.2 Adaptive Equalization Using RAGS-RLS

A second application of RAGS-RLS is for adaptive equalization of underwater wireless communication channels. Results regarding structure of the data for this problem may be found in Sections 3.4 and 3.6. The most relevant of these are recapped below.

As shown in Theorem 3.4.2, the received signal in a coherent communication system is cyclostationary if the input signal is wide-sense cyclostationary (as it almost invariably is) and the channel variations are wide-sense stationary. Section 3.4, and in particular Theorem 3.4.1 shows that the frequency coefficients of wide-sense cyclostationary processes are characterized by a graphical model. Section 3.4.6 presents a framework for multichannel frequency-domain adaptive equalization that defines a multichannel input $\boldsymbol{X}$ to the equalizer (defined in (3.17)) characterized by the graphical model of Figure 3-12. The graph is comprised of $\mathcal{O}(M/T)$ disjoint cliques, each of size $\mathcal{O}(T)$.

Additionally, in Section 3.5, it was shown that the effect of having a finite amount of data for processing is to cause “conditional correlation spillover” in the frequency domain. As shown in Section 3.6 for the case of underwater acoustic communication data, this means in practice that it may be beneficial to augment the theoretical graph with edges between cliques corresponding to adjacent frequency coefficients. The number of adjacent cliques that are connected is represented by d, so that $d = 1$ corresponds to the case of the theoretical graph of Figure 3-12, $d = 2$ means that each clique of that graph is connecting to the adjacent ones (this case is shown explicitly in Figure 3-21, and so on.

Julian Day	Epoch Start Hour	Wave Height	Waveheight Swell	Waveheight Wind
290	2:00 AM	0.7 m	0.53 m	0.3 m/s
294	12:00 PM	1.63 m	1.43 m	0.3 m/s
296	12:00 AM	1.47 m	0.6 m	1.13 m/s
300	8:00 AM	3.4 m	2.87 m	1.5 m/s
300	8:00 PM	2.53 m	2.23 m	0.87 m/s

Figure 5-5: Environmental conditions of the representative epochs chosen from the SPACE08 experiment. These represent a wide range of environmental conditions to provide a spectrum of channel characteristics typical to wireless underwater communication.

In this section, the graphical model structure of Figure 3-21 is exploited in conjunction with the RAGS-RLS algorithm in order to perform adaptive equalization of the wireless underwater communication channel. The uncoded m-sequence data from the SPACE08 experiment that was described in Section 3.6 is also used to test the performance of the adaptive equalizers. The field data results show that significant complexity gains with little performance loss, as predicted by the analysis and simulation of Section 5.3, can be realized in a challenging real-world application.

The results presented are for the 200 m range acoustic communication data from the SPACE08 experiment, meaning that the transmitter and receiver were separated by 200 m, which would be considered relatively short-range underwater acoustic communication. The channel is characterized by significant time-varying multipath [22], [37].

As in Section 3.6, the received signal is collected into time-domain blocks of duration 4.6 ms, resulting in block lengths of $M_t = 60$ at an effective baseband sampling rate of about 13 kHz. The frequency coefficients corresponding to baseband frequencies between -4 kHz and 4 kHz are computed (spaced at about 215 Hz, which corresponds to a discrete time frequency spacing of $2\pi/60$) are retained, giving a total of 37 frequencies per received channel. The adaptation dimension (system length being tracked) is $M = 37R$, where R is the number of receivers.

Recall from Section 3.6 that the data in the SPACE08 experiment was collected in two hour intervals termed "epochs." Five representative epochs have been chosen, some of whose relevant environmental conditions are shown in Figure 5-5. These epochs represent a sufficiently wide variety of environmental conditions, so that a diverse set of channel characteristics typical to underwater communication systems is obtained. The performance of the equalizers shown is averaged across the chosen epochs.

The different algorithms under consideration are used to track equalizer coefficients for $N_{\text{track}} = 2 \times 10^5$ symbol times during each chosen epoch ($\sim$ 30 seconds), after an initial settling period of 2×10^3 symbol times ($\sim$ 0.3 seconds). Note that all the algorithms are operating in pilot mode (i.e., they have access to the true transmitted symbol)—the settling period is merely to capture steady state error, which is what is primarily relevant in this application.

The metrics of performance are the mean squared output prediction error (i.e., the error in predicting $y(n)$), given by

$$\text{Prediction Error} = \frac{1}{N_{\text{track}}} \sum_{n=1}^{N_{\text{track}}} |y(n) - \hat{y}(n)|^2 \tag{5.21}$$

and the symbol error rate, given for BPSK signals by

$$p_{\text{error}} = \frac{1}{N_{\text{track}}} \sum_{n=1}^{N_{\text{track}}} \text{ind}\left(\text{sgn}(\hat{y}(n) \neq y(n)\right) . \tag{5.22}$$

In the above, $\hat{y}(n)$ is the soft output of the adaptation algorithm and “ind” represents the boolean indicator function, which is 1 when its argument is true and 0 otherwise.

To help interpret the symbol error rate results, the capacities of binary symmetric channels (BSC’s) with probability of error equal to the symbol error rate achieved by the corresponding adaptation algorithms are also plotted. These is given by

$$\text{Capacity} = 1 + p_{\text{error}} \log_2 p_{\text{error}} + (1 - p_{\text{error}}) \log_2 (1 - p_{\text{error}}) , \tag{5.23}$$

where p_{error} is the error rate achieved by the corresponding adaptation algorithm.[5]

The rationale behind presenting the BSC capacity is that this is the highest rate of a code that could correct the remaining errors in the equalized output in a practical coded communication system. In other words, the entire communication system from the code bits

[5] Note, of course, that the symbol error represents the transition probability of the BSC only because the modulation scheme is BPSK.

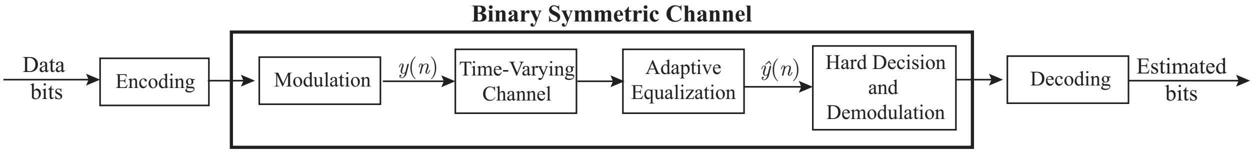


Figure 5-6: A conceptual diagram of the coded communication system demonstrating how the end-to-end channel post-encoding (including modulation, transmission, equalization and symbol decision) can be modeled as a binary symmetric channel. The capacity of the channel for equalizers that use different adaptation algorithms represents the highest rate code that could be used to correct errors in the BSC.

before modulation to the to the demodulated equalizer output can be thought of as a BSC, and its capacity is the highest code rate that can be used for error-free communication across that channel. Figure 5-6 shows a conceptual diagram of the coded communication system and the binary symmetric channel.[6]

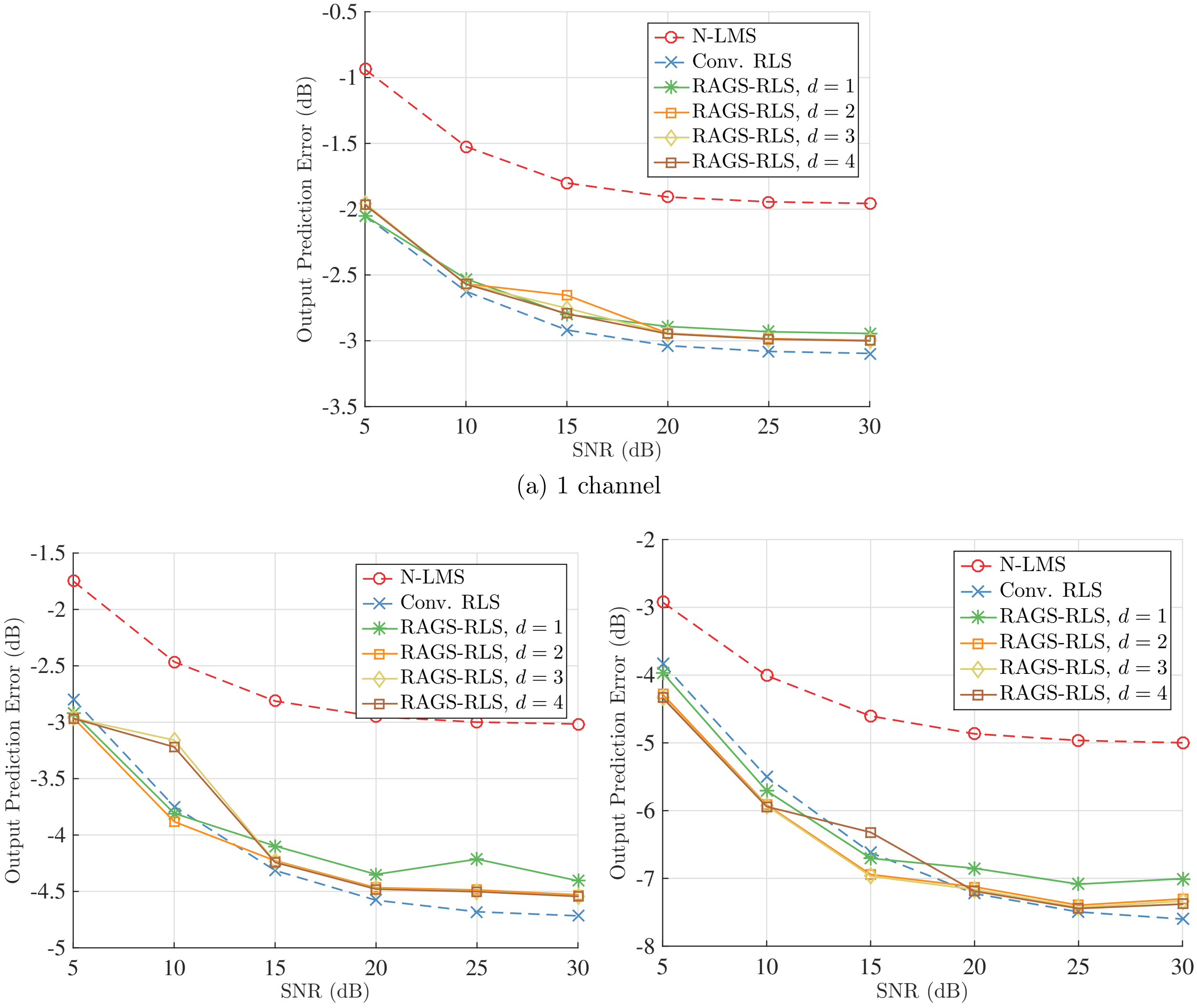
The results are plotted as a function of SNR for different numbers of receivers ($R = 1, 2, 4$). Although the native SNR of the data (i.e., the SNR of the collected data) varies from epoch to epoch, for the 200 m range data under consideration, the native SNR is always above 30 dB. The SNR is degraded to any desired level (below 30 dB) by adding noise acquired in the experiment. Results are presented for SNRs ranging from 5 dB to 30 dB.

The algorithms for which results are presented are the N-LMS algorithm with $\mu_{\text{nlms}} = 0.5$, the RLS algorithm with $\lambda_r = 0.995$ and the RAGS-RLS algorithm with $\lambda_s = 0.9967$. The parameters for all the algorithms have been chosen by trial and error to be the ones that maximized the performance for that particular algorithm[7] (in other words, the results were obtained for different values of the parameters and the best results for each algorithm have been chosen). Finally, for the RAGS-RLS algorithm, results are presented for $d = 1, 2, 3, 4$. This will provide insight into whether the "expected absolute covariance structure" obtained in Section 3.5 really does affect practical performance.

The prediction error, SER and BSC capacity results are in Figure 5-7, Figure 5-8 and

[6] Figure 5-6 is representative of a typical traditional communication system and does not encompass practical improvements such as soft decoding, which could further improve performance. The coded system is only used in this work as a guideline to interpret the differences in symbol error rates.

[7] Section 5.4.4 provides some insights into why the optimal λ_s is larger than the optimal λ_r, which may appear somewhat counterintuitive at first.

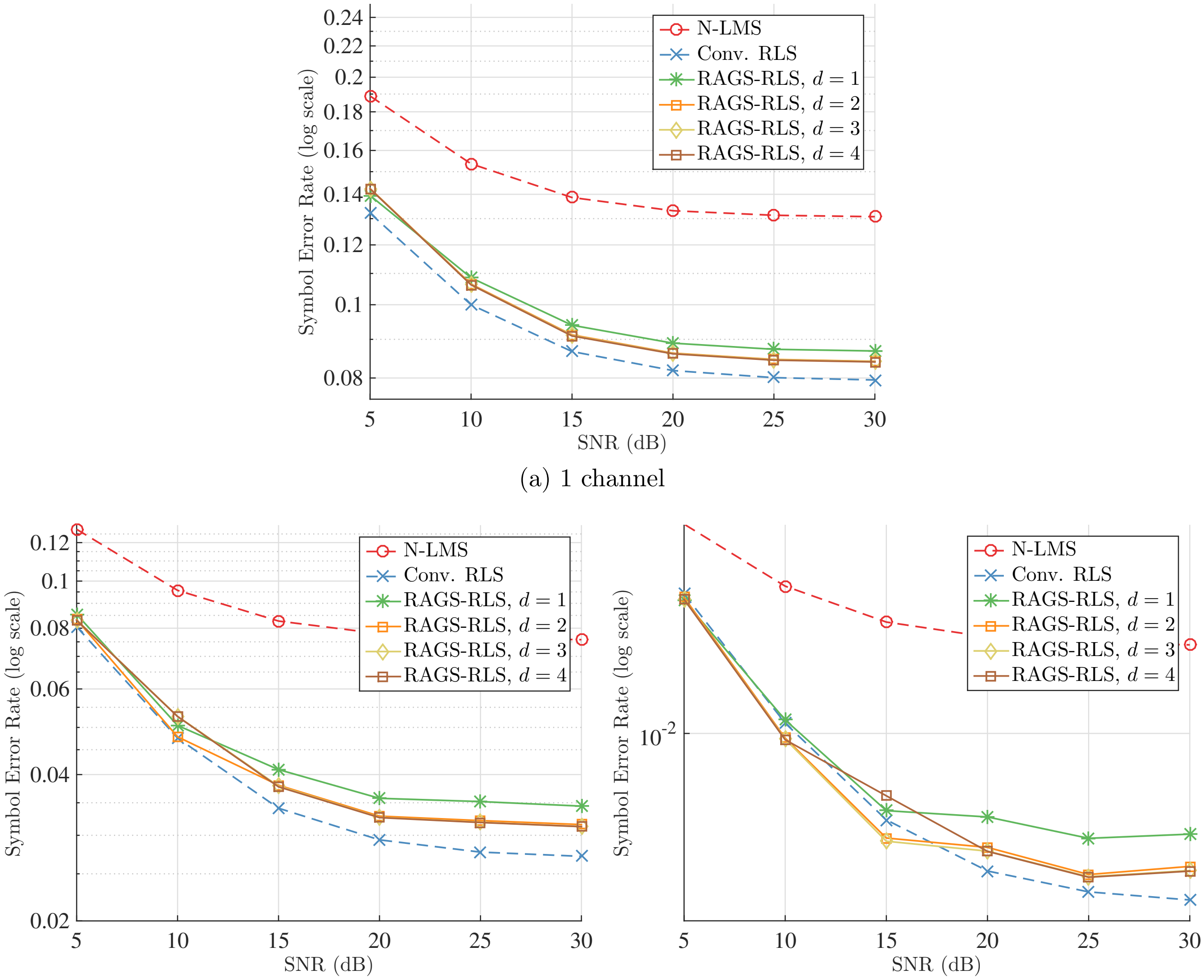


(a) 1 channel

(b) 2 channels

(c) 4 channels

Figure 5-7: Output prediction error (dB) for different adaptation algorithms used for adaptive equalization as a function of SNR (dB) with different numbers of channels ($R = 1, 2, 4$).

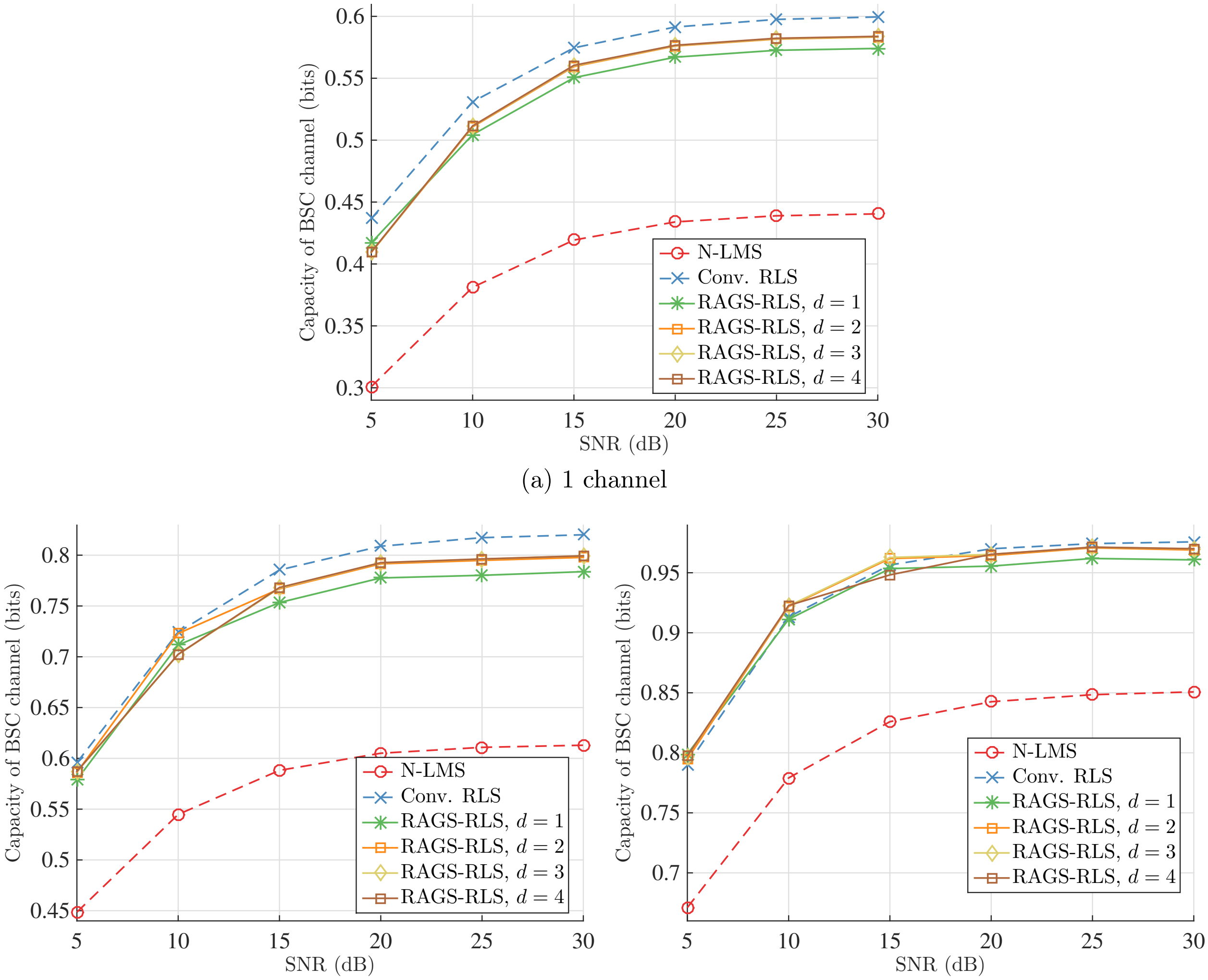


(a) 1 channel

(b) 2 channels

(c) 4 channels

Figure 5-8: Symbol error rate (log-scale) for different adaptation algorithms used for adaptive equalization as a function of SNR (dB) with different numbers of channels ($R = 1, 2, 4$).



(a) 1 channel

(b) 2 channels

(c) 4 channels

Figure 5-9: Binary symmetric channel capacity corresponding to symbol error rates of Figure 5-9, with adaptive equalizers using different adaptation algorithms as a function of SNR (dB) with different numbers of channels ($R = 1, 2, 4$). This represents the highest rate codes that could be used at the output of the adaptive equalizer to achieve error free communication (in theory).

Algorithm	$\frac{\text{Complexity of Algorithm}}{\text{Complexity of RLS}}$
N-LMS	0.0068
RAGS-RLS, $d = 1$	0.0387
RAGS-RLS, $d = 2$	0.1819
RAGS-RLS, $d = 3$	0.4449
RAGS-RLS, $d = 4$	0.8042

Figure 5-10: Complexity of various algorithms for adaptive equalization

Figure 5-9, respectively; and Figure 5-10 shows the complexities of N-LMS and RAGS-RLS in this application as a fraction of the complexity of the conventional RLS algorithm.

Clearly, while N-LMS is the fastest among all the algorithms, it is also the least effective. RAGS-RLS, on the other hand, has a performance very close to that of RLS. The performance of RAGS-RLS for $d = 1$ is worse at higher SNRs in comparison to that of RLS, whereas at low SNRs the performance is effectively identical. Note that when $R = 2, 4$ and in the low SNR regime, the RAGS-RLS algorithm can actually outperform conventional RLS in this application. At higher SNRs, the loss of performance ranges from 0.1 dB (with 1 channel) to about 0.6 dB with 4 channels, which translates into slight losses in the symbol error rate. The result that the performance gap between the conventional and RAGS-RLS algorithms narrows as SNR decreases matches the analytical intuition obtained from Section 5.3.1 that GS-LS outperforms LS at low SNRs due to the decreased importance of the effective noise issue at low SNRs.

The BSC capacities of Figure 5-9 show that the symbol error rate differences between RAGS-RLS and RLS correspond to very small differences in the code rate that would be required to achieve error-free communication across the channel when the two algorithms are used for equalization. Using $d = 1$, the difference in the code rate required in the worst case is 0.02, meaning that an increased redundancy of about 2 bits per 100 would be required for error-free communication when the equalization is performed with RAGS-RLS over when it is performed with RLS. To achieve error-free communication while using N-LMS for equalization, the amount of redundancy would have to be increased by 1 in 6 bits (the code rate would have to be increased by 18%).

The performance difference between RAGS-RLS and RLS can be further reduced by using $d = 2, 3, 4$. In these cases, RAGS-RLS has prediction MSE between 0.1 dB and 0.2 dB larger than RLS, which translates into the system using RAGS-RLS requiring 1 extra redundancy bit per 100 code bits to achieve error-free communication over the system using conventional RLS.

The performance of RAGS-RLS is evidently better when $d = 2$ that when $d = 1$, which indicates that the expected absolute covariance structure does have some value. There are a few statistical aberrations that cause a loss of performance when $d = 3, 4$—-these are most likely due to over-modeling. It is possible that when an edge is modeled that is not truly present in the data, the correlation due to the noise can cause performance degradation, which is likely the reason for the minor losses in the prediction error performance.

Considering the complexity gains that are indicated by Figure 5-10, it is clear that even with $d = 1$, it is possible to come close to the performance of RLS for a small fraction of the computational cost in an application where, heretofore, RLS has been the algorithm of choice. When $d = 2$, further improvement is demonstrated for a larger computational cost. There are diminishing returns for increased computation that may make it less appealing to increase d further, especially when the over-modeling issue is considered.

The complexity gains obtained herein are very compelling. The differences in the code-rates required for error-free communication between a RAGS-RLS and RLS based equalizer are so small that it is almost certain that in practical underwater acoustic communication, the same code would be used in the two cases. The lowered equalization complexity is desirable in and of itself, and, additionally, can lead to lower power requirements for communication, which is an extremely useful result in, for instance, mobile robotics and underwater sensor networks.

It has thus been shown in this chapter that it is possible to exploit the graphical model structure to reduce computation, in addition to improving performance as is done by GEM-LS. Particularly for the case of recursive estimation and tracking using the RAGS-RLS algorithm, there appears to be a compelling argument that if structure is available in the input, tracking can be done for a fraction of the computational cost with small losses in performance. This has been demonstrated in both theory and practice. Thus, structured inputs admit solutions for tracking problems that are simultaneously computationally efficient and accurate.

5.6 Some Comments and Looking Ahead

This chapter relaxes the GEM-LS algorithm to leverage the graphical structure to obtain a computationally simple method for estimation, termed the RAGS-LS algorithm. The recursive formulation of the RAGS-LS algorithm, termed RAGS-RLS is shown to be an efficient and powerful method to track time-varying systems. In doing so, the algorithms present methods that complete the set of potential objectives as stated in Section 2.3.1: between the GEM-LS, RAGS-LS and RAGS-RLS algorithm, methods are available to either improve the performance or reduce the complexity of estimation problems over state-of-the-art methods; and to perform recursive estimation and tracking very efficiently.

At this juncture, it is helpful to summarize the overall framework presented in this thesis for exploiting input graphical model structure in a given system identification application. The first step is to use the principles outlined in Section 3.1 to understand the inverse covariance structure of the data in a given application. As evidenced by Section 3.4, some applications may have structure when viewed in a suitable transform domain; and thus, it may be necessary to investigate the data quite thoroughly to convince oneself that graphical model structure exists in a particular application. Unfortunately, there is no "one-size-fits-all" solution to knowing whether graph structure exists in a particular application or not—finding the structure may simply be a matter of experience or physical insight. Of course, methods do exist for graph learning with some classes of data [23], [39], [114] and these can perhaps indicate the presence of consistently identifiable structure. That done, depending upon whether the objective is complexity reduction or accurate performance, a suitable algorithm can be chosen and applied using the techniques indicated in Chapters 4 and 5.

In the next chapter, the contributions of the thesis are summarized, and several open questions related to the work of the thesis are put forth. Finally, the thesis is placed in a broader framework for future study.

Chapter 6

Concluding Remarks

The problem of linear system identification and tracking has a long history; and yet many issues connected to this problem remain relatively unexplored. Primarily, solutions to this problem in the regime where limited or very noisy data is available revolve around the notion of constraining the solution space in which a search for the solution is conducted in some meaningful way.

The work of this thesis introduces a fundamentally novel way of constraining linear problems: that of reducing problem dimensionality using structure on the inputs. It has been shown that the presence of input structure has the potential to either improve the performance or reduce the complexity of estimating and tracking linear systems from their inputs and outputs.

In this final chapter of the thesis, the contributions of the thesis are first outlined and several open questions directly related to the thesis are raised. The work of the thesis can be placed in a broader context that leads to several other deep and interesting questions regarding learning and inference—these points are raised at the end of the chapter.

6.1 Summary of Contributions

The general contribution of the thesis is to show how structure on the input to a system can make identifying the system easier in several ways. The thesis begins by exploring what kind of structure is exploitable in this context.

While there are several kinds of structure that may potentially be manifest in data, Section 2.3 reveals that not every kind structure can easily be exploited in system identification. The main issue that arises is that any constraints placed on the inputs also affect the output of the system, and it is not always easy to enforce such constraints.

However, the notion of probabilistic graphical models allow such constraints to be carried through the system. Probabilistic graphical models capture conditional distribution structure and are a representation of how a joint distribution factors into functions over subsets of the random variables. Thus, it is possible to factor include the output of the system in the distribution being considered and account for the operation of the system in the context of graphical models.

In Chapter 3, several real-world system identification problems are described, and it is shown that each of these problems has inputs that are characterized by graphical models. The problems introduced are:

1. Initializing and tracking filters to cancel acoustic echo introduced by a reverberant environment, such as a room or automobile cabin. In this case, the underlying graph is a trivial graph with all independent nodes.

2. Identification of a fractionally spaced channel, where the graph that characterizes the data is a graph with several unconnected cliques.

3. Adaptive equalization in wireless communication systems. In this case, it can only be assumed that the input processes are cyclostationary, and it is not obvious that a graphical model characterization exists. However, it was shown that suitable graphical model structure exists in the frequency domain due to the spectral properties of cyclostationary processes. An adaptive multichannel equalization framework was developed in the frequency-domain that was able to exploit the structure.

A second contribution of Chapter 3 is the introduction of an approach to thinking about structure with finite data. Structure has previously rarely been modified to handle limited data. However, it is shown in Section 3.5 that, when the amount of data available is small, the structure (and in particular, the conditional independence relationships among the random variables) that is observed may deviate quite significantly from the expected ensemble structure. One way to handle that is simply to model the expected observed structure. The benefits of this approach are possibly debatable, but initial results in Section 5.5.2 show that better performance is obtained by modeling expected observed structure rather than ensemble structure. Further investigation of the issues surrounding this effect are certainly warranted.

Chapter 4 sets up the complete graph for the parameter identification problem and introduces the GEM-LS algorithm—the first algorithm that this thesis contributes. GEM-LS iterative algorithm based on the Expectation-Maximization framework. Various useful properties of the GEM-LS algorithm are proved, including:

- Under mild conditions, GEM-LS converges to a unique point regardless of the amount of data available.

- Under more stringent, but still reasonable conditions, the convergence of GEM-LS is unaffected by assuming that the input is characterized by an incorrect graphical model.

- The complexity of GEM-LS is at most as large as that of conventional LS.

- In the limit of infinite data (but with infinite noise), for a suitable choice of the iteration, the GEM-LS algorithm is able to outperform conventional LS by performing linear shrinkage towards the starting point of the algorithm.

- In the general case, GEM-LS performs shrinkage along a data drive curve that appears on average to be superior to performing shrinkage along a line. While proving this rigorously is challenging, insight into why is provided by viewing the overall problem as the combination of several smaller linear shrinkage problems.

The problem of which iteration to stop the GEM-LS algorithm at does not appear to have a simple solution (see Section 6.2), so the simple heuristic method of pre-computing an average optimum iteration is motivated and used. Using this method, the GEM-LS algorithm has excellent performance both in simulation and in the real-world applications of initializing AEC filters and channel identification when compared to conventional LS and various regularized LS solutions.

While the GEM-LS algorithm is useful for improving the performance of linear system identifiers, its complexity is, in general, no better than that of conventional LS[1] in the absence of parallelization (a parallel implementation can realize significant computational savings in theory). Moreover, it is not well-suited to the problem of tracking in a time-varying environment because of the lack of a recursive form (see Section 6.2).

A relaxation of the GEM-LS algorithm, termed the RAGS-LS algorithm is derived in Chapter 5 that exploits the graphical structure for computational benefits. RAGS-LS does not necessarily improve upon the conventional LS or regularized LS; but for graphs with small clique sizes, it is extremely computationally efficient. It was shown that the performance loss is linked to the fact that the model is not consistently enforced throughout the algorithm and manifests as an effective SNR degradation. As a result, the algorithm is forced to solve the small dimensional problems in a high noise environment, leading to a trade-off between the problem dimension and noise that favors the RAGS-LS algorithm (over conventional LS) only at very high SNRs or for very large problems.

With some modifications to the innovation, however, a recursive form of the algorithm, termed the RAGS-RLS algorithm, is shown to be very useful in the real world. In particular, it is shown that over sufficient time, RAGS-RLS converges to the same solution as does conventional RLS; but with considerably fewer operations. The work of Chapter 5 shows that the direct application of the results of Wiesel et al. [181] on the ML estimate of the structured covariance matrix leads to poor performance, but in the RLS framework changing the implementation can recover most if not all of the lost performance.

In the real-world problem of tracking adaptive equalizer coefficients for wireless underwater communication channels, RAGS-RLS is found to have performance close to that of conventional RLS for a small fraction of the computational cost. This has the potential to drive the next generation of underwater acoustic communication modems, as the redundancy required for error-free communication using RAGS-RLS for adaptation is nearly identical to that required for error-free communication using RLS; and in practice, the same error-correcting code would almost certainly be used in both the cases. Lower complexity

[1] It should be noted that the GEM-LS *can* be faster than conventional LS in some regimes. However, it complexity scales the same way as the conventional LS algorithm.

solutions are advantageous as they lead to lower power and faster receivers—key issues in underwater communication.

Moreover, by capturing the expected observed structure rather than the true distribution structure, it is possible to further reduce the performance gap between RAGS-RLS and RLS at the expense of some reduction in computational complexity gains, thereby indicating that the “expected absolute value” structure can be helpful in some applications.

The thesis thereby explores a fundamentally different approach towards constraining the linear learning problem and presents methods to exploit those constraints to improve the problem either from the point of view of reducing computation or reducing error.

6.2 Open Questions

Some major questions that arose during the work but which were not answered in the thesis are now briefly considered.

Frequency-Domain Covariance and Inverse Covariance Matrices with Finite Fourier Transform Length

While characterizing the frequency-domain covariance and inverse covariance matrices in Section 3.5.2, the following issue was raised: if $\mathcal{X}_\omega, \mathcal{X}_\nu$ are uncorrelated frequency coefficients and are approximated by $\mathcal{X}_\omega^{M_t}, \mathcal{X}_\nu^{M_t}$, which are frequency coefficients at ω, ν that are computed using a finite-length time-domain process of length M_t, then how does the correlation between $\mathcal{X}_\omega^{M_t}$ and $\mathcal{X}_\nu^{M_t}$ behave; and in particular, how does it vary with M_t?

It was verified in Figure 3-16 that the correlation falls of as $1/M_t$ for large M_t. However, a proof of this fact was not provided, primarily because the cases for which a proof could be found were too limited and not thought to be that useful. A proof of this or, more generally, an analysis of the behavior of the frequency-domain covariance matrix would be of great interest, as this matrix is extensively used in signal processing.

Similarly, the properties of the inverse covariance matrix from a more analytical viewpoint would also be of great interest. This is likely to be even more challenging than the properties of the covariance matrix, but even more useful.

Performance Analysis of GEM-LS and a Stopping Criterion

In Chapter 4, one of the key results was that GEM-LS can outperform conventional LS with a suitable choice of iteration: a property that was shown to be related to shrinkage. However, this immediately raises two closely related questions:

1. How much performance gain can be expected?

2. How can the optimum stopping iteration be found?

An analytical answer to the first question was provided only in the case when the number of available data samples N is very large, in which case it was shown that the performance gain is related to optimal linear shrinkage along the line from the starting point of the iterations to the eventual (LS) solution. The analysis in the general case, however, is hard because the update equation depends upon a random matrix $\boldsymbol{\rho}_{\boldsymbol{x}}(N)$ that is not easily mapped to any of the well-studied classes of random matrices.

However, such an analysis would be of great interest moving forward. It would help quantify what can be expected out of the GEM-LS algorithm and identify a variety of regimes in which performance gains are expected; and additionally provide insight into determining when to stop.

This brings us to the second of the problems stated above: how should we choose when to stop the GEM-LS algorithm? It was explained that the data-driven criteria that were investigated in Section 4.4.3 were not successful at predicting a good stopping point for the algorithm, so a simulation based look-up table approach was chosen. The problem with such an approach is that it is not as robust as would be ideal. Analytically, the stopping iteration appears to depend on the precise data inverse covariance matrix, which is unknown, and while this factor did not have a noticeable adverse effect on the performance of GEM-LS in practice, it cannot be guaranteed that this will always be the case.

Potential methods to stop the algorithm can come from investigating and exploiting the properties of the curve along which the GEM-LS solution proceeds, which relates this problem to the previous one. Alternately, it may be possible to intelligently divide the data and obtain meaningful solutions, potentially by combining more than one GEM-LS solution together. At all events, it appears likely that the analysis of performance will shed light on the issue of iteration choice.

The Absence of a "GEM-RLS"-Type Algorithm

The RAGS-LS algorithm of Chapter 5 is recast into a recursive framework; coupled with suitable windowing and modification to the innovation, it has application to time-varying systems. One may wonder why a similar result was not shown for the GEM-LS algorithm. Why not obtain a recursive framework for the GEM-LS algorithm, which would then be likely to improve upon the tracking performance of the conventional RLS for a suitable iteration?

Such an algorithm was in fact developed and tested. It is perhaps interesting to note that the so-called GEM-RLS algorithm with a single iteration is exactly the same as the RAGS-RLS algorithm, but with the innovation term multiplied by $1/(D+1)$.

However, GEM-RLS was not found to perform well while tracking time-varying systems. The precise reasons are unknown, but it is worth noting that GEM-LS is able to effect improvements by computing ever-improving guesses of the parameter, because it obtains ever-improvement estimates of *all* the clique outputs.

When cast in a recursive framework, as multiple iterations are run, the algorithm is forced to only improve its estimate of the clique outputs corresponding to the current input. Intuitively, it may then be expected that, in the long run, the best the algorithm can do is to reach the same steady state statistics as the RLS algorithm, as it is not allowed to go back and improve past clique output estimates and take advantage of those. Indeed direct averaging showed that for a suitable choice of forgetting factor (different from the forgetting factor for both RLS and RAGS-RLS) a recursive, exponentially weighted implementation of GEM-LS was exactly the same as RLS. In practice, it was found to perform rather worse for that choice of forgetting factor.

This does not account for the fact that the forgetting factor affects the averaging window—-the effect of this on the estimated inverse covariance matrix is ignored by the direct averaging analysis. In practice, to truly compare the algorithms, it would likely be necessary to either improve the analysis (which is likely to be a difficult, albeit interesting, question) or to resort to a simulation to obtain the "best" performance of GEM-RLS to compare with the optimal

RLS solution (under some fixed set of environmental conditions). This was not found to be possible within the time-constraints.

This raises several questions, the most fundamental of which is: is it possible at all to exploit input structure in some way to improve the performance (and not just the complexity) of tracking systems? While the intuition based on the results of this thesis is "probably yes," the question of how to do so remains an open one.

Further Applications

The thesis has, by and large, focused on applications where the underlying structure of the input distribution is a consequence of the fact that the input comprises successive samples of a random process and the properties of the underlying process lead to those samples being structured in some way.

This is truly a matter of convenience rather than of necessity, because research has shown that distribution structure is ubiquitous in the universe. Examples of applications where it has been suggested that the algorithms of this thesis are applicable include tracking stock prices (where multiple stocks are treated as the input to a system that generates future prices, and the stocks are linked to one another through a graph), massive arrays in environmental science (where the signal at elements of the array at a great distance may potentially have mutual independence properties), and so on.

Given the recent popularity of graphical models in representing distribution structure in a variety of field, we believe that this is a matter of casting the applications into a suitable framework as defined herein; and also that the algorithms and results of this thesis have wide applicability.

Incorporating Structure on the System

In this work, the issue of structure on the system has not been considered. In fact, imposing constraints on the system is an aspect of design that was specifically avoided, in order to isolate the benefits of exploiting input structure.

However, it is expected that very efficient and accurate estimation and tracking systems can be designed by simultaneously accounting for both input structure and system structure—whether that be in the form of regularization constraints, or graphical models, or by some other means; as it imposes further (hopefully meaningful) constraints on the problem.

Perhaps the easiest of such constraints to add to the framework of this thesis are, of course, graphical model constraints. That is, suppose the parameter being estimated can also be modeled as a realization from a distribution that exhibits structure. Then, the entire problem can be modeled using a large augmented graph with some hidden nodes (similar to the augmented graph of Section 4.1, but with nodes for the system as well).

A similar situation can apply to time-varying systems as well, if the underlying system and dynamical model for its evolution can be represented using graphs. This would likely require a dynamical graphical model framework **Casteights2012**, [8], [20]. Inference, estimation and tracking on such graphs presents an interesting set of issues.

Less clear is how constraints such as sparsity or bounded power (ℓ_2-norm) can be imposed on the problem while simultaneously handling input structure. Some ad-hoc methods have been encouraging—for example, improved performance can result from diagonally loading RAGS-LS solutions over each clique. The design and analysis of such systems will be of interest.

One caveat in this is that general wisdom holds that the more constraints that are imposed upon a problem, the less robust it generally is to mismodeling. This can be an issue when simultaneously attempting to account for structure on different parts of the problem. Imposing constraints suitably while simultaneously maintaining robustness is a problem of interest as well.

6.3 Broader Themes

The results of this thesis inform a variety of broader questions and raise issues that will likely be of interest to the signal processing and machine learning communities.

While this thesis has dealt with the issue of linear parameter identification, this can be viewed as a particular example of a learning problem. Broadly speaking, parameter learning is the problem of estimating an unknown distribution parameter from data relevant to the parameter.

In this thesis, input structure has been applied, which is a kind of *meta-data*, or information about data. This raises a very interesting and relatively unexplored question: how can meta-data affect learning in general? It seems intuitively clear that meta-data can affect how a learning system should model the problem of interest, as with the results of this work where the algorithms use the structure to (implicitly) split the linear problem into smaller problems. The linear problem considered here is special because the "sub-problem models" are just smaller versions of the original learning problem, but that may not be the case in general.

It seems that, as learning systems get increasingly sophisticated in the way that they handle and make inferences from data, the notion of how to use this kind of side-information has not received the same degree of attention. A truly intelligent system (such as a human being) would, of course, be able to reconcile all of these effects in a consistent manner. In design, however, the results of this work provide hints that incorporating meta-data even in limited ways can dramatically improve a system's ability to perform inference.

The algorithms presented here—in particular, the GEM-LS algorithm—as well as a number of parameter learning algorithms in literature raise the question of how well one can learn a linear system (or more generally, a system with some particular model). Evidently, a variety of algorithms can easily outperform maximum-likelihood (which for this problem is LS) in different regimes, whereas maximum-likelihood, from the point of view of just the observed data, is an optimal and Cramer-Rao bound achieving estimator. The difference, of course, is the set of assumptions that each of the other estimators make.

This leads to a rather general question: what kind of information is it meaningful to have access to? Is it possible to develop a consistent theory of what kind of information it is possible to exploit in learning and inference? This is related to the question often asked in information theory of "how much information is enough" for a particular task, but is perhaps more difficult to answer because it is difficult to quantify how much information is provided by a particular regularizer (say); or a particular kind of structure.

Nonetheless, this discussion indicates that an important set of questions underlies the basis of this thesis. Evidently, learning can be improved by restrictions placed on the data, or on the parameter being learned—or, more generally, by meta-data. The questions of what kinds of meta-data, how much they can help, and how they can be exploited appear to be at the heart of designing future generations of intelligent signal processing and learning systems.

Bibliography

[1] Abraham, D. A. and Owsley, N. L., "Preprocessing for High Resolution Beamforming," in *Proc. 23rd Asilomar Conf. Signals, Syst. and Comput.*, 1989, pp. 797–801.

[2] Abramovich, Y. I. *et al.*, "Sample-Deficient Adaptive Detection: Adaptive Scalar Thresholding Versus CFAR Detector Performance," *IEEE Trans. Aerospace and Electric Syst.*, vol. 46, no. 1, pp. 32–46, 2010.

[3] Akhtman, J. and Hanzo, L., "Decision Directed Channel Estimation Aided OFDM Employing Sample-Spaced and Fractionally-Spaced CIR Estimators," *IEEE Trans. Wireless Commun.*, vol. 6, no. 4, pp. 1171–1175, 2007.

[4] Akkarakaran, S. and Vaidyanathan, P. P., "Bifrequency and Bispectrum Maps: A New Look At Multirate Systems With Stochastic inputs," *IEEE Trans. Signal Process.*, vol. 48, no. 3, pp. 723–736, 2000.

[5] Allen, J. B. and Berkley, D. A., "Image Method for Efficiently Simulating Small-Room Acoustics," *J. Acoust. Soc. Amer.*, vol. 65, no. 4, pp. 943–950, 1979.

[6] Andersen, I. N., "Sample-Whitened Matched Filters," *IEEE Trans. Inform. Theory*, vol. 19, no. 5, pp. 653–660, 1973.

[7] Aster, R. *et al.*, *Parameter Estimation and Inverse Problems*, 2nd ed. Waltham, MA: Elsevier, Ltd., 2012.

[8] Bach, F. and Jordan, M., "Learning Graphical Models for Stationary Time Series," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2189–2199, 2004.

[9] Bajwa, W. *et al.*, "Compressed Channel Sensing: A New Approach to Estimating Sparse Multipath Channels," *Proc. IEEE*, vol. 98, no. 6, pp. 1058–1076, 2010.

[10] Bao, B. Y. Z. *et al.*, “Universality for the Largest Eigenvalue of Sample Covariance Matrices With General Population,” *Ann. Stat.*, vol. 43, no. 1, pp. 382–421, 2015.

[11] Barber, D. and Cemgil, A., “Graphical Models for Time Series,” *IEEE Signal Process. Mag.*, vol. Nov, pp. 18–28, 2010.

[12] Beeri, C. *et al.*, “On the Desirability of Acyclic Database Schemes,” *J. ACM*, vol. 30, no. 3, pp. 479–513, 1983.

[13] Bello, P., “Characterization of Randomly Time-Variant Linear Channels,” *IEEE Trans. Commun. Syst.*, vol. 11, pp. 360–393, 1963.

[14] Benesty, J. and Gay, S., “An Improved PNLMS Algorithm,” in *IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2002, pp. 1881–1884.

[15] Benesty, J. and Huang, Y., *Adaptive Signal Processing: Applications to Real-World Problems*. Berlin, Germany: Springer-Verlag, 2003.

[16] Berg, E. and Friedlander, M., “Probing the Pareto Frontier for Basis Pursuit Solutions,” *SIAM J. Computation*, vol. 31, no. 2, pp. 890–912, 2008.

[17] ——, “Sparse Optimization With Least-Squares Constraints,” *SIAM J. Optimization*, vol. 21, no. 4, pp. 1201–1229, 2011.

[18] Berger, C. R. *et al.*, “Application of Compressive Sensing to Sparse Channel Estimation,” *IEEE Commun. Mag.*, pp. 164–174, 2010.

[19] Betlehem, T. and Abhayapala, T. D., “Theory and Design of Sound Field Reproduction in Reverberant Rooms,” *J. Acoust. Soc. Amer.*, vol. 117, no. 4, pp. 2100–2111, 2005.

[20] Bilmes, J., “Dynamic Graphical Models,” *IEEE Signal Process. Mag.*, vol. 27, no. 6, pp. 29–42, 2010.

[21] Bishop, C. M., *Pattern Recognition and Machine Learning*. New York, NY: Springer-Verlag, 2006.

[22] Blair, B. J. S., “Analysis of and Techniques for Adaptive Equalization for Underwater Acoustic Communication,” Ph.D. Thesis, MIT/WHOI, 2011.

[23] Bresler, G., “Efficiently Learning Ising Models on Arbitrary Graphs,” in *Symp. Theory of Computation*, ACM Press, 2015, pp. 771–782.

[24] Buchner, H. and Spors, S., “A General Derivation of Wave-Domain Adaptive Filtering and Application to Acoustic Echo Cancellation,” in *Proc. 42nd Asilomar Conf. Signals, Syst. and Comput.*, 2008, pp. 816–823.

[25] Buck, J. R. and Wage, K. E., “A Random Matrix Theory Model for the Dominant Mode Rejection Beamformer Notch Depth,” in *IEEE Stat. Signal Process. Workshop (SSP)*, 2012, pp. 820–823.

[26] Bühlmann, P. and Geer, S. van de, *Statistics for High-Dimensional Data.* Berlin, Germany: Springer-Verlag, 2011.

[27] Cai, T. and Zhou, H., “Minimax Estimation of Large Covariance Matrices Under ℓ_1-Norm,” *Statista Sinica*, vol. 22, pp. 1319–1378, 2012.

[28] Candès, E. *et al.*, “Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information,” *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489 –509, 2006.

[29] Candès, E. J. and Wakin, M. B., “An Introduction To Compressive Sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. March 2008, pp. 21–30, 2008.

[30] Carbonelli, C *et al.*, “Sparse Channel Estimation With Zero Tap Detection,” *IEEE Trans. Wireless Commun.*, vol. 6, no. 5, pp. 1743–1753, 2007.

[31] Casteigts, A. *et al.*, “Time-varying graphs and dynamic networks,” *Int. J. Parallel, Emergent and Distributed Syst.*, vol. 27, no. 5, pp. 387–408, 2012.

[32] Chandar, V. *et al.*, “A Simple Message-Passing Algorithm for Compressed Sensing,” in *IEEE Int. Symp. Information Theory*, 2010, pp. 1968–1972.

[33] Chaudhuri, S. *et al.*, “Estimation of a Covariance Matrix With Zeros,” *Biometrika*, vol. 94, no. 1, pp. 199–216, 2007.

[34] Chen, S. S. *et al.*, “Atomic Decomposition by Basis Pursuit,” *SIAM J. Scientific Computing*, vol. 20, pp. 33–61, 1998.

[35] Chen, S. and Donoho, D. L., “Basis Pursuit,” in *Asilomar Conf. Signals, Syst. and Comput.*, 1994, pp. 41–44.

[36] Chhetri, A. S. *et al.*, “Acoustic Echo Cancellation for High Noise Environments,” in *IEEE Int. Conf. Multimedia and Expo.*, 2006, pp. 905–908.

[37] Chitre, M. *et al.*, “Underwater Acoustic Channel Characterisation for Medium-Range Shallow Water Communications,” in *Proc. OCEANS ’04*, vol. 1, 2004, pp. 40–45.

[38] Chitre, M. *et al.*, “Recent Advances in Underwater Acoustic Communications & Networking,” in *Proc. OCEANS ’08*, Quebec City, 2008, pp. 1–10.

[39] Chow, C. and Liu, C., “Approximating Discrete Probability Distributions with Dependence Trees,” *IEEE Trans. Inform. Theory*, vol. 1, no. 3, pp. 462–467, 1968.

[40] Cioffi, J. M., “Limited-Precision Effects in Adaptive Filtering,” *IEEE Trans. Circuits and Syst.*, vol. 34, pp. 821–833, 1987.

[41] Copas, J. B., “Regression, Prediction and Shrinkage,” *J. Royal Stat. Soc. Series B (Methodological)*, vol. 45, no. 3, pp. 311–354, 1983.

[42] Cotter, S. F. *et al.*, “Sparse Solutions to Linear Inverse Problems With Multiple Measurement Vectors,” *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2477–2488, 2005.

[43] Couillet, R. and Debbah, M., *Random Matrix Methods for Wireless Communication.* Cambridge, U.K.: Cambridge University Press, 2011.

[44] Cover, T. M. and Thomas, J. A., *Elements of Information Theory*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2006.

[45] Cowell, R. E. *et al.*, *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks.* Berlin, Germany: Springer-Verlag, 1999.

[46] Date, P. *et al.*, “Filtering and Forecasting Commodity Futures Prices Under an HMM Framework,” *Energy Economics*, vol. 40, pp. 1001–1013, 2013.

[47] Davenport, W. B. and Root, W. L., *An Introduction to the Theory of Random Signals and Noise.* Hoboken, NJ: Wiley-IEEE Press, 1958.

[48] Deistler, M., "System Identification and Time Series Analysis: Past, Present, and Future," in *Stochastic Theory and Control*, Pasik-Duncan, B., Ed., Berlin, Germany: Springer-Verlag, 2002, pp. 97–109.

[49] Dempster, A., "Covariance Selection," *Biometrics*, vol. 28, no. 1, pp. 157–175, 1972.

[50] Dempster, A. *et al.*, "Maximum Likelihood From Incomplete Data Via the EM Algorithm," *J. Royal Stat. Soc. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[51] Detert, T. *et al.*, "Fractional Spaced Channel Estimation and Shortening for Joint Delayed Decision Feedback Sequence Estimation," in *IEEE Veh. Technology Conf.*, 2006, pp. 2167 –2172.

[52] Dokmanic, I. and Vetterli, M., "Room Helps: Acoustic Localization With Finite Elements," in *IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2012, pp. 2617–2620.

[53] Donoho, D. L. *et al.*, "Message-Passing Algorithms for Compressed Sensing," *Proc. Nat. Academy Sciences United States Amer.*, vol. 106, no. 45, 18 914–âĂŞ18919, 2009.

[54] Duttweiler, D. L., "Proportionate Normalized Least-Mean-Squares Adaptation in Echo Cancelers," *IEEE Trans. Speech and Audio Process.*, vol. 8, no. 5, pp. 508–517, 2000.

[55] Efron, B. *et al.*, "Least Angle Regression," *Ann. Stat.*, vol. 32, no. 2, pp. 407–499, 2004.

[56] Efron, B. and Morris, C., "Stein's Paradox in Statistics," *Scientific American*, vol. 236, pp. 119–127, 1977.

[57] Eleftheriou, E. and Falconer, D. D., "Tracking Properties and Steady-State Performance of Rls Adaptive Filter Algorithms," *IEEE Trans. Acoustics, Speech, and Signal Process.*, vol. 34, no. 5, pp. 1097–1110, 1986.

[58] Elliott, R. J. *et al.*, "Financial Signal Processing: A Self-Calibrating Model," *Int. J. Theoretical and Applied Finance*, vol. 04, no. 4, pp. 567–584, 2001.

[59] Falconer, D. *et al.*, "Frequency Domain Equalization for Single-Carrier Broadband Wireless Systems," *IEEE Commun. Mag.*, pp. 58–66, 2002.

[60] Fisher, R., "On an Absolute Criterion for Fitting Frequency Curves," *Messenger Math.*, vol. 41, pp. 155–160, 1912.

[61] Florencio, D. and Zhang, Z., "Maximum A-Posteriori Estimation of Room Impulse Responses," in *IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2015, pp. 728–732.

[62] Flynn, C. J. *et al.*, "On the Sensitivity of the LASSO to the Number of Predictor Variables," in *Joint Statistical Meetings: Section on Statistical Learning and Data Mining*, Montreal, Canada, 2013. [Online]. Available: http://arxiv.org/abs/1403.4544.

[63] Folland, G. B., "Section 2.2: A Convergence Theorem," in *Fourier Analysis and Its Applications*, Providence, RI: American Mathematical Society, 1992, ch. 2, pp. 42–49.

[64] Foster, D. P. and George, E. I., "The Risk inflation Criterion for Multiple Regression," *Ann. Stat.*, vol. 22, no. 4, pp. 1947–1975, 1994.

[65] Franke, J., "Inverse Least Squares and Classical Least Squares Methods for Quantitative Vibrational Spectroscopy," in *Handbook of Vibrational Spectroscopy*, Hoboken, NJ: John Wiley & Sons, 2006. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/0470027320.s4603/.

[66] Friedman, J. *et al.*, "Sparse Inverse Covariance Estimation With the Graphical LASSO," *Biostatistics (Oxford, England)*, vol. 9, no. 3, pp. 432–441, 2008.

[67] Gardner, W. A. and Franks, L. E., "Characterization of Cyclostationary Random Processes," *IEEE Trans. Inform. Theory*, vol. I, no. 1, pp. 4–14, 1975.

[68] Gardner, W. A. *et al.*, "Cyclostationarity: Half a Century of Research," *Signal Process.*, vol. 86, pp. 639–697, 2006.

[69] Geman, S, "A Limit Theorem for the Norm of Random Matrices," *Ann. Probability*, vol. 8, no. 2, pp. 252–261, 1980.

[70] Gladyshev, E., "Periodically and Almost-Periodically Correlated Random Processes With Continuous Time Parameter," *Theory of Probability & Its Applicat.*, vol. 8, no. 2, pp. 173–177, 1963.

[71] Goransson, B. *et al.*, "Spatial and Temporal Frequency Estimation of Uncorrelated Signals Using Subspace Fitting," in *8th IEEE Signal Process. Workshop Stat. Signal and Array Process.*, Corfu, 1996, pp. 94–96.

[72] Gustafsson, T. *et al.*, "Estimation of Acoustical Room Transfer Functions," in *39th IEEE Conf. Decision and Control*, Sydney, Australia, 2000, pp. 5184 –5189.

[73] Habets, E. A., "Room Impulse Response Generator," Department Elektrotechnik-Elektronik-Informationstechnik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, Tech. Rep., 2010.

[74] Hammersley, J. M. and Clifford, P, *Markov Fields on Finite Graphs and Lattices*, Unpublished Manuscript, 1971. [Online]. Available: http://www.statslab.cam.ac.uk/~grg/books/hammfest/hamm-cliff.pdf.

[75] Hassibi, B. *et al.*, "H^∞ Optimality of the LMS Algorithm," *IEEE Trans. Signal Process.*, vol. 44, no. 2, pp. 267–280, 1996.

[76] Hastie, T. *et al.*, *The Elements of Statistical Learning*, 2nd ed. New York City, NY: Springer Publishing, 2009.

[77] Hayes, M. H., *Statistical Digital Signal Processing and Modeling*. Hoboken, NJ: John Wiley & Sons, 1996.

[78] Haykin, S., *Adaptive Filter Theory*, 4th ed. New York City, NY: Pearson Education, 2002.

[79] Heidemann, J. *et al.*, "Underwater Sensor Networks: Applications, Advances and Challenges," *Philosoph. Trans. Royal Soc. A: Math, Physical and Engineering Sciences*, vol. 370, pp. 158–175, 2012.

[80] Henniger, J, "Functions of Bounded Mean Square and Generalized Fourier-Stieltjes Transforms," *Canadian J. Math.*, vol. 22, no. 5, pp. 1016–1034, 1970.

[81] Hero, A. *et al.*, "Highlights of Statistical Signal and Array Processing," *IEEE Signal Process. Mag.*, vol. 15, pp. 21–64, 1998.

[82] Herrholz, E. and Teschke, G., "Compressive Sensing Principles and Iterative Sparse Recovery for Inverse and Ill-Posed Problems," *Inverse Problems*, vol. 26, no. 12, 2010.

[83] Ho, B. and Kalman, R. E., “Editorial: Effective Construction of Linear State-Variable Models From Input/Output Functions,” *at-Automatisierungstechnik*, vol. 14, no. 1-12, pp. 545–548, 1966.

[84] Hoerl, A. E. and Kennard, R. W., “Ridge Regression: Biased Estimation for Nonorthogonal Problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[85] Hsieh, C.-j. *et al.*, “QUIC: Quadratic Approximation for Sparse Inverse Covariance Estimation,” *J. Mach. Learning Research*, vol. 15, pp. 2911–2947, 2014.

[86] Hsu, S.-h. *et al.*, “Online Recursive independent Component Analysis for Real-Time Source Separation of High-Density EEG,” in *36th Annu. Int. Conf. IEEE Eng. in Medicine and Biology Society*, 2014, pp. 3845–3848.

[87] Hurd, H. L., “An investigation of Periodically Correlated Stochastic Processes,” Ph.D. Thesis, Duke University, 1969.

[88] ——, “Correlation Theory of Almost Periodically Correlated Processes,” *J. Multivariate Analysis*, vol. 37, pp. 24–45, 1991.

[89] James, W. and Stein, C., “Estimation With Quadratic Loss,” in *Proc. 4th Berkeley Symp. Math. Stat. and Probability*, vol. 1, 1961, pp. 361–379.

[90] Jaynes, E., “Information Theory and Statistical Mechanics,” *Physical Review*, vol. 106, no. 4, pp. 620–630, 1957.

[91] Jeon, S. *et al.*, “Haptic Augmentation in Soft Tissue Interaction,” in *Multisensory Softness*, Di Luca, M., Ed., London, U.K.: Springer-Verlag, 2014, ch. 12, pp. 241–257.

[92] Johnson, C. C. *et al.*, “High-Dimensional Sparse Inverse Covariance Estimation Using Greedy Methods,” in *Proc. 15th Int. Conf. Artifical Intelligence and Stat.*, 2012, pp. 574–582.

[93] Johnstone, I. M. and Silverman, B. W., “Empirical Bayes Selection of Wavelet Thresholds,” *Ann. Stat.*, vol. 33, no. 4, pp. 1700–1752, 2005.

[94] Kalman, R., “A New Approach to Linear Filtering and Prediction Problems,” *J. Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[95] Klipp, E. *et al.*, *Systems Biology.* Hoboken, NJ: John Wiley & Sons, 2013.

[96] Koller, D. and Friedman, N., *Probabilistic Graphical Models: Principles and Techniques*, 1st ed. Cambridge, MA: The MIT Press, 2009.

[97] Kraay, A. and Baggeroer, A., "A Physically Constrained Maximum-Likelihood Method for Snapshot-Deficient Adaptive Array Processing," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4048–4063, 2007.

[98] Lauritzen, S. L., *Graphical Models.* Oxford, U.K.: Oxford University Press, 1996.

[99] Letac, G and Massam, H, "All Invariant Moments of the Wishart Distribution," *Scandinavian J. Stat.*, vol. 31, no. 2, pp. 295–318, 2004.

[100] Li, H. *et al.*, "Computationally Efficient Maximum Likelihood Estimation of Structured Covariance Matrices," *IEEE Trans. Signal Process.*, vol. 47, no. 5, pp. 1314–1323, 1999.

[101] Ljung, L., "Perspectives on System Identification," *Annu. Reviews in Control*, vol. 34, pp. 1–12, 2010.

[102] Lopes, P. A. and Gerald, J. B., "New Normalized LMS Algorithms Based on the Kalman Filter," in *IEEE Int. Symp. Circuits and Syst.*, 2007, pp. 117 –120.

[103] Lopes, P. A. and Piedade, M. S., "A Kalman Filter Approach to Active Noise Control," in *10th European Signal Process. Conf.*, 2000, pp. 2937–2942.

[104] Maiwald, D. and Kraus, D., "Calculation of Moments of Complex Wishart and Complex Inverse Wishart Distributed Matrices," *IEE Proc. Radar, Sonar and Navigation*, vol. 147, no. 4, pp. 162 –168, 2000.

[105] Malioutov, D. *et al.*, "A Sparse Signal Reconstruction Perspective for Source Localization With Sensor Arrays," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 3010–3022, 2005.

[106] Mamon, R. S. *et al.*, "Adaptive Signal Processing of Asset Price Dynamics With Predictability Analysis," *Inform. Sciences*, vol. 178, pp. 203–219, 2008.

[107] Manolakis, D. G. *et al.*, *Statistical and Adaptive Signal Processing: Spectral Estimation, Signal Modeling, Adaptive Filtering and Array Processing*. Norwood, MA: Artech House, 2000.

[108] Marsaglia, G., "Choosing a Point From the Surface of a Sphere," *Ann. Math. Stat.*, vol. 43, no. 2, pp. 645–646, 1972.

[109] Martinsson, P. *et al.*, "A Fast Algorithm for the inversion of General Toeplitz Matrices," *Computers & Math. with Applicat.*, vol. 50, pp. 741–752, 2005.

[110] Marvasti, F. *et al.*, "A Unified Approach to Sparse Signal Processing," *EURASIP J. Advances in Signal Process.*, vol. 44, no. 1, pp. 1–45, 2012.

[111] Matus, F., "On Equivalence of Markov Properties over Undirected Graphs," *J. Applied Probability*, vol. 29, no. 3, pp. 745–749, 1992.

[112] Matz, G. and Hlawatsch, F., "Fundamentals of Time-Varying Communication Channels," in *Wireless Communication Over Rapidly Time-Varying Channels*, Hlawatsch, F. and Matz, G., Eds., Waltham, MA: Elsevier, Ltd., 2011, ch. 1, pp. 1–63.

[113] Mehlig, B and Chalker, J, "Statistical Properties of Eigenvectors in Non-Hermitian Gaussian Random Matrix Ensembles," *J. Math. Physics*, vol. 41, p. 3233, 2000.

[114] Meinshausen, N and Bühlmann, P, "High-Dimensional Graphs and Variable Selection with the Lasso," *Ann. Stat.*, vol. 34, no. 3, pp. 1436–1462, 2006.

[115] Meng, Z. *et al.*, "Distributed Learning of Gaussian Graphical Models Via Marginal Likelihoods," *J. Mach. Learning Research*, vol. 31, pp. 39–47, 2013.

[116] Mitra, U. *et al.*, "Structured Sparse Methods for Active Ocean Observation Systems With Communication Constraints," *IEEE Commun. Mag.*, vol. 53, pp. 88–96, 2015.

[117] Molisch, A. F., "Ultra-Wide-Band Propagation Channels," *Proc. IEEE*, vol. 97, no. 2, pp. 353–371, 2009.

[118] Montanari, A., "Graphical Model Concepts in Compressed Sensing," in *Compressed Sensing, Theory and Applications*, Cambridge, U.K.: Cambridge University Press, 2012, ch. 9, pp. 394–438.

[119] Morelli, M. *et al.*, "Channel Estimation for Adaptive Frequency-Domain Equalization," *IEEE Trans. Wireless Commun.*, vol. 4, no. 5, pp. 2508–2518, 2005.

[120] Nadakuditi, R. R. and Silverstein, J. W., "Fundamental Limit of Sample Generalized Eigenvalue Based Detection of Signals in Noise Using Relatively Few Signal-Bearing and Noise-Only Samples," *IEEE J. Select. Topics in Signal Process.*, vol. 4, no. 3, pp. 468–480, 2010.

[121] Nadakuditi, R. R., "Applied Stochastic Eigen-Analysis," Ph.D. Thesis, MIT/WHOI, 2007.

[122] Napolitano, A., "Generalized Almost-Cyclostationary Processes and Spectrally Correlated Processes: Two Extensions of the Class of the Almost-Cylostationary Processes," in *9th Int. Symp. Signal Process. and Its Applicat.*, Sharjah, UAE, 2007, pp. 1–6.

[123] ——, *Generalizations of Cyclostationary Signal Processing: Spectral Analysis and Applications*. Hoboken, NJ: Wiley-IEEE Press, 2012.

[124] Natarajan, B., "Sparse Approximate Solutions to Linear Systems," *SIAM J. Computation*, vol. 24, no. 2, pp. 227–234, 1995.

[125] Neeser, F. and Massey, J., "Proper Complex Random Processes With Applications to Information Theory," *IEEE Trans. Inform. Theory*, vol. 39, no. 4, pp. 1293–1302, 1993.

[126] Newey, W. K. and McFadden, D. L., in *Handbook of Econometrics, Volume 4*, Engle, R. and McFadden, D. L., Eds., Amsterdam, Netherlands: Elsevier B.V., pp. 2111–2245.

[127] Nuttall, J. and Willett, P., "Adaptive-Adaptive Subarray Narrowband Beamforming," in *IEEE Int. Conf. Acoust., Speech, and Signal Process.*, 1993, pp. 305–308.

[128] Ottersten, B. *et al.*, "Covariance Matching Estimation Techniques for Array Signal Processing Applications," *Digital Signal Process.*, vol. 8, pp. 185–210, 1998.

[129] Pajovic, M., "The Development and Application of Random Matrix Theory in Adaptive Signal Processing in the Sample Deficient Regime," Ph.D. Thesis, MIT/WHOI, 2014.

[130] Pajovic, M. and Preisig, J. C., “Performance Analysis of the Least Squares Based LTI Channel Identification Algorithm Using Random Matrix Methods,” in *Proc. 49th Annu. Allerton Conf. Commun., Control, and Comput.*, 2011, pp. 516–523.

[131] ——, “Performance Analysis of Least Squares-Based Multichannel Decision Feedback Equalization of Time-Varying Channels,” in *IEEE Global Telecommun. Conf. (GLOBECOM)*, 2013, pp. 3312–3317.

[132] Papp, J. C. *et al.*, “Physically Constrained Maximum Likelihood Mode Filtering,” *J. Acoust. Soc. Amer.*, vol. 127, no. September 2009, pp. 2385–2391, 2010.

[133] Park, S. *et al.*, “Gaussian Assumption: The Least Favorable but the Most Useful,” *IEEE Signal Process. Mag.*, vol. 30, pp. 183–186, 2013.

[134] Pejoski, S. and Kafedziski, V., “Method for Direction of Arrival Estimation of Uncorrelated Wideband Signals Based on Sparse Covariance Fitting,” *Telfor J.*, vol. 6, no. 2, pp. 115–120, 2014.

[135] Peligrad, M. and Wu, W. B., “Central Limit Theorem for Fourier Transforms of Stationary Processes,” *Ann. Probability*, vol. 38, no. 5, pp. 2009–2022, 2010.

[136] Pfanzagl, J., “Consistent Estimators,” in *Parametric Statistical Theory*, Berlin, German: De Gruyter, 1994, ch. 6, pp. 187–224.

[137] Picard, J. S. and Weiss, A. J., “Direction Finding of Multiple Emitters by Spatial Sparsity and Linear Programming,” in *9th Int. Symp. Commun. and Inform. Technology*, 2009, pp. 1258–1262.

[138] Plackett, R., “Studies in the History of Probability and Statistics XXIX: the Discovery of the Method of Least Squares,” *Biometrika*, vol. 59, no. 2, pp. 239–251, 1972.

[139] Poulsen, A. J. *et al.*, “Robust Adaptive Vector Sensor Processing in the Presence of Mismatch and Finite Sample Support,” in *5th IEEE Sensor Array and Multichannel Signal Process. Workshop*, 2008, pp. 473–477.

[140] Pourahmadi, M., “Covariance Estimation: the GLM and Regularization Perspectives,” *Stat. Science*, vol. 26, no. 3, pp. 369–387, 2011.

[141] Preisig, J. C. *et al.*, "Reduced Bandwidth Frequency Domain Equalization for Underwater Acoustic Communications," in *6th IEEE Sensor Array and Multichannel Signal Process. Workshop*, 2010, pp. 93–96.

[142] Proakis, J., *Digital Communication*, 4th ed. New York City, NY: McGraw-Hill, 2001.

[143] Reed, I. *et al.*, "Rapid Convergence Rate in Adaptive Arrays," *IEEE Trans. Aerospace and Electronic Syst.*, vol. AES-10, no. 6, pp. 853–863, 1974.

[144] Reina, G. *et al.*, "Adaptive Kalman Filtering for GPS-Based Mobile Robot Localization," in *IEEE Int. Workshop Safety, Security and Rescue Robotics*, 2007, pp. 1–6.

[145] Ribeiro, F. *et al.*, "Using Reverberation to Improve Range and Elevation Discrimination for Small Array Sound Source Localization," *IEEE Trans. Audio, Speech and Language Process.*, vol. 18, no. 7, pp. 1781–1792, 2010.

[146] Rice, S. O., "Mathematical Analysis of Random Noise," *Bell System Tech. J.*, vol. 23, no. 3, pp. 282 –332, 1945.

[147] Sasiadek, J. Z. *et al.*, "Fuzzy Adaptive Kalman Filtering for INS/GPS Data Fusion," in *Proc. 15th IEEE Int. Symp. Intelligent Control (ISIC 2000)*, 2000, pp. 181–186.

[148] Sayed, A. H., *Adaptive Filters*. Hoboken, NJ: John Wiley & Sons, 2008.

[149] Schlee, F. H. *et al.*, "Divergence in the Kalman Filter," *AIAA J.*, vol. 5, no. 6, pp. 1114–1120, 1967.

[150] Schmandt, B. and Clayton, R. W., "Analysis of Teleseismic P-Waves With a 5200-Station Array in Long Beach, California: Evidence for an Abrupt Boundary to Inner Borderland Rifting," *J. Geophysical Research: Solid Earth*, vol. 118, pp. 5320–5338, 2013.

[151] Sclove, S. L., "Improved Estimators for Coefficients in Linear Regression," *J. American Stat. Assoc.*, vol. 63, no. 322, pp. 596–606, 1968.

[152] Selfon, S., *''XBox, Listen": driving gameplay with Kinect audio input and speech recognition*, Presented at Game Developers Conf. 2011. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=26102.

[153] Sen Gupta, A. and Preisig, J., “A Geometric Mixed Norm Approach to Shallow Water Acoustic Channel Estimation and Tracking,” *Physical Commun.*, vol. 5, no. 2, pp. 119–128, 2012.

[154] Sherman, J. and Morrison, W. J., “Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix,” *Ann. Math. Stat.*, vol. 21, no. 1, pp. 124–127, 1950.

[155] Silverstein, J. W. and Tulino, A. M., “Theory of Large Dimensional Random Matrices for Engineers,” in *IEEE Int. Symp. Spread Spectrum Techniques and Applicat.*, vol. 2, 2006, pp. 458–464.

[156] Skretting, K. and Engan, K., “Recursive Least Squares Dictionary Learning Algorithm,” *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2121–2130, 2010.

[157] Song, M. S. *et al.*, “An Interactive 3-d Audio System With Loudspeakers,” *IEEE Trans. Multimedia*, vol. 13, no. 5, pp. 844–855, 2011.

[158] Stein, C., “Inadmissibility of the Usual Estimator for the Mean of a Multivariate Normal Distribution,” in *Proc. 3rd Berkeley Symp. Math. Stat. and Probability*, 1956, pp. 197–206.

[159] Stigler, S. M., “Gauss and the Invention of Least Squares,” *Ann. Stat.*, vol. 9, no. 3, pp. 465–474, 1981.

[160] Stoica, P. *et al.*, “SPICE: a Novel Covariance-Based Sparse Estimation Method for Array Processing,” *IEEE Trans. Signal Process.*, vol. 59, no. 2, pp. 629–638, 2011.

[161] Stojanovic, M., “OFDM for Underwater Acoustic communications: Adaptive Synchronization and Sparse Channel Estimation,” in *IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2008, pp. 5288–5291.

[162] Strohmer, T., “Measure What Should be Measured: Progress and Challenges in Compressive Sensing,” *IEEE Signal Process. Letters*, vol. 19, no. 12, pp. 887–893, 2012.

[163] Stuart, A. and Ord, J. K., *Kendall'S Advanced Theory of Statistics, Vol. 1*, 5th ed. London, U.K.: Edward Arnold Publishers Ltd., 1994.

[164] Takashima, R. *et al.*, "Single-Channel Sound Source Localization Based on Discrimination of Acoustic Transfer Functions," in *Advances in Sound Localization*, Strumillo, P., Ed., Rijeka, Croatia: InTech, 2011, ch. 3, pp. 39–54.

[165] Tan, I. *et al.*, "Measurement and Analysis of Wireless Channel Impairments in DSRC Vehicular Communications," in *IEEE Int. Conf. Commun.*, 2008, pp. 4882–4888.

[166] Tao, T. and Vu, V., "Random matrices: Universal Properties of Eigenvectors," *Random Matrices: Theory and Applicat.*, vol. 1, p. 25, 2011.

[167] Tarantola, A., 2nd ed. Philadelphia, PA: Society for Industrial and Applied Math.

[168] Taubock, G., "A Maximum Entropy Theorem for Complex-Valued Random Vectors, With Implications on Capacity," in *IEEE Inform. Theory Workshop*, 2011, pp. 375–379.

[169] Tibshirani, R, "Regression Shrinkage and Selection Using the LASSO," *J. Royal Stat. Soc. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[170] Tibshirani, R. and Wasserman, L., *Sparsity and the LASSO*, Class notes, Statistical Machine Learning, Carnegie Mellon University, 2014. [Online]. Available: http://www.stat.cmu.edu/~larry/=sml/sparsity.pdf.

[171] Tikhonov, A. N., "On the Stability of Inverse Problems," *Doklady Akademii Nauk SSSR*, vol. 39, no. 5, pp. 195–198, 1943.

[172] Tomasi, B. *et al.*, "A Study on the Wide-Sense Stationarity of the Underwater Acoustic Channel for Non-Coherent Communication Systems," in *European Wireless*, Vienna, Austria, 2011, pp. 500–505.

[173] Tropp, J. A. and Wright, S. J., "Computational Methods for Sparse Solution of Linear Inverse Problems," *Proc. IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[174] Tulino, A. M. and Verdú, S., *Random Matrix Theory and Wireless Communications*. Delft, Netherlands: Now Publishers Inc, 2004.

[175] Turin, G. L., "An Introduction to Matched Filters," *IRE Trans. Inform. Theory*, vol. 6, no. 3, pp. 311–329, 1960.

[176] Van Trees, H. Hoboken, NJ: John Wiley & Sons, Inc.

[177] Verdú, S. and Poor, H. V., "On Minimax Robustness: A General Approach and Applications," *IEEE Trans. Inform. Theory*, vol. 30, no. 2, pp. 328–340, 1984.

[178] Vershynin, R., "introduction to the Non-Asymptotic Analysis of Random Matrices," in *Compressed Sensing, Theory and Applications*, Eldar, Y. C. and Kutyniok, G., Eds., Cambridge, U.K.: Cambridge University Press, 2012, ch. 5, pp. 210–268.

[179] Wang, J. *et al.*, "On Spectral Theory of Cyclostationary Signals in Multirate Systems," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2421–2431, 2005.

[180] Werner, K. *et al.*, "On Estimation of Covariance Matrices With Kronecker Product Structure," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 478–491, 2008.

[181] Wiesel, A. *et al.*, "Covariance Estimation in Decomposable Gaussian Graphical Models," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1482–1492, 2010.

[182] Wiesel, A. and Hero, A. O., "Distributed Covariance Estimation in Gaussian Graphical Models," *IEEE Trans. Signal Process.*, vol. 60, no. 1, pp. 211–220, 2012.

[183] Williams, V. V., "Multiplying Matrices Faster Than Coppersmith-Winograd," in *Proc. 44th Annu. ACM Symp.*, 2012, pp. 887–898.

[184] Wu, W. B., "Fourier Transforms of Stationary Processes," *Proc. Amer. Math. Soc.*, vol. 133, no. 1, pp. 285–293, 2004.

[185] Xiao, H. and Wu, W. B., "Covariance Matrix Estimation for Stationary Time Series," *Ann. Stat.*, vol. 40, no. 1, pp. 466–493, 2012.

[186] Yardibi, T. *et al.*, "Source Localization and Sensing: A Nonparametric Iterative Adaptive Approach Based on Weighted Least Squares," *IEEE Trans. Aerospace and Electric Syst.*, vol. 46, no. 1, pp. 425–433, 2010.

[187] Yellepeddi, A. and Florencio, D., "Sparse Array-Based Room Transfer Function Estimation for Echo Cancellation," *IEEE Signal Process. Letters*, vol. 21, no. 2, pp. 230–234, 2014.

[188] Yellepeddi, A. and Preisig, J. C., "Adaptive Equalization in a Turbo Loop," *IEEE Trans. Wireless Commun.*, vol. 14, no. 9, pp. 5111–5122, 2015.

[189] Zhou, S. *et al.*, "Time-Varying Undirected Graphs," *Mach. Learning*, vol. 80, no. 2, pp. 295–319, 2010.